無閉路有向グラフの最適系列分割問題に関する研究

加地太一

①

無閉路有向グラフの最適系列分割問題に関する研究

加地太一

平成９年４月

目次

第１章 序論

１．１ 本研究に関する背景

グラフのノードに関して、与えられた目的関数を最適化するような分割を求める問題をグラフ分割問題と呼ぶ。グラフ分割問題は情報科学、システム工学、オペレーションズ・リサーチ、電気工学、応用数学分野などの研究テーマとして長年にわたり多くの研究者達によって取り扱われてきた。重要な研究テーマとされてきた理由として、多くの応用問題への利用性と、その簡明さのためにアルゴリズム研究での一つの試金石となりえたためである。

グラフ分割問題はそれぞれの応用の場面にあわせて各種の形態が考えられている。基本的な最小カットを求める２分割問題[7],[16]から始まり、分割に対して要求される部分集合の数を k とする k 分割問題へと展開される[8]。さらに、部分集合のサイズの制約が組み込まれる有界集合への最小カット問題などがあげられる[1],[5]。また、分割対象となるグラフに対して制約を加える、あるいは、特定することにより各種各様の問題[10]へと展開されてきた。

グラフ分割問題はいくつかの重要な問題に対して、基本的な概念を提供し、多くの応用問題に利用されてきた。一つの主要な応用問題として VLSI の設計の問題があげられる[9]。VLSI の設計において、複雑な電気回路がシリコンウェハー上に構成され一つのチップを作り上げる。この複雑な電気回路は多くのモジュールからなり、それらの間には物理的な結線で連結され互いに関連して配置される。この VLSI の設計では各結線の全長を短くし、必要とされるシリコンウェハーの領域を最小化することが課題となる。これは各モジュールを効果的にグループ化し配置することによって、この課題の解決へとつながる。ここで、

モジュールをノードとし電気的結線をエッジとするならば、この問題はグラフ分割問題へと帰着される[4]。また、その他にも施設配置問題[2]、プログラム分割問題[11]、構造モデル分析問題[15],[12]など多様な問題へ利用されている。

このグラフのノードの部分集合へのクラスタリングは組合せ問題へとつながる。グラフ分割分割問題は Garey, Jonson, Stockmeyer[5]によって NP 困難な問題と証明されたが、すでに多くの研究者達はその問題が手に負えない困難な問題であることを確信していた。それゆえに、小さな問題に対して、あるいはグラフに特定の構造をもつ問題に対しては分枝限定法[6]や動的計画法[10]を用いて厳密解を得ることができたが、多くの大きな問題に対してはヒューリスティックなアルゴリズムを用いた研究へと移っていった。すなわち、厳密な最適解を求めるのではなく最良な近似解を探索する方策をとる戦略を用いる。その代表的なヒューリスティックアルゴリズムは Kernighan-Lin のアルゴリズムであり、その性能については評価が高い。また、近年にいたっては各種のヒューリスティックな戦略を統合化することによって、さらに強力なアルゴリズムへと展開されてきた[13],[14]。

一方、最適化問題とは、最も適切な計画、設計、方策などを作成し、または選択する問題を対象とする。最適化は極めて普遍的な問題であり、歴史的にも人類の活動とともに始まったともいえる。その中で、いくつかの問題が数理的に取り扱われたのも新しい話ではない。しかし、今日言うような最適化問題が明確に意識され、数理的にまとまった体系を形づくるとともに、広範囲な応用分野が開拓されだしたのは，１９４０年代後半にオペレーションズ・リサーチの研究が始まった頃であるとみてよい。そして，１９５０年代後半にコンピュータが実用化されるとともに、最適化の研究は質的にも量的にも、飛躍的な発展を逐げた。それは、最適化の計算は、コンピュータの能力をまってはじめて実用的な意味で可能になったからである。また逆に、VLSI の設計、ハードウェ

ア、ソフトウェアの設計、ジョブ割り当てなどにも欠かせない道具となり、情報科学・工学にとっても最適化は重要な基礎をなすものである。

特に、対象が組合せ的・離散的な条件の下での最適化を、組合せ最適化問題として取り扱う。対象が組合せ的・離散的であるとき、独自の視点が要求される。すなわち、これらの問題は全ての可能解を列挙し、その中から最適解を導き出すという自明なアルゴリズムが存在するが、実用性の観点からは無意味である。組合せ的爆発により、もはや計算機が扱える範囲を超えてしまうからである。したがって、すべてを列挙することなく最適なものをいかに効率よく見出すかがポイントとなる。

組合せ最適化問題では厳密解を求める場合、膨大な数の可能解の中から何らかの方法で最適解を探索するというアプローチがとられる．最適解を見失うことなく、探索領域をいかに限定するかが重要であって、この目的に動的計画法(dynamic programming)や分枝限定法(branch-and-bound)などが用いられる。

しかし、組合せ最適化問題の多くは本質的に複雑度が高く、厳密な最適解を効率よく求めるのは困難である。だが、最適値に近い値をもつ可能解でよければ効率よく求め得る可能性があり、実用上大きな意味がある。近似最適解をうまく求めるには、問題の解や構造に関する知識をいかに利用するかがポイントになるのでヒューリスティック解法と呼ばれる。さらに、近年ではそのヒューリスティックな知識を組み合わせてより高度なアルゴリズムを構成するためのメタ戦略の研究が盛んに行われている。その中には局所探索法、Simulated Annealing 法、遺伝アルゴリズム、Tabu Search 法など、最近話題のアプローチも含まれている。

１．２ 本研究の目的

本研究の目的は多くの工学的問題への応用が期待される無閉路有向グラフの

系列分割問題を提案し、その解法を研究することである。

１９７０年代にKernighanはグラフの頂点の番号が連続的に保持される特殊化した無向グラフ（一列化グラフ）に対して、分割される各頂点部分集合がその番号の連続性を保持し、かつ、その大きさがそれぞれある上限数以下に限定される条件の下で、カットエッジのコストの総和を最小化する“一列化グラフの系列分割問題”を提示した。この問題はコンピュータのプログラムをある決まったサイズのページ・セグメントの集まりに、ページ間の制御移動頻度を最小化するように分割配置するプログラム分割問題として提示された。

しかし、Kernighanの系列分割問題では、ある与えられた一列化グラフ上で系列を保存する分割問題を考えているが、これに対して多くのシステム問題ではその構造はより一般的な無閉路有向グラフで構成されており、その上で系列保存の性質をもつ要素集合の分割問題を考えることによって、より応用性の広い多様な問題が扱えるようになる筈である。そこで、無閉路有向グラフの場合に問題を拡張し、無閉路有向グラフの系列分割問題を提案することにする。このためには、半順序集合に準拠する理論展開と問題記述の様式の確立が有効であると予想される。さらに、これらの準備のもとで、 Kernighanの系列分割問題を無閉路有向グラフの場合に一般化し、無閉路有向グラフ上での最適系列分割問題を定義する。この種の問題としては無閉路有向グラフの構造を利用するスケジューリング問題、ラインバランシング問題などが挙げられるが、一つの具体的事例として次のような並列処理の問題も考えられる。

問題：先行順位を持ついくつかの生産プロセスからなるシステムが与えられているとする。各要素プロセスは、その先行プロセスの生産物を入力として受け取り生産活動を行い、その産出物をそれを必要とする後続プロセスに渡す。また、各プロセスは生産活動のために一定の資源量を必要とするが、１ステーシ

ョンあたり使用できる資源量は定まっているとする。中間製品の移動には各ステーション間で、ある輸送コストを必要とするが、同一ステーション内の輸送コストは無視できるものとする。このとき各生産プロセスを、輸送コストの総和が最小になるように、先行順位の制限を無視せずに、ステーションごとに利用可能な資源量の範囲内で、各ステーションに配置せよ。

この問題の解法を考えるにあたって、基本となる一列化グラフの系列分割問題に対して考えなければならない。すなわち、この問題が以後提案される解法の構成要素となり、その中で多数回反復して用いられることにより、その特性を明らかにし、あわせて改良が望まれる。この一列化の問題に対して、Kernighanはその特殊化されたグラフの構造に着目して、動的計画法の考えを用い、グラフのエッジ数に線形な計算時間で計算可能な厳密解法を構成している。しかし、Kernighan はそのアルゴリズム全体の詳細な計算量、および、その特性については大きくふれてはいない。そこで著者は、このアルゴリズムが今後の研究展開において演ずる基本的な役割にかんがみ、この問題の詳細な評価検討を実験的および理論的に進める。

また、kernighan はこのアルゴリズムの構成に動的計画法を用いているが、最適性の原理が適用可能であるとき、動的計画法と分枝限定法には大きな共通点があることが知られており、動的計画法の計算手順は分枝限定法の一種とみなされる。この観点から見ると kernighan のアルゴリズムは必ずしもその探索法および限定操作の構築において最良の方策が採られているわけではなく、さらに改善の余地が残されているものと考えてよい。この方針に従い、効率のよいアルゴリズムの実現と適正な性能評価を目指した研究開発を試み、以後の研究の準備とする。

本研究で提案する問題の解法の研究を進めるには、動的計画法，分枝限定法

などを適用して、この提案された問題に対して、より効果的な厳密解法を導く研究開発を行い、実用的なアルゴリズムを確立するとともに、その性能を評価し利用限界を明らかにしたい。

しかし、無閉路有向グラフの系列分割問題に対する厳密解法は、普通のグラフ分割問題と同様に、ある特定の構造を持つグラフに対しては非常に有効な解法となりえても、ランダムな構造をもつ一般のグラフ構造に対しては計算量が指数的オーダーで増大し実質的な計算が困難となる場面が当然予想される。そこで次の課題として、ランダムで巨大な無閉路有向グラフにも充分対応可能な近似解法の導出が必要となる。ここではTabu Search法やSimulated Annealing法に代表されるメタヒューリスティックのパラダイムに基づく近代的な近似解法の枠組みで、効率的な手法を考案するとともにその性能評価を行うことによって、本問題の安定で効率のよいアルゴリズムの確立に寄与することを目標として研究を進めたい。

１．３ 本研究の構成

本論文は大きく分けて三部から構成されている。まず第一部では、本研究で必要となる組合せ最適化問題の諸解法の概略を説明している。次に第二部では系列分割問題の構成について述べており、ここで本研究の起点となり、重要な構成要素でもあるKernighanの研究の紹介と、著者による無閉路有向グラフの系列分割問題の提案とその定式化に関する研究が第３、４章に示されている。最後に第三部では、系列分割問題の解法を扱っている。まず、 Kernighanが提示した一列化グラフの最適系列分割算法の動作解析と、分枝限定法による算法の改善と評価に関する研究を第５章に与え、続いて、著者が提案する無閉路有向グラフの最適系列分割問題の厳密解法および近似解法に関する研究を第６、7章にわたって述べている。以上、序章と結論を含めて論文は以下の8章から構成され

ており、その各章の詳細を以下に述べる。

第１章では本研究に関する背景として、グラフ分割問題と最適化問題の大まかな経緯と概要を述べ、本研究にいたる目的および本論文の構成を記述した。

第２章では本研究で使用する組合せ最適化へのアプローチとして用いられた種々の解法について詳細な説明を加える。まず、厳密解を得るためのアプローチとして代表的な解法である動的計画法と分枝限定法について基本的な説明を与える。両者とも膨大な数の可能解の中から、何らかの手段を用いて探索領域を限定して厳密解をいかに効果的に見出すかがその主要な問題点であり、その組合せ問題への適用に関して、多くの戦略の研究工夫がなされてきた。しかし、組合せ最適化問題の多くはNP困難な問題のクラスに属し、これらの工夫にもかかわらず、厳密な最適解を実用時間内で求めることは一般に極めて困難である。これに代わって近似最適解を求める戦略が考えられ、数々のヒューリスティックなアルゴリズムが開発されてきたが、特に近年はこれらの知識を組み合わせてより高度なアルゴリズムを構成するメタヒューリスティックと呼ばれる解法が多くの成果を上げている。ここではその代表的な手法であるTabu Search法とSimulated Annealing法について、その基本となる近傍構造の定義と局所探索法を含めて詳細な説明を加える。

第３章では、本研究の起点であり、また同時に本研究で提案する問題の構成要素でもあるKernighanの問題についてふれる。すなわち、連続した頂点列をもつ無向グラフに対して、その系列性を保存し、あるブロックサイズ以下に、カットエッジのコストの総和を最小に分割する“一列化グラフの系列分割問題”についての詳細な定義と定式化の説明を行う。さらに、Kernighanによって提案された動的計画法にもとづくアルゴリズムを紹介する。最後に、その解法の特性と解法の計算時間がエッジ数に線形に比例するというKernighanの主張についての説明を加える。

第４章では一列化グラフの最適系列分割問題を拡張し、多くのシステム構造に対応でき、汎用性の高い無閉路有向グラフの最適系列分割問題を提案する。ここでは一列化グラフの系列分割問題の特徴的な性質を取り出し、無閉路有向グラフの場合に一般化し、無閉路有向グラフの系列分割を定義する。また、その性質を検討し、切断の鎖によって無閉路有向グラフの系列分割を一意に表現できることを示す。さらに、切断を簡潔に表現する切断決定子を導入し、切断の鎖を切断決定子列で表すことによって、最適系列分割問題の明確な定式化が可能となることを述べる。

第５章では本問題の構成要素であり、以後の解法の中で多数回反復使用される一列化グラフの最適系列分割問題の解法に対しての分析とその改良アルゴリズムを示す。まず、Kernighan の解法に対して、Kernighan が示すところの特性に関して、さらに詳細な計算時間の評価式を導出し、各種パラメータとの関連において計算時間特性のパフォーマンスについての検討を試みる。そこでは、理論的ならびに実験的な両側面から分析することにより、Kernighan アルゴリズムの挙動について多く知見と今後の指針を得ている。さらに、Kernighan が用いた動的計画法の構成において、必ずしも最良の探索法を用いていないこと、およびブロックサイズによる限定操作の利用の観点からも、標準的な動的計画法の上に組み立てられた Kernighan のアルゴリズムは改善可能と考えられる。そこで、動的計画法と多くの共通点をもち、柔軟に各戦略を組み込みやすい分枝限定法を利用した、より効率的なアルゴリズムの提案と実現について述べるとともに、この提案アルゴリズムの特性と性能について詳細な検討を加え、有効な構成要素となりうることを確信している。

第６章では本研究で提案した無閉路有向グラフの最適系列分割問題に対する有効な厳密解法を構築する。この解法では、まず、すべての切断からなる探索空間を切断決定子を節点とする多分岐平衡木で表現する。この木は、与えられ

た切断決定子から１レベル高い切断決定子を導く基本操作を用いた分枝規則の組織的な適用によって、空の木から高速に生成することができる。次に無駄のない効率的な探索方法を実現するために木を縮約し既約グラフとする。この既約グラフ上で動的計画法にもとづき、この問題の特徴的な性質を加味した最適化算法を構成する。さらに並列に近い構造のグラフに対して、この算法の計算量が多項式オーダーになることについて述べる。

第７章では、無閉路有向グラフの最適系列分割問題に対するメタヒューリスティックなパラダイムを用いた二つの近似解法を提案する。この問題の厳密解法では、グラフが並列構造をもつとき、その計算量は多項式オーダーであり実用時間内での計算の可能性があるが、無閉路有向グラフが巨大でランダムな構造を有するとき指数的オーダーの計算量となり実質的に計算が極めて困難となる。そこで、巨大でランダムな無閉路有向グラフにも対応できるよう、従来の組合せ最適化問題を解くための種々の戦略を有機的に結合させたメタヒューリスティックによる近似解の導出を試みる。メタヒューリスティックの解法としてTabu Search 法、Simulated Annealing 法、遺伝アルゴリズムなどが知られている。初めに、Tabu Search 法による適用を試みるが、本問題が複雑度の高いグラフ分割問題のため、標準的な Tabu Search 法の適用では効果的な結果は得られにくい。そこで、多重的な頂点移動による多様性と部分的最適化を組入れることによる収束性の強化を取り入れた複合移動を用いて Tabu Search 法の性能を引き出す試みを行っている。次に、Tabu Search 法の局所解に落ち込みやすい欠点を補うため、確率的な要素を取り入れた複合移動を用い Simulated Annealing 法の適用を試みている。これらのアルゴリズムに対する数値実験を行いその性能を評価するとともに、両者のアルゴリズムに有効であった複合移動のメタヒューリスティックへの一般化を示す。

第８章では、本研究の全体的なまとめと今後の研究課題について論じている。

1．4 まとめ

第一章は序章であり、グラフ分割問題とそれに関連する組合せ最適化問題に関するこれまでの研究を概観することによって、本研究の背景を説明するとともに、本研究の目的となる検討課題を整理している。それに基づき検討課題をまとめると次のようになる。

(1) 要素間に先行順位関係を持つシステムの構造は無閉路有向グラフで表現される。 Kernighan の一列化グラフの最適系列分割問題の考えをこのようなシステムに一般化できれば、より広範で興味のある問題を取り扱うことができる。このような方向への研究は従来まだ行われていない。

(2) Kernighan の最適系列分割問題の研究では、そのアルゴリズムの具体的で詳細な計算量や、それから導かれる動作特性についてはあまり深く検討されていないが、このアルゴリズムが今後の研究展開で演ずる基本的な役割にかんがみ、その検討をしておく必要がある。

(3) 最適性の原理が適用可能であるとき、最適性の原理と分枝限定法における優越関係には大きな共通点があり、動的計画法の計算手順は分枝限定法による構成法の一種であると見なされる。この観点に立てば、 Kernighan のアルゴリズムは必ずしもその探索法および限定操作の構築において最良の方策がとられているわけだはなく、さらに改善の余地が残されていると考えてよい。

(4) 無閉路有向グラフ上で系列分割を定義し、最適系列分割を正確に定式化するための数学的な道具を準備することが必要である。その上で厳密解を求める算法を構成しなければならない。

(5) 無閉路有向グラフの最適系列分割問題は計算困難な問題に属し、厳密解法の計算時間が実際的な計算時間内で終了しない場面が当然考えられる。これに対応するため、ヒューリスティックな知識を用いたメタ戦略による近似解法

を採用することになるが、この問題では分割数やブロック・サイズが決定されるべき変数となるので、有効な近傍構成に困難な問題が生じ、これを何らかの方法で解決しなければならない。

参考文献

[1] Barnes, E.R. : An algorithm for partitioning the nodes of a graph, SIAM J. Alg. Disc. Meth. 3(4), pp541-550(1982).

[2] Dai, W. and Kuh,E. : Simultaneous floor planning and global routing for hierarchical building block layout, IEEE Trans. Comput. Aided Design ICs Syst.(1987).

[3] Feo,T.A. and Khellaf,M. : The Complexity of Graph Partitioning, Manuscript, Operations Research Group, Department of Mechanical Engineering, University of Texas at Austin(1988).

[4] Feo,T.A. and Khellaf,M. : A Class of Bounded Approximation Algorithms for Graph Partitioning, Networks, Vol.20,pp181-195(1990).

[5] Garey,M.R., Johnson,D.S. and Stokmeyer,L.S. : Some simplified NP-complete graph problems, Theor. Comput. Sci. 1(1),pp237-267(1976).

[6] Gorinshteyn,L.L. : The partitioning of graphs, Eng. Cybern. 7,1,pp.76-82(1969).

[7] Johnson,D.S., Aragon,C.R., Mcgeoch,LA.. and Schevon, C. : Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning, Operations Research,Vol.37,No.6,pp.865-892(1989).

[8] Johnson, D.S., Papadimitriou,C.H., Seymour,P. and Yannakakis,M. : The complexity of multiway cuts. Manuscript, AT&T Bell Laboratories, Murray Hill, NJ(1984).

[9] Kernighan,B.W. and Lin,S. : An Efficient Heuristic Procedure for Partitioning Graph, Bell Syst. Tech. J. 49, pp.291-307(1970).

[10] Kernighan,B.W. : Optimal Sequential Partitions of Graphs, J.ACM, Vol.18, No.1, pp.34-40(1971).

[11] Kral,J. : To the problem of segmentation of a program. Info. Process. Machines(1965).

[12]Ohuchi,A. and Kaji,T. : An Idea Processor of the FISM/KJ Method, Proc. of 1992 IEEE Intern. Conf. on System, Man, and Cybernetics, pp.1448-1451(1992).

[13]茨木 : 離散最適化法とアルゴリズム (岩波講座 応用数学) 、岩波書店(1993).

[14]久保 : メタヒューリスティックス, 離散構造とアルゴリズムIV, 近代科学社, pp.171-230(1995).

[15]寺野 : システム工学入門, 共立出版(1985).

[16]藤沢、久保、森戸 : Tabu Search のグラフ分割問題への適用と実験的解析, 電学論 c ,Vol.114,No.4,pp.430-437(1994).

第２章　組合せ最適化問題

モータやポンプのような機器，装置の類を設計するような場合，あるいはより複雑な集積回路，コンピュータや通信のネットワークを計画あるいは設計するような場合、われわれはそれらの計画や設計を何らかの意味で最も適切に行いたいと考えるであろう。物を作る場合だけでなく，たとえば個人の限られた費用の有効な使い方、企業での生産や販売の有利な方策、あるいは国の望ましい土地利用や経済運用のあり方を考えるといったことも、しばしば現れる重要な問題である。このように、最も適切な計画、設計、方策などを作成し、または選択することを最適化（optimaization）と呼ぶ。

工学の本質は対象に対する操作（オペレーション）の研究であるから、最適化は工学において基本的なしかも中心的な課題として登場する。その他、医学、農学などの白然系の分野、また経済学や行動科学といった人文・社会系の分野でも、それはしばしば深い興味の対象とされている。最適化は何も学問上の特別な問題というわけではない。日常生活で，たとえば費用や時間の使い方，行動の計画や決定といったことは、（潜在的なことが多いにしても）絶えずわれわれの関心事となり、また悩みの種ともなっている。限られた時間や財産を最大限有効に活用して，なるべく大きな満足を得たいとは、誰しもが常に望んでいることなのである。つまり最適化は，われわれにとって極めて板元的でかつ普遍的な問題といってよい。そして一方では、合理的な最適化や最適決定は、適な情報の習得と処理によってはじめて可能になるという意味で、すぐれて情報科学的な問題である。

特に、この最適化問題において、基礎となる空間や可能領域が組合わせ的であるとき、組合わせ最適化問題（combinatorial optimization problem）という。組

合わせ的とは、離散量や有限（あるいは可算無限）集合のもつ数学的性質を指す。離敵的最適化（discrete optimization）や離散的計画法（discrete programming）も同様の意味をもつ．具体的な対象には、離散変数をもつ整数計画問題、グラフやネットワークに関する最適化問題、有限個の作業の最適順序を定めるすスケジューリング問題、n 地点の最適な訪問順序を求める巡回セールスマン問題などきわめて広範囲にわたる。組合せ最適化には、整数計画法、ネットワーク計画法、マトロイド理論、分枝限定法、動的計画法、線形計画法などの成果が広く利用されている。また、これらの多くの問題は実質的な時間内で計算困難な NP のクラスに属する。そのため、良好な近似解で代替し問題の解とする各種の近似解法の研究が近年盛んに行われている。

本研究においても、この組合わせ最適化問題の一つであるグラフの分割問題に関しての研究を行う。本章ではこの研究で利用する数種の厳密解法と近似解法の基本を、数理的に整理して考察する。

２．１ 厳密解法

２．１．１ 動的計画法

動的計画法（dynamic programming）は、１９５０年代後半に R. Bellman によって開拓され、組合わせ最適化問題のみならず、制御問題、配分問題、平滑問題、マルコフ決定問題など広汎な領域にわたる応用が研究されている手法である。その本質はいわゆる最適性の原理(principle of optimality)を利用して、計算効率を高める点にある。

最初に最適性の原理を抽象的な形で述べ、以下資源配分問題の例を通して理解をはかる。決定(decision)を一次元の系列に並べたものを方策(policy)といい、ここでは組合わせ最適化を、最適な方策を見出す問題としてとらえる。適用の

方向に応じて、２種の記述が可能である。

最適性の原理（順方向） 最適方策では中間の状態が何であっても、その状態を生じさせた最適方策の前半部分が、最初の状態からその状態を得る方策のうちで最適であるという性質。

最適性の原理（逆方向） 最適方策では、最適方策の前半が何であっても、その後半部分は、前半部分から生じた状態に関して最適方策を構成しているという性質。

最適性の原理は、すべての問題に対して成立するわけではなく、また同じ問題であっても、状態のとらえ方などに応じて異なってくる。最適性の原理が成立すれば、すべての方策のうちで最適方策になりうるものを相当限定することが可能になり、計算の手間を節約できる。そのための考え方および計算手順を動的計画法と呼ぶ。

動的計画法の適用例として、以下の資源配分問題を考えよう。

$$\max \quad z = \sum_{j=1}^{N} f_j(x_j)$$

$$subject\ to \quad \sum_{j=1}^{N} x_j = M \tag{2.1}$$

x_j : 非負整数　　$(j = 1, 2, \ldots, N)$

これは総量Mの離散的な資源をN種の活動に最適配分する問題である。各$f_j(x_j)$は一次関数でさえあればよく、特に仮定を設けない。整数計画問題の一種であるが、ごく簡単な制約条件を一個だけもつのが特徴である。マンパワー・プランニング、投資計画、生産計画、コンピュータの記憶領域の最適利用、最適観

測計画等の簡単な場合を含む。

この問題に動的計画法を適用するために

$$F_k(m) = \max\{\sum_{j=1}^{k} f_j(x_j) | \sum_{j=1}^{k} x_j = m, x_j: \text{ 非負整数}\} \qquad (2.2)$$

$$k = 1, 2, \ldots, N; m = 0, 1, \ldots, M)$$

とおこう。ここで、$F_N(M)$が求める最適値である。$F_k(m)$の計算法を与える漸化式は、

$$F_1(m) = f_1(m) \qquad (\mathrm{m} = 0, 1, \ldots, M)$$

$$F_{k+1}(m) = \max_{0 \le i \le m}(F_k(m-i) + f_{k+1}(i)) \qquad (2.3)$$

$$(k = 1, 2, \ldots, N\text{-}1; m = 0, 1, \ldots, M)$$

である。初期条件$F_1(m)$を与える(2.3)の第一式の正当性はその定義よりあきらかである。第２式はx_{k+1}に総量mの資源のうちiだけ割り当てるとすれば、残量m-iは$x_1, x_2, \ldots, x_k$に振り分けられる。x_{k+1}による目的関数値への貢献は$f_{k+1}(i)$であり、$x_1, x_2, \ldots, x_k$による貢献分$\sum_{j=1}^{k} f_j(x_j)$の最大値は$F_k(m\text{-}i)$である。つまり、$x_{k+1}=i$を仮定する場合、両者の和$F_k(m\text{-}i) + f_{k+1}(i)$が目的関数の最大値$F_{k+1}(m)$を与える。$i$の値は前もってわかるわけではないので、可能なすべての値を考慮して、(2.3)の第２式を得る。これは最適性の原理の成立を示している。

(2.3)の計算は$k = 1, 2, \ldots, N$-1 の順に、それぞれmのすべての値に対し$F_{k+1}(m)$を求めていけばよい。全計算量は、$f_j(x_j)$の計算を$N(M+1)$回、加算を$NM(M+1)/2$回、２数の比較を$NM(M\text{-}1)/2$回である。$\sum x_j = M$を満たすすべての整数ベクトルの個数とくらべるとはるかに小さく、列挙法より効率が良い。

動的計画法の場合、一般的形式では目的関数zが

$$f_j(x_1, \ldots, x_j) = f_j(f_{j\text{-}1}(x_1, \ldots, x_{j\text{-}1}), x_j) \qquad (j=2, 3, \ldots, N)$$

$$z = f_N(x_1, \ldots, x_N) \qquad (2.4)$$

の形に分離され、しかも、各$f_j(f_{j\text{-}1}, x_j)$が、$f_{j\text{-}1}$に関して非減少であれば十分である。

このとき、(2.2)内の $\sum_{j=1}^{k} f_j(x_j)$ を $f_k(x_1, \ldots, x_k)$に置き換えて $F_k(m)$を定義すると(2.3)は

$$F_1(m) = f_1(m) \qquad (m = 0, 1, \ldots, M)$$
$$F_{k+1}(m) = \max_{0 \le i \le m} f_{k+1}(F_k(m-i), i) \tag{2.5}$$
$$(k = 1, 2, \ldots, N\text{-}1;\ m = 0, 1, \ldots, M)$$

となる。すなわち、$x_{k+1}=i$ のとき、f_{k+1}の第一成分f_kに関する単調性のために、第一成分をその最大値 $F_k(m\text{-}1)$に固定すればよく、他の可能性は考慮しなくてよい。これが最適性の原理である。

また、逆方向の最適性の原理に関しても同様な議論が可能である。(2.2)に対応して

$$G_k(m) = \max\{\sum_{j=k}^{N} f_j(x_j) | \sum_{j=k}^{N} x_j = m,\ x_j:\ \text{非負整数}\}$$
$$(k=1, 2, \ldots, N;\ m=0, 1, \ldots, M) \tag{2.6}$$

とおけば、(2.3)に相当する。

$$G_N(m) = f_N(m) \qquad (m=0, 1, \ldots, M)$$
$$G_{k-1}(m) = \max_{0 \le i \le m}(f_{k-1}(i) + G_k(m-i)) \tag{2.7}$$
$$(k=2, 3, \ldots, N;\ m=0, 1, \ldots, M)$$

が得られ、$G_1(M)$が求める最適値である。$G_k(m)$の計算は(2.3)とは逆方向となり、$k=N, N\text{-}1, \ldots, 1$ の順に求めることとなる。

２．１．２ 分枝限定法（branch-and-bound）

条件を満足する解を求めることが目的であり、解になりえないという判断がついたところでそれ以上の探索を打ち切って後戻りを行い探索を実行することをバックトラックという。バックトラックの解の探索過程は、木の形で表現で

きるが、ここで分枝というのは、解を段階的に決める際、ある段階から次の下位段階へ行くときに、すべての可能性をあげ、枝分かれして探索することをいう。そして、それまでに探索した範囲内での最適値 a を記録していく。もしある段階でそれより先の探索をしても、a の値の改善がまったく見込めないということがわかる場合はそれ以上の探索を打ち切って（枝刈りという）、一段前へバックトラックして未探索の枝があれば、それ以下の探索に移る。したがって、計算の過程は木で表現できる。このとき重要なのは、a の見積もりをすることで、その見積もりが的確であると枝刈りが正確にできるが、この見積り自体が一般にそう簡単ではない。この見積りは厳密な値でなく、簡単に計算できる近似値を用いることが多い。たとえば、制約の一部を省いて解の範囲をゆるめた（緩和した）場合の最適値を打切りの目安に使うなどの工夫をする。解の数の削減とともに、見積もりのための計算量も考えてアルゴリズムを設計しなければならない。

分枝限定法は組合せ最適化問題に対する代表的な解法の一つであり、一言で述べれば、要領のよい列挙法である。整数計画問題、ナップザック問題、巡回セールスマン問題、スケジューリング問題など広範囲の問題、とくにＮＰ困難性に代表されるような難しい問題に対する実用的厳密解法の手法とされている。

分枝限定法の第一の基本は、与えられた問題 P_0 をいくつかの部分問題に分解する分枝操作である。分枝操作は生成された部分問題にも次々と適用され、計算経過は、図 2.1 のような探索木に表わされる。部分問題の規模がある程度小さくなると厳密に解くことができるので、最終的にすべての葉節点の部分問題がすでに解かれたという探索木が得られ、そのとき原問題も解かれる。

しかし、可能な分枝操作をすべて実行するのでは単なる列挙法であって。大規模な問題を扱うことはできない。そのため、生成された部分問題 P_i にテストを加え、その結果、P_i の最適解が求まる場合や、あるいは逆に、P_i から原問題の

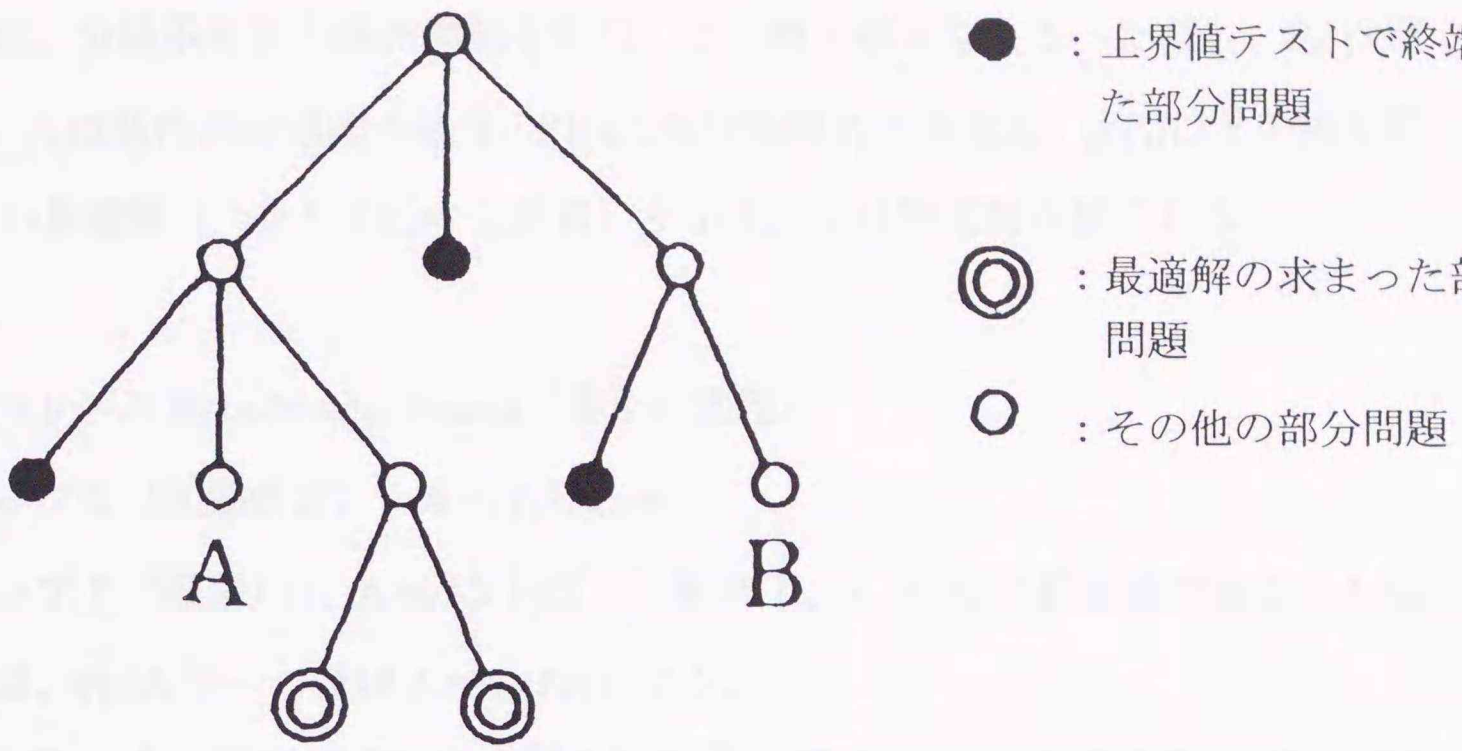


図２．１ 分枝限定法に対する探索木

最適解が得られないことが結論できる場合には、ただちに P_i を終端し以後の考察から除くという操作が加えられる。これを限定操作という。

計算のある時点において、まだ分解も終端もされていない部分問題は活性であるという（図 2.1 の例ではＡとＢの部分問題）。また、それまでに得られている解のうち最良のものを暫定解として保持しておく。こうすれば、計算終了時、暫定解は原問題の最適解を与える。分枝限定法の各反復では、活性部分問題の中から一つの部分問題 P_i を選びテストを加える。テストにはその制約条件をゆるめた緩和問題 $\overline{P_i}$ を解くことが多い。このとき、(a) $\overline{P_i}$ の最適解がの制約条件を満たせば、それは P_i の最適解である、(b) $\overline{P_i}$ の最適値が暫定解の値以上であれば、

P_i から暫定解より良い解を得ることはできない（最小化問題の場合）、(c)$\overline{P_i}$ が実行可能解をもたなければ P_i ももたない、などの性質を限定操作に利用できる。

次に、分枝限定法の緩和問題を利用した一般手順を与える。ただし、P_0 は原問題、A は活性部分問題の集合、$f(P_i)$は部分問題 P_i の最適値、$g(P_i)$はその緩和問題$\overline{P_i}$ の最適値（つまり $f(P_i)$の上界値）を示す。 z は暫定解の値である。

アルゴリズム Branch- and- Bound（最小化問題）

ステップ０（初期設定）：A:={P_0}; z:=∞;

ステップ１（探索）： A:=∅ならば、計算終了。z が P_0 の最適値である。A≠∅ならば、Pi∈A を一つ選び A:=A-{Pi}とする。

(.) ステップ２(限定操作) :Pi の緩和問題$\overline{P_i}$ を解いて、g(Pi)を求める。g(Pi)=f(Pi)であれば、z:=min{z,f(Pi)}とし、ステップ１へ戻る。また、g(Pi)≥z ならば、ステップ１へ戻る。

ステップ３（分枝操作）：分枝操作によって Pi を $P_{i_1}, P_{i_2}, \cdots, P_{i_k}$ に分解し、A:=A∪{ $P_{i_1}, P_{i_2}, \cdots, P_{i_k}$ }としたのち、ステップ１へ戻る。

分枝限定法をアルゴリズムとして具体的に定めるには、

(1) 分枝規則

(2) 探索法

(3) 削除規則（限定操作）

(4) 優越関係

(5) 下界値関数

(6) 上界値

の細部を決定することが必要である。詳細は第４．２章で述べ、本研究問題に対して具体的に検討を加える。

2．2 メタヒューリスティックによる近似解法の構成

組合わせ最適化問題の多くはNP困難な問題のクラスに属し、問題のサイズに対して計算時間が指数関数的に増大する。このため、前章で述べた厳密解法では実用的な時間内で解くことは困難であり、通常、近似的な値を持つ解によって代替する戦略が取られる。このため、多くの近似解法が提案され実用化が計られている。本研究ではその中で、最近多くの成果をあげているパラダイムであるメタヒューリスティックを採用し、その効果について論じる。

メタヒューリスティックは従来の組合わせ最適化問題を解くための種々の戦略を有機的に結合させたり、あるいは反復させたものであり、従来の近似解法を超えたパラダイムとして注目を浴びている。このとき、各手法においていくつかのパラメータを制御することによって、様々な戦略を構成できる。すなわち、メタヒューリスティックとはヒューリスティックにパラメータを追加し、そこで生まれた自由度を用いて問題を巧く解くテクニックであり、それを工夫することによって、組合わせ最適化問題に対する実用的なツールとなり得る。

このように、メタヒューリスティックの実用化にはパラメータの適正化が不可欠であるが、このパラメータの適正値は通常、系統的な数値実験によって得られる。

ここでは、Local search法、Tabu Search法、Simulated Annealing法などのメタヒューリスティックを示す。多くのメタヒューリスティックは自然界にアナロジーを持つことを一つの特色とする（それが必要条件ではない）。 Tabu Search法は人間の記憶過程にアナロジーを持つ。すなわち、最近記憶した情報をもとに、再び、無駄な同一過程を繰返さない戦略からなる。Simulated Annealing法 は物理現象の焼き鈍しのアナロジーから構成される。すなわち、高い温度から低い温度へと十分にゆっくりと冷却するとき、結晶構造が最大化する現象を組合わせ問題の最適化問題に置き換えたものである。また、ここではふれないが、

生物の進化の現象を取入れた Genetic Algorithm 法もこの範疇に属するものである。

２．２．１ 諸定義と問題

ここでは、以後の説明のために必要となる問題の定義、および諸記号について簡単に記述する。

NP 困難な、問題である離散的最適化問題は一般に以下のように与えられる。

$$\begin{cases} Min & f(x) \\ Subject\ to & x \in X \end{cases}$$

ここで、x は n 次元ベクトルからなる解を表わし、X は実行可能解の集合を表わす離散的な性質を持つ集合を意味する。また、関数 $f(x)$は目的関数と呼ぶ。特にこだわらない限り、以上のように最小化問題について考える。ここで２つの典型的な組合わせ最適化問題を示す。

巡回セールスマン問題（Traveling Salesman Problem）

巡回セールスマン問題とは、n 個の頂点（都市）から成るグラフ $G=(V,E)$、枝上の距離関数 d が与えられたとき、すべての点をちょうど１回ずつ経由する巡回路で、枝上の距離の合計を最小にするものを求める問題である。

グラフ２分割問題（Graph Two Partitioning Problem）

無向グラフ $G=(V,E)$があたえられたとき（$n=|V|$は偶数とする）、頂点集合 V の一様分割(L, R)とは $L \cap R = \varnothing$, $L \cup R = V$, $|L| = |R| = n/2$ を満たす頂点の部分集合の対である。グラフ２分割問題とは、 L と R の間にある枝のコストの総和を最小にする一様分割(L, R)を求める問題である。

2．2．2 近傍構造

メタヒューリスティックのアルゴリズムの構造は近傍を基礎として構築され、メタヒューリスティックの効率は近傍の構造に強く依存していることが経験的に知られている。以下に近傍の定義と各問題への代表的な近傍の構成について述べる。

実行可能解の集合 X を与えたとき、近傍 N は以下の写像と定義される。

$$N : X \to 2^{|X|}$$

すなわち、実行可能解の集合から、そのべき集合（部分集合の集合）への写像が近傍である。

実行可能解 $x \in X$ で、$f(x) \leq f(y),\ \forall y \in N(x)$ を満たすものを（近傍 N に対する）局所最適解（locally optimal solution, local opt）と呼ぶ。

巡回セールスマン問題の近傍

巡回セールスマン問題に対する古典的な近傍構造として 2-opt 近傍と 3-opt 近傍があげられる。順列 ρ に対応する巡回路に含まれる枝の集合を $T(\rho)$、$G=(V, E)$ 上の Hamilton 閉路の集合を $\mathcal{H}$ と記す。

定義２．１ 巡回セールスマン問題に対する 2-opt 近傍 N_2 は以下のように定義される。（図２．２(a)参照）

$$N_2 = \{ \rho' \in \mathcal{H} : T(\rho') = T(\rho) - \{e_1,e_2\} \cup \{e_3,e_4\},\ e_1,e_2 \in T(\rho),\ e_3,e_4 \in E - T(\rho) \}$$

定義２．２ 巡回セールスマン問題に対する 3-opt 近傍 N_3 は以下のように定義される。（図２．２(b)参照）

$$N_3 = \{ \rho' \in \mathcal{H} : T(\rho') = T(\rho) - \{e_1,e_2,e_3\} \cup \{e_3,e_4,e_6\},\ e_1,e_2,e_3 \in T(\rho),\ e_3,e_4,e_6 \in E - T(\rho) \}$$

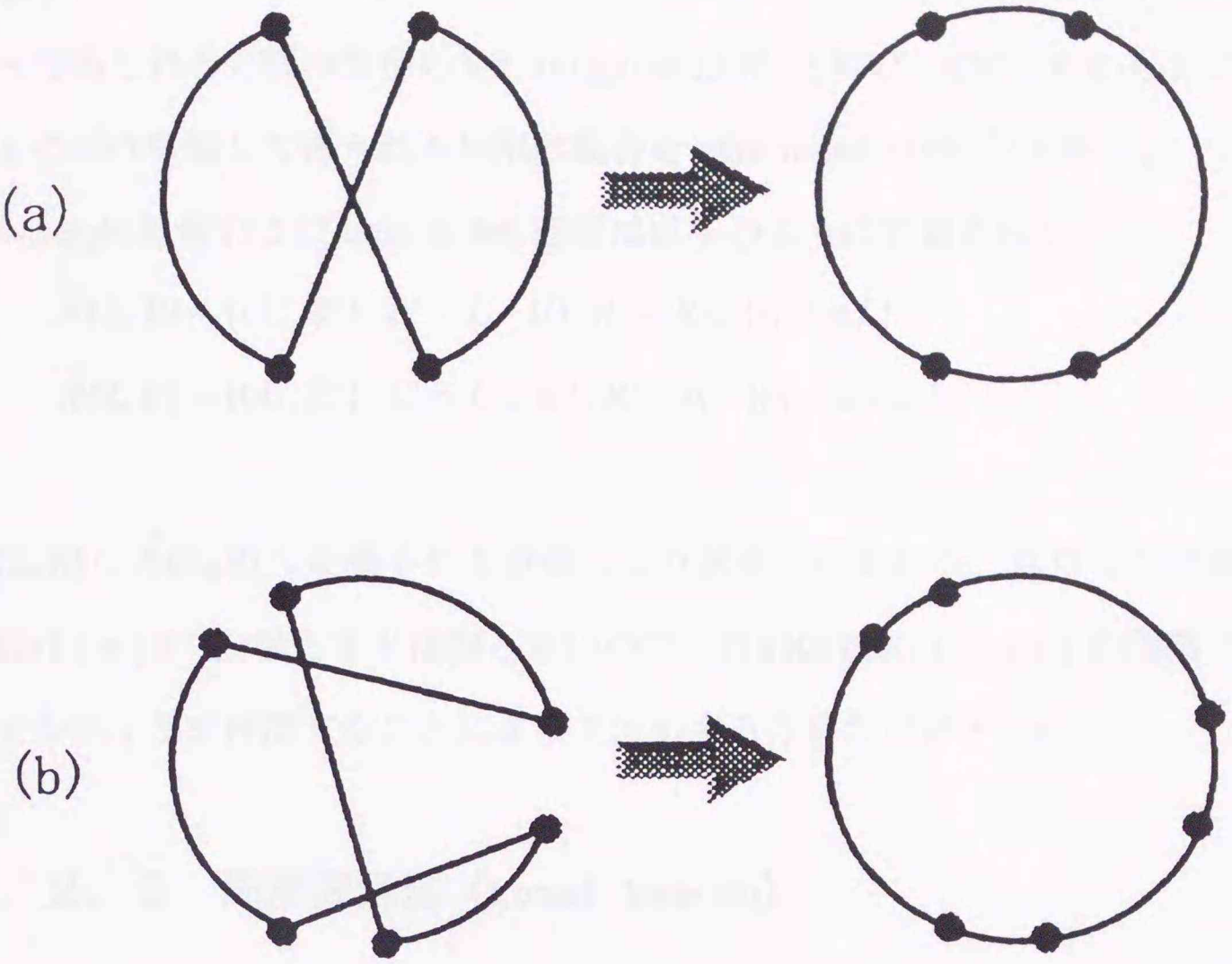


図２．２ 巡回セールスマン問題に対する近傍操作

(a) 2-opt.

(b) 3-opt.

2-opt、3-opt の拡張として Lin と Kernighan の近傍がある。この近傍を用いた局所探索法は Lin-Kernighan opt と呼ばれ、今日でも巡回セールスマン問題に対する最も効率的な近似解法の一つである。

グラフ２分割問題の近傍

グラフ分割問題の代表的な近傍は以下のようなものであり、本研究においても、この考え方をベースとした近傍構造を利用している。

定義２．３　２分割 (L, R)において、ＬからＲに頂点を１つだけ移動することによって得られる分割の集合を left-to-right 近傍 $\overrightarrow{N}$ と呼び、逆に、Ｒから Ｌに頂点を１つだけ移動して得られる分割の集合を right-to-left 近傍 $\overleftarrow{N}$ と呼ぶ。

left-to-right 近傍および right-to-left 近傍は以下のように定義される。

$$\overrightarrow{N}(L,R) = \{(L',R') : L' = L - \{l\}, R' = R \cup \{l\}, l \in L\}$$

$$\overleftarrow{N}(L,R) = \{(L',R') : L' = L \cup \{r\}, R' = R - \{r\}, r \in R\}$$

$\overrightarrow{N}(L,R) \cup \overleftarrow{N}(L,R)$ で定義される近傍により探索が行われる。ただしこの場合、常に$|L| = |R|$ を満たすとは限らないので、目的関数に$|L| - |R|$ の関数であるペナルティ項を付加することによって対処する方策などがとられる。

２．２．３　局所探索法（Local Search）

何らかの方法で得られた可能解 x に対して、その近傍 $N(x)$を定義し、$N(x)$中の可能解の中で目的関数値を改善できるものがあれば、それに置き換えるという方法により解の探索を進め、改善が得られなくなるまで反復するアルゴリズムを局所探索法（ Local Search）という。ここで取り上げるメタヒューリスティックは Local Search を基礎として構築される。その一般的アルゴリズムは図２．３となり、関数 improve(x)を用いることによって記述する。

$$improve(x) = \begin{cases} \forall\, x' \in N(x) & if\ f(x') < f(x) \\ \varnothing & otherwise \end{cases}$$

局所探索法の進行の様子を図２．４に示す。ただし、$x^{(k)}$は k 番目の可能解である。探索が停止すると、そのときの $x^{(k)}$は近傍 $N(x^{(k)})$内に $x^{(k)}$より良い解がないという意味で、局所最適解（local optimal solution）である。

```
procedure local search
x := some initial feasible solution;
while improve(x) ≠ ∅ do
    x := improve(x);
return x;
```

図２．３ 局所探索法（local search）のアルゴリズム

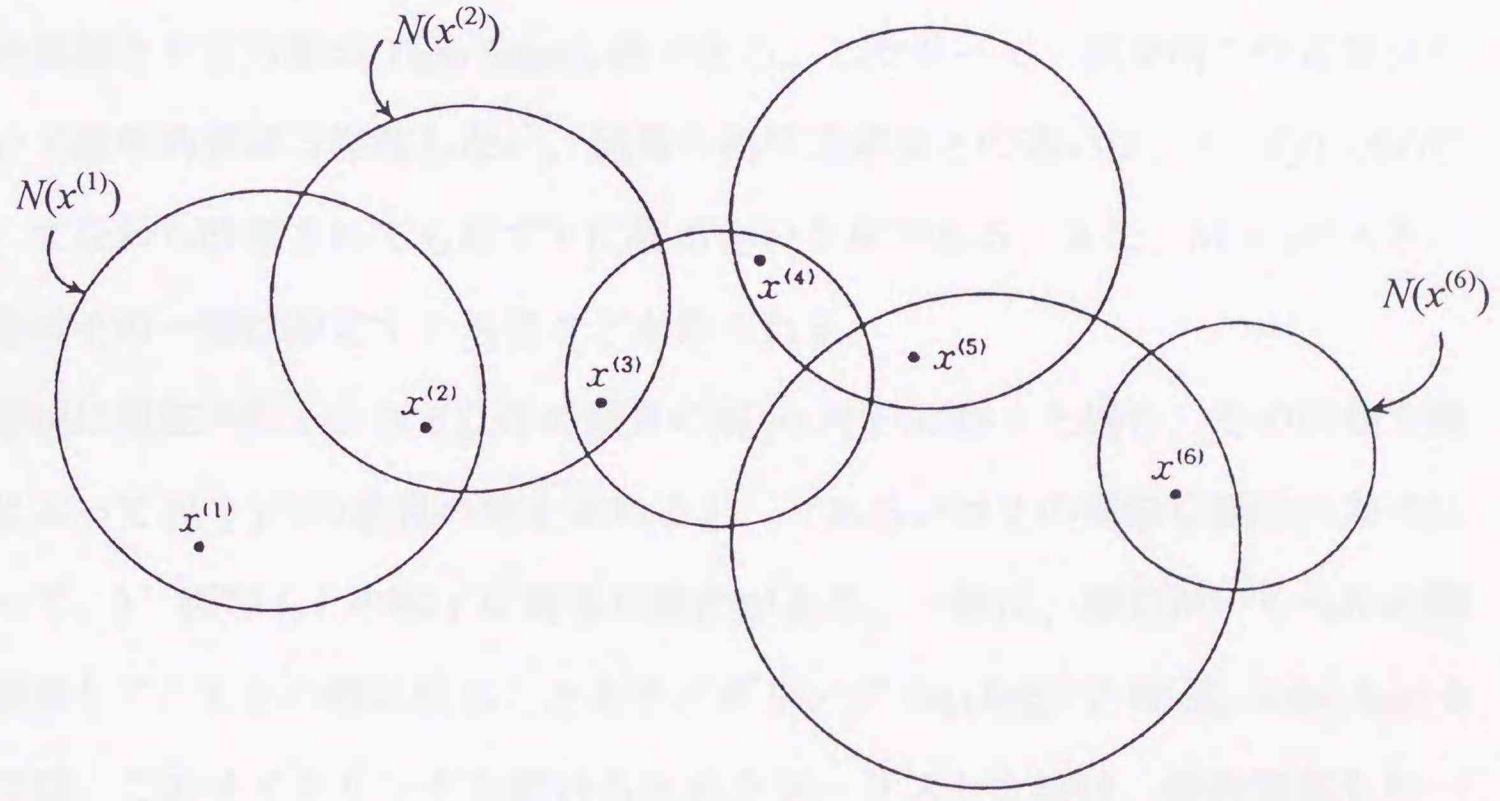


図２．４ 局所探索法の進行

近傍 $N(x)$の定義を定めると、局所探索法の解は初期解によって決定される。一般に、良い初期解から出発すると良い局所最適解が得られる傾向がある。

局所探索法では、未探索の領域にさらに良い解が残っているという危惧を消すことはできないが、この点を改善するため以下のような処置が取られる。

（1）初期解をいろいろ試みる。

（2）探索に確率的動作を導入する。

（3）目的関数値が改善されなくても新しい解に移動する可能性を残す。

2．2．4 Tabu Search 法

Tabu Search 法は F. Glover によって提案された局所探索法の変形である。近傍集合 $N(x)$に対して最近訪れたことがない解集合の中で、x 以外の最良の解 y を次の候補とする方法が Tabu Search 法である。したがって、初期解の決定部分を除いて確率的要素は存在しない。通常の局所探索法との違いは、$\Delta = f(y) - f(x)$が正、すなわち改悪されても必ず y に移るという点である。また、$N(x)$が大きい場合はその一部に限定する処置などが取られる。

単純に現在の解 x から x 以外の最良の解 $y \in N(x)$に移った場合、その同様な操作によって $N(y)$内の最良の解を求めると、（あるいはその同様な操作の系列によって、） 再びもとの解 x に戻る可能性がある。一般に、探索がいくつかの解を経由して、もとの解に戻ることをサイクリング（cycling）と呼ぶ。Tabu Search 法では、このサイクリングを避けるためタブーリストを設け、最近探索した一部の領域への進行を禁止する。すなわち、$N(x)$内の探索はタブーリストに抵触しないものに限定して実行される。この考え方は Glover とは独立に P. Hansen によって Steepest Ascent Mildes Descent 法という名前で同時期に提案されている。

Tabu Search 法では、近傍から禁断リスト tabu_list を除いた中で最も良い解へ移動する。移動を行うための関数 $move(x)$を以下のように定義する。

$$move(x)=\begin{cases} x' & if\ f(x')\le f(y)\ for\ all\ y\in\{N(x)-tabu\_list\} \\ x & if\ N(x)-tabu\_list=\varnothing \end{cases}$$

この関数を用いることによってTabu Search法の概要は図２．５のように書くことができる。なお、tabulengthはタブーリストtabu_listの長さである。

ここで、タブーリストの要素として解そのものを保持しておくことが最も単純な考えであるが、通常、この方法では効果的な結果は得られない。解の移動(x,

```
procedure Tabu Search
begin
    t := 0;
    x0 := some initial solution;
    tabu_list := ∅;
    tabulength := a positive integer;
    while stopcriterion ≠ yes do begin
        xt+1 := move(xt);
        tabu_list := tabu_list ∪ {xt} - {xt-tabulength};
        t := t + 1;
    end
    return x;
end;
```

図２．５ Tabu Search法のアルゴリズム

x')に関係する何かを属性と呼び、それを保持することによって、逆向きの変化方向を禁止する。すなわち、解の移動に関係した、頂点、あるいは辺が属性に対応し、この頂点、あるいは辺が再び、解の移動に関係しないような方式をとることが有効とされている。

また、タブーリストの情報を最近の tabulength 個に限定しているのは、これらの解空の探索が無駄に終わった場合、再びもとの探索解に戻ってそこから探索をやり直す（タブーリストが変化しているため、前回とは異なる探索となる）という可能性を残すためである。

なお、タブーリストも必ずしも絶対的なものではなく、目的関数値の改善が大きければ、タブーリストの条件を無視するという Aspiration Criteria という戦略も考えられている。また、最近の tabulength 個のみ制約の条件とする短期的な考え方に対して、長期的な巡回を防ぐため、頻繁に移動を繰返す解に対して移動しにくくする長期メモリーと呼ばれる制約の考え方もある。

Tabu Search 法の終了規則としては、暫定解の改良がある数以下の反復で生じなかった場合に停止する方法などが考えられる。探索結果の評価を行い、不十分であると判断されると、同様の試行を続ける。このとき、

(a) その前の試行の探索をより詳細にする。

(b) その前の試行で探索されなかった領域を探索する、などの判断によって、次回の初期解および探索パラメータが適応的に設定される。

Tabu Search 法は非常に柔軟性の高い探索の枠組みであり、過去の探索過程のデータをタブーリストという形で記憶しておき、それを十分利用することで探索を実現するものである。各種スケジューリング問題、グラフ彩色問題、充足可能性問題など、いろいろな問題への良好な適用結果が報告されている。

2．2．5　Simulated Annealing 法

Simulated Annealing 法は最適値問題の確率的解法の一つで、Kirkpatrick らの論文[8]によって組合わせ最適化問題に応用され広く知られるようになった。アニーリングとは焼き鈍しという意味である。固体物理学における物質の結晶化の過程を模倣してるのでこの名前がつけられた。Metropolis[10]らによって編み出された熱平衡状態のシミュレーションを行う計算手続きがもとになっている。

Simulated Annealing 法は局所探索をランダムに行い。さらに解の改良が見られない場合でも新しい解に移る確立を残すことで局所最適解に捕捉されてしまうことを防ぐ特徴を持っている。

現在の解 x に対し、$y \in N(x)$をランダムに選び、

$$\Delta = f(y) - f(x)$$

と置く。最小値問題を前提としているので、$\Delta < 0$ ならば改善、$\Delta > 0$ ならば改悪である。このとき、$\Delta \leq 0$ ならば直ちに新しい解 y に移るが、$\Delta > 0$ であっても、確立

$$e^{-\Delta T} \tag{2.8}$$

で y に移るのである。パラメータ T (>0) は温度と呼ばれている。T は適当な初期値から始め、その温度での試行が十分になされたと判断すると、小さな値

$T := r\,T$　（ r は $0 < r < 1$ を満たす定数 ）

に直す。その結果、移動の確立（２．８）は小さくなる。Simulated Annealing 法による探索は T が十分小さくなったと判断される（凍結温度）と打ち切られ、この探索過程で得られた最良の可能解が出力される。図２．６にそのアルゴリズムの構成を示す。

上記ののパラメータの T_0(初期温度)、stoptmp（凍結温度）、R(初期反復数)、phi（温度縮小率）、tau（反復増加率）はあらかじめ設定しておく。これらのパラメータがアルゴリズムの挙動に大きな影響力をもつ。類似の問題での結果を参考にして、経験的に定めることが多い。

```
Procedure simulated annealing;
begin
    tmp := T0(initial temperature);
    r := R;
    x := initial solution; x*:=x;
while T > stoptmp do begin
    for i:= 1 to r do begin
        y = a solution randomly selected in N(x)
        Δ = f(y) - f(x)
        if   Δ   <= 0   then
            x = y;
        else
            if   exp(-Δ / tmp) >= random[0,1) then
                x = y;
        if f(x) < f(x*) then x* := x;
    end;
        tmp := tmp × phi;
        r := r × tau;
    end;
    best solution := x*;
end;
```

図２．６　Simulated Annealing 法のアルゴリズム

また、Simulated Annealing 法に対しては多くの数学的アプローチが試みられている。Simulated Annealing 法の数学的モデルとしてよく知られている論文は German and German[3], Gidas[4], Lundy and Meees[9], Mittra et al.[11], Hajek[6] がある。これらの論文の多くは Simulated Annealing 法の分析のためにマルコフモデルを用い、マルコフ連鎖の平衡分布を Simulated Annealing 法の平衡状態と関連づけ考察している。これらの分析の大きな主張点は Simulated Annealing 法が漸近的に大域的最適解に収束することを証明している。ただし、Sasaki and Hajek[13]等は、もし Simulated Annealing 法の過程を厳密に再現するならば、簡単な問題でさえ収束の時間は指数的オーダーとなることを述べている。

大域的最適解への収束の証明を以下に簡単に行う。Simulated Annealing 法によって生成された数列$\{x_k\}$はマルコフ連鎖 M を導く。近傍 $N(x)$の中からランダムにyを選択する確率を$R(x,y)$とすると、マルコフ連鎖Mの遷移確率は次のようになる。

$$P_k(x,y)=\begin{cases}0 & if\ y\notin N(x)\ and\ y\neq x\\ R(x,y) & if\ y\in N(x)\ and\ f(y)\le f(x)\\ R(x,y)\exp\{-(f(y)-f(x))/T_k\} & if\ y\in N(x)\ and\ f(y)>f(x)\\ 1-\sum_{x\neq x'}P_k(x,x') & if\ y=x\end{cases}$$

ここでは、強い仮定のもとでの解析のみを示す。まず、以下の仮定のもとで考察を考察を進める。

a) 温度の数列$\{T_k\}$が一定の値をとる。

b) マルコフ連鎖Mは既約である。すなわち任意の２つの解x,yを結ぶパスが０でない確率で存在する。

c) マルコフ連鎖Mは可逆である。すなわち、ある段階kでの解の集合をF_k、またその上で解xが実現する確率分布を$\pi_k(x)$で表わすと、

$\pi_k(x)R(x,y)=\pi_k(y)R(y,x)$

が成立する。

上記の仮定のもとではマルコフ連鎖 M は定常であり、既約かつ非周期的であるので、F^*を大域的集合とすると、Markov ergodic 収束定理[7]により

$$\lim_{k\to\infty} \Pr ob\{x_k \in F^*\} = \frac{\sum_{x\in F^*} \pi_\infty(x) \exp\{-f(x)/T\}}{\sum_{x\in F} \pi_\infty(x) \exp\{-f(x)/T\}}$$

が成り立つ。よって、Tをエルゴード性を壊さないように、非常にゆっくりと0に収束させたときには

$$\lim_{T\downarrow 0} \lim_{k\to\infty} \Pr ob\{x_k \in F^*\} = 1$$

となり、漸近的収束が言える。

Simulated Annealing 法の応用例としては、巡回セールスマン問題をはじめとする数多くのＮＰ困難な問題に適用されている。また、工学分野ではＶＬＳＩの設計における配置問題、配線問題に有効に使われている。その他、画像処理の問題などにも用いられている。

２．３ まとめ

本章では、基本となる組合せ最適化問題に対する各種の解法について概要を説明し、本論文で必要となる基本知識を総括している。本論文中で使用する厳密解法では、数理的に動的計画法と分枝限定法の計算手順を示した。また、近似解法で、従来の近似解法を超えたパラダイムとして注目を浴びているメタヒューリスティックを使用し、その基本的構造である近傍構造について容易な説明を加え、さらに、その近傍構造を土台とする Tabu Search 法と Simulated Annealing 法の２つの手法について記述した。Tabu Search 法は人間の記憶過程にアナロジーを持つパラダイムであり、最近記憶した情報をもとに、再び、無駄な同一過程を繰返さない戦略からなる。Simulated Annealing 法 は物理現象の焼き鈍しの

アナロジーから構成される。すなわち、高い温度から低い温度へと十分にゆっくりと冷却するとき、結晶構造が最大化する現象を組合わせ問題の最適化問題に置き換えたものである。また、多くの良好な結果がこの２つのアプローチから得られている。

参考文献

[1] Aarts,E. and Korst,J. : Simulated Annealing and Boltzmann Machines, John Wiley & Sons(1989).

[2] Gass, S.I. and Harris, C.M. : Encyclopedia of Operations Research and Management Science", Kluwer Academic Publishers, (1996).

[3] Geman, S., and Geman, D. : Stochastic Relaxation, Gibbs Distribution, and Baysian Restoration of Images, IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-6, pp.721-741(1984).

[4] Gidas, B. : Non-stationary Markov Chains and Convergence of the Annealing Algorithm, Jl. Statistical Physics, 39, pp.73-131(1985).

[5] Glover, F. : A User's Guide to Tabu Search, Annals of Operations Research,41,pp.3-28(1993).

[6] Hajek, B. : Cooling Schedules for Optimal Annealing, Math. Operations Research, 13, pp.311-321(1988).

[7] Kemeny, J. and Snell, J.L. : Finite State Markov Chains, D.Van Nostrand Company(1960).

[8] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. : Optimization by Simulated Annealing, Science, 220, pp.671-680(1983).

[9] Lundy, M. and Mess, A. : Convergence of an Annealing Algorithm, Math. Programming, 34, pp.111-124(1986).

[10] Metropolis, W., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E. : Equation of State Calculation by Fast Computing Machines, Journal of Chemical Physics, 21, pp1087-1092(1953).

[11] Mitra, D., Romeo, F. amd Sangiovanni-Vincentelli, A. : Convergence and Finit-time

Behavior of Simulated Annealing, Advances Applied Provability, 18, pp.747-771(1986).

[12]Reeves,C.R : Modern Heuristic Techniques for Combinatorial Problems, BlackWell(1993).

[13]Sasaki , G.H. and Hajek, B. : The Time complexity of Maximum Matching by Simulated Annealing, Jl. Assoc. Computing Machinery, 35, pp.387-403(1988).

[14]茨木：組合せ最適化, 産業図書(1983)

[15]茨木：離散最適化法とアルゴリズム(岩波講座 応用数学)、岩波書店(1993).

[16]伊理、今野、刃根：最適化ハンドブック, 朝倉書店(1995).

[17]大矢、今井、小嶋、中村、廣田：数理情報科学辞典, 朝倉書店(1995).

[18]島内、有澤、野下、浜田、伏見：アルゴリズム辞典, 共立出版(1994).

[19]北川 : OR 辞典, 日科技連(1975).

[20]久保：モダンヒューリスティックスの新展開－Gnetic Algorithm,Simulated Annealing,Tabu Search,Neural Net 法は本当に有効か－,日本 OR 学会第 30 回シンポジウム(1994).

[21]久保：Local Search から Simulated Annealing,Tabu Search へ ,モダンヒューリスティックス（平成 6 年度第 2 回 OR セミナー）,pp.1-32(1994).

[22]久保：メタヒューリスティックス, 離散構造とアルゴリズムIV, 近代科学社, pp.171-230(1995).

[23]長尾、石田、稲垣、田中、辻井、所、中田、米澤：情報科学辞典, 岩波書店(1990).

[24]西川、三宮、茨木：最適化, 岩波書店(1982).

第３章　一列化グラフの最適系列分割問題

本章では、提案する無閉路有向グラフの最適系列分割問題に先立ち、本研究の起点であり、また、無閉路有向グラフの最適系列分割問題に関する解法の重要な構成要素となる一列化グラフの最適系列分割問題について紹介を行う。

重みをもつ n 個のノードとコストをもつエッジからなるグラフ G を考える。ここで、G を各部分集合に分割し、その分割された異なる部分集合間のエッジコストの総和が最小とする問題を分割問題と呼ぶ。これは、またクラスタリングとも呼ばれる。この分割問題の中で、最も難しい問題として、各部分集合の大きさを限定する条件のもとでコストを最小化する問題が挙げられる。この問題の応用としては、回路のボード上において組み込まれる要素が限定された場合の回路設計問題などに広く利用されている。この問題に対しては n が非常に小さい場合、分枝限定法などを用い正確に解くことができる。また、大規模な問題に対してはヒューリスティックな戦略を用いた近似解法が開発されている。

Kernighan は、この各部分集合の大きさを限定した分割問題の特別なケースについて問題を提案し、グラフのエッジ数に線形な計算時間で計算可能な厳密解法を提示した。この特別なケースの問題はグラフ G のノードが連続的な番号 1, 2, ... , n を保持したグラフに対して、各頂点に与えられた重みの総和がブロックサイズ $B(>0)$以下であり、かつ、部分集合の頂点番号が連続的に保持される条件のもとで、カットされる辺のコストの総和が最小となるよう分割する問題である。この応用事例としては、G のノードをコンピュータプログラムの各命令として表し、エッジをその命令の順序関係とし、エッジのコストを命令間の推移

の相対頻度を表すものとする。そのとき、Kernighan の問題はプログラムをサイズ B のページへ、ページ間の推移頻度を最小化するよう分割配置する問題（ページング）となる。

本章ではこの Kernighan の問題について説明し、その解法について紹介する。さらに、この問題の性質を明らかにし、Kernighan が述べるところの計算時間について説明を加える。

3．1 諸定義

連続的に番号づけられた n 個の頂点からなる集合 $V = \{1, 2, \ldots, n\}$、辺集合 E からなる無向グラフを一列化グラフと呼び、 $G(V, E)$ で表わす。各無向辺を頂点対$\{i, j\}$で表すとき、辺$\{i, j\}$には非負のコスト $c(i, j)$を付与する。さらに各頂点は $W = \{w(1), w(2), \ldots, w(n)\}$の重みをもつ。ただし、$0 < w(i) \leq B$, $1 \leq i \leq n$ である。ここで B はブロックサイズと呼ばれる数である。本論文ではすべての重み $w(i)$ と B は正の整数とする。便宜上、人為的に番号 $n+1$ の頂点を加え、$c(n, n+1) = 0$ とする。

次の２つの制約（Ⅰ）、（Ⅱ）のもとで G を k 個の部分グラフ $G_i(V_i, E_i)$, $i = 1, 2, \ldots, k$ に分割することを一列化グラフの系列分割と呼ぶ。ただし k はあらかじめ与えていない数である。

（Ⅰ）G_iの頂点の重みの総和　≦　B

（Ⅱ）任意のの各頂点番号は連続的な番号をもつ。

ここで一列化グラフの最適系列分割問題とはこの系列分割の中で、切断される辺のコスト、すなわち異なる部分グラフ中に両端点をもつ辺のコストの総和を最小にするように分割する問題である。部分グラフ G_i の頂点集合 $V_i = [b_i, b_{i+1}) = \{b_i, b_i + 1, \ldots, b_{i+1} - 1\}$ はブロックと呼び、ブロックの数 k を分割のサイズと呼ぶ。$k^* = \lceil |G| / P \rceil$ は分割数の下界値となり、 $k^* \leq k$ となる。（$\lceil x \rceil$ は x 以下の最大

の整数である。）ここで、 $k^* / k\ (\geq 1)$を膨張率と呼ぶ。V_iでの一番小さな頂点番号b_iをブレーク・ポイントと呼ぶ。ブレーク・ポイントb_iは頂点b_i-1と頂点b_iの間に切断が存在することを意味する。集合$\{ b_1, b_2, \ldots, b_k \}$は一意的に分割$\{ G_1, G_2, \ldots, G_k \}$を表現する。有向グラフに対しては、（ｉ，ｊ）と（ｊ，ｉ）の辺はｉ＜ｊである単一辺｛ｉ，ｊ｝でおきかえると同時に、全ての有向辺を無向辺で置き換える。その時、コストは $c'(i,j) = c(i,j) + c(j,i)$ とする。

図３．１は連続的な頂点集合Ｖ＝｛１，２，・・・　，１０｝からなり、上部の数字のコストをもつ辺からなるグラフである。また各頂点の重みは１とする。図３．１上の破線で切断された分割がブロックサイズＰ＝４での最適解を与える。このとき、各ブロックは頂点集合$Ｖ_1$＝｛１，２，３，４｝，$Ｖ_2$＝｛５，６｝，$Ｖ_3$＝｛７，８，９，１０｝からなり、ブレーク・ポイントの集合は｛１，５，７，１１｝となる。また最小となるコストの総和は８３である。１１は便宜上、加えた頂点である。

分割を表現するブレーク・ポイントの集合｛$ｂ_1$，$ｂ_2$，・・・，$ｂ_k$｝，$ｂ_1$

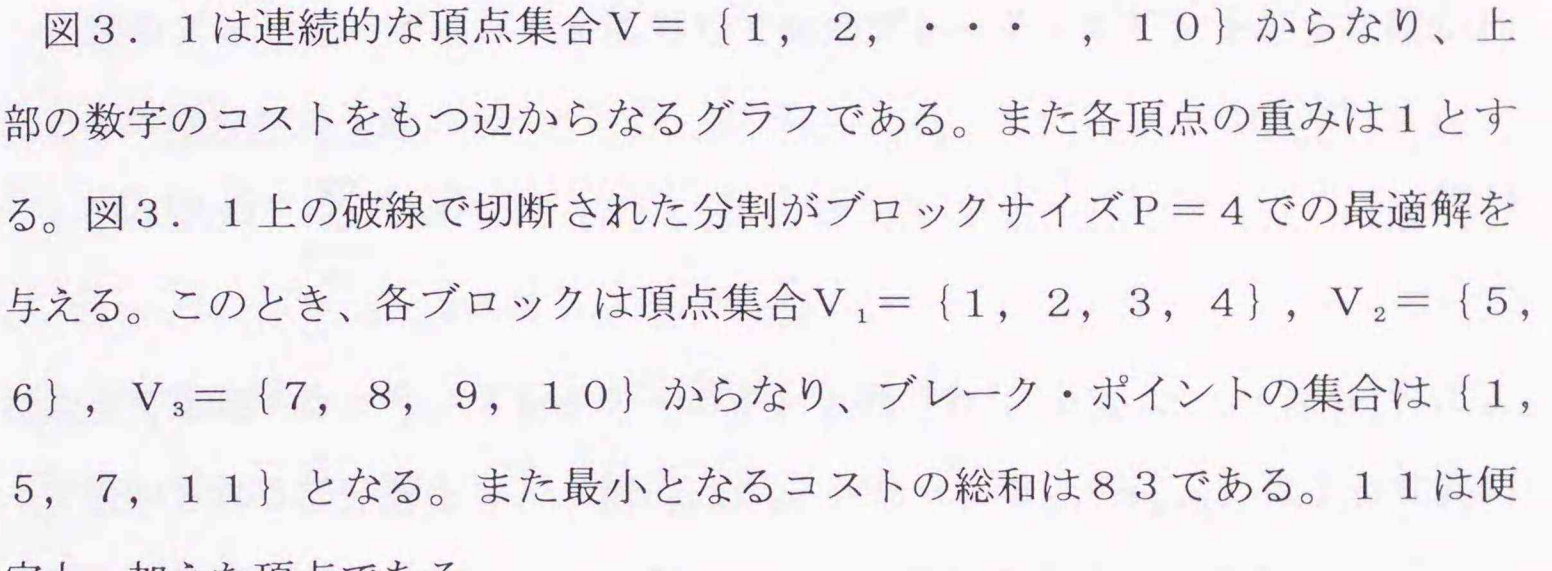


図３．１　一列化グラフの系列分割

＝1、b_k＝n＋1を与えられた問題の完成解、また各ブロックが系列分割の制約条件（Ⅰ）、（Ⅱ）をみたす完成解を可能解と言う。第m番目以降のブレーク・ポイントの値が不定である列｛b_1，b_2，・・・，b_m，*，*，．．．，*｝を部分解、その最後の確定要素であるb_mをこの部分解の最終ブレーク・ポイントと呼ぶ。完成解の目的関数値および部分解のコストはそれらのブレーク・ポイントにより切断されたグラフGの辺のコストの総和である。

任意のブレーク・ポイントxに対して次のブレーク・ポイントがyに置かれたときの増分コストを

$$C_f(y,x) = \sum_{\substack{y \le i < x \\ j \ge x}} c(i,j) \tag{3.1}$$

によって定義する。今、ブレイク・ポイント列｛b_1，b_2，・・・，b_m｝によって表わされる部分解をπ、｛b_1，b_2，・・・，b_m、b_{m+1}｝によって表わされる部分解をπ´とすれば、πとπ´に対して次の関係が成立する。

$$\mathrm{Cost}(\pi') = \mathrm{Cost}(\pi) + C_f(b_m, b_{m+1}) \tag{3.2}$$

3．2 Kernighanによるアプローチ

3．2．1 動的計画法によるアルゴリズムの構成

以上の問題に対してKernighanは動的計画法（ＤＰ）を適用し効率的な優れたアルゴリズムを構成した。以下にそのアルゴリズムの概略を記す。

最終ブレイクポイントがxである最良な部分解のコストをＴ（ｘ）とすると、Ｔ（ｘ）は関係（３．２）から最適性の原理が導かれことにより以下の関数方程式により漸化的に求められる。

$$T(x) = \min_y \{T(y) + C_f(y, x)\} \tag{3.3}$$

yはxの一つ前のブレイクポイントであり、”yからｘ－１までの距離≦B”の

範囲で選ぶ。$C_f(y, x)$は一つ前のブレイクポイントｙに対して、次のブレイクポイントがｘとなる時のコストの増分であり、（３．２）式で示されている。

この漸化式を用い順次求まる最適部分解のコストＴ（ｘ）と一つ前のブレイクポイントをＬ（ｘ）に保存しておく。最終解に到達したとき保存してあるブレイクポイントから分割が求まる。以下にKernighanが論文中で与えたアルゴリズムKDP(Kernighan's Dynamic Programming Algorithm)を示す。

１：Ｓｅｔ　Ｔ（１）＝０；　Ｌ（１）＝０；

２：ｆｏｒ　ｘ＝２　ｔｏ　ｎ＋１　do

　　Ｔ（ｘ）＝ｍｉｎ$_y$［Ｔ（ｙ）＋C_f（ｙ、ｘ）］

但し、最小化はd(y, x-1)≦Bをみたす全てのｙの上でとる。（d(i, j)は頂点ｉからｊまでの距離を表わし、重みの総和$w(i)+w(i+1)+\ldots+w(j)$である。）もし一つ以上のｙがこの条件をみたすならば、その最小のものを選ぶ。そして

　Ｌ（ｘ）＝ｙ

とおく。

３：Ｓｅｔ　Total_Cost＝Ｔ（ｎ＋１）；

　　Ｓｅｔ　z_0＝ｎ＋１；

　　Ｓｅｔ　ｋ＝０；

４：ｗｈｉｌｅ（ｚ＞１）｛

　　z_{k+1}＝Ｌ（z_k）；

　　ｋ＝ｋ＋１；

　｝

５：ブレイクポイントは大きいほうから

z_1, z_2, ・・・, z_kである。

Kernighan はこのアルゴリズムの辺コストへの参照回数が辺の個数に線形比例することを述べている。しかしこれはステップ２における C_f（y、x）の計算過程の一部であり、アルゴリズム全体の計算量については考慮していない。

３．２．２ 解法の特性

ＫＤＰアルゴリズムの最適性の証明、およびそのアルゴリズムの特性と計算時間についての定理を導く。

定理３．１ 手続きＫＤＰは一列化グラフの最適系列分割を発見する。

証明： 手続きＫＤＰによって定義された２つのブレイク・ポイント z_0 と z_1 に対して $T(z_0)=T(z_1)+C_f(z_1, z_0)$となる。そのとき、 $T(z_0)$は、$T(z_1)+C_f(z_1, z_0)$が $d(z_1、z_0\text{-}1)\leq B$ であるノード上に関して最小コストであるときに限り、最小コストである。順番に、 $T(z_1)$は $T(z_2)+C_f(z_2, z_1)$が $d(z_2、z_1\text{-}1)\leq B$ であるノード上の最小コストであるときに限り、最小コストとなる。そのようにして、 $T(z_k)= T(1)$となるまで行ったならば、 $T(1)$は０となり明らかに最小値となるので、 $T(z_0)$までの系列により、この手続きは最小コスト分割を導出する。

□

定理３．２ KDP によって生成された分割はすべての一列化グラフの最適系列分割の集合の中で膨張率が最小となる。

証明： KDP によって求められた最適系列分割に対するブレイク・ポイントの集合を $BP=\{b_1,\ldots,b_k\}$とする。 $BP'=\{b'_1,\ldots,b'_m\}$を $m<k$ である同一のコスト、あるいはそれ以下のコストを伴う分割を表わすものとする。構造上、 b_k は $d(y, n)\leq B$ で

あるノードyの範囲内の最小コストをもつブレイク・ポイントの位置となる。もし、そのとき、最小コストを示す分割が一つしか存在しないならば、b'_mは必然的にb_kと等しい。しかし、もし、同一コストをともなう複数の分割が存在するならば、b'_mはb_kより大きな番号とならなければならない。なぜならば、b_kは最小コストの中で、一番小さな番号であるノードとして選択されるためである。同様に、b_{k-1}は$d(y, b_{k-1}) \leq B$であるノードyの範囲上で最小コストであり、かつ番号が最小であるブレイク・ポイントの位置となる。以上から、$m \geq k$となり、矛盾する。

□

定理３．３　KDPによって求まる系列分割の膨張率は２未満となる。

証明： ブレイク・ポイントができるだけ接近して置かれたとき、最悪の膨張率となる。しかし、このとき３つのブレイク・ポイント列に対する距離は少なくともBよりも大きな値となる。すなわち、ある$e>0$に対してのB+eの値となることを意味する。なぜなら、もし、その距離がB以下であるなら、２つのブロックは、コストを増加することなしに、１つのブロックによってまとめることが可能であるためである。これは、ＫＤＰによって実現されることを考慮すると。最悪のケースの膨張率は$2B/(B+e)$、すなわち２未満となる。

□

定理３．４　ＫＤＰの計算時間はグラフのエッジ数に線形に比例する。

証明：　Ｔ（ｘ）＝ｍｉｎ$_y$｛Ｔ（ｙ）＋C_f（y, x）｝の計算において、任意のxに対しての最小化の処理数は$d(y, x-1) \leq B$として限定されている。$Cf(y, x)$は$Cf(y, x-1)$を用い、以下の式で計算される。

$$C_f(y, x)=C_f(y, x-1)+(c(x-1, x) + c(x-1,x+1) + \ldots + c(x-1, n))$$

$$- (c(y, x-1) + c(y-1,x-1) + ... + c(x-2, x-1))$$

すべてのＴを計算する過程で、おのおののエッジは２度参照するのみで計算可能であり、ＫＤＰの計算時間はエッジ数に線形に依存する。

□

３．３ まとめ

本章ではKernighanが示すところの一列化グラフの最適系列分割問題について紹介し、提案された算法について記述し、さらにその問題の特性について説明を加えている。一列化グラフの最適系列分割問題とは、連続的な頂点番号を保持した一列化グラフに対し、その系列性を保持するとともに容量制約のもと、カットエッジのコストの総和を最小化する問題である。また、Kernighanはこの問題に対して、動的計画法を適用し算法を構成している。また、その主張点として計算時間がエッジ数に線形となることを報告している。本章ではこれらのKernighanの研究について概括し、本研究への起点とするものである。

参考文献

[1] Kernighan,B.W. : Optimal Sequential Partitions of Graphs, J.ACM, Vol.18, No.1, pp.34-40(1971).

第４章　無閉路有向グラフの最適系列分割問題

グラフの頂点番号を１からｎまで連続的な数とする一列化グラフに対して、各ブロックの頂点番号の連続性を保持する一列化グラフの系列分割問題について前章で論じた。

これに対して、多くのシステムの構造はより一般的な無閉路有向グラフで構成されており、その上での系列分割について検討する必要がある。たとえば、無閉路有向グラフの構造を利用するスケジューリング問題、ラインバランシング問題、および並列処理等の問題が考えられる。

本論文ではこの無閉路有向グラフの場合に系列分割問題を拡張し、より応用性の広い問題を扱えるようにする。一列化グラフの系列分割問題では分割を容易に表現することができ、効率的な算法を組むことができた。しかし、無閉路有向グラフの系列分割問題の場合、問題を表現し、効果的な解法を構築するために、いくつかの点において検討が必要となる。

そこで、本章では一列化グラフの系列分割の本質と考えられる"各ブロック内の頂点番号は連続的な番号付けを保持する"、および"ブロックの前後関係が決定できる"という特徴的な性質を取り出し、これを無閉路有向グラフの場合に一般化し、無閉路有向グラフの系列分割を定義する。また、その性質を検討し、切断の鎖によって無閉路有向グラフの系列分割を一意に表現できることを示す。さらに、切断を簡潔に表す切断決定子を導入することによって最適系列分割問題を定義する。また、一列化グラフの系列分割問題と無閉路有向グラフの系列分割問題との関係について議論する。

4．1　無閉路有向グラフの系列分割

4．1．1　無閉路有向グラフの系列分割の構成

単一の入口と出口を持つ無閉路有向グラフ$D(V,E)$が与えられたとき、Dの有向辺が定めるV上の関係の反射的かつ推移的な閉包をとって得られる順序関係を$\preceq$とする。順序関係$\preceq$はDが無閉路であることから反対称性をみたし、半順序関係となる。このようにして、Dから得られる半順序集合を$(V,\preceq)$で表す。また、$v\preceq u$かつ$v\neq u$を$v\prec u$で表す。

定義４．１　空でない部分集合$A\subset V$から誘導された[5]Dの部分グラフを$\mathcal{D}(A)$で表す。$\mathcal{D}(A)$の任意の２点を結ぶD内の有向路がすべて$\mathcal{D}(A)$の有向路となるとき、$\mathcal{D}(A)$は系列を保持するDの部分グラフであるという。

補題４．１　$\mathcal{D}(A)$が系列を保持するDの部分グラフであるとき、$\mathcal{D}(A)$から導かれる半順序関係$\preceq_A$はAに制限されたDの半順序関係$\preceq$に等しく、$(A,\preceq)\equiv(A,\preceq_A)$となる。

定義４．２　Vの部分集合Aが、$A^c\times A$から選んだ２元対(x,y)に関して、xとyが$\preceq$において比較可能ならば常に$x\prec y$が成立するとき、$\mu=(A^c,A)$をAによって定まるVの切断という。そして、Aをこの切断の上組、A^cを下組という。

定義４．３　２つの切断(A^c,A)，(B^c,B)に対して$A\supseteq B$が成立するとき、(A^c,A)は(B^c,B)の前にあるといい、$(A^c,A)\prec=(B^c,B)$で表す。また$A\supset B$のとき、真に前にあるといい$(A^c,A)\prec\neq(B^c,B)$で示す。

定義４．４　Vの互いに素な部分集合XとYがそれぞれVのある切断の下組と上組に含まれるならば、XとYは切断により分離されるといい、$X|Y$で表す。XとYについて$X|Y$または$Y|X$が成り立つとき、XとYは分離可能であるという。

$X|Y$は定性的には“XがYより前にある”ことを、またXとYを結ぶ辺が存在するときには“それらの辺はすべて同じ向きをもつ”ことを意味している。ここで、無閉路有向グラフの系列分割を次の様に定義する。

定義４．５　$D(V,E)$の頂点集合Vの分割$\{V_1,V_2,\ldots,V_k\}$がその任意の成分集合V_i,V_jに対し、$i<j$のとき$V_i|V_j$が成立し、かつ$V_1|V_2|\ \ldots\ |V_k$を満たすように並べることができるとき、この分割を$D(V,E)$の系列分割という。

このとき、各分割成分V_iは次の性質を持つ。

性質４．１

(1)各Ｖ$_i$は系列を保持するＶの部分集合である。

(2)任意に選ばれた２つの部分集合Ｖ$_i$とＶ$_j$は分離可能である。

(3)分割を形成している各部分集合の間には$V_{i_1}|V_{i_2},V_{i_2}|V_{i_3},\cdots,V_{i_p}|V_{i_1}$　（１≦ｐ≦ｋ）という分離のサイクルは存在しない。

性質４．１の(1)はKernighanの連続番号の保持と相応し、(2)と(3)は分割を形成している各部分集合の前後関係が決定できることを保障する。また、これらの条件は独立ではなく、(2)と(3)を満足する分割は(1)を満たす。

図４．１は無閉路有向グラフの系列分割の一例であり、切断を破線で表し、その右側が上組、左側が下組となることを示す。系列分割$V_1|V_2|\ldots|V_k$が与えられたとき、各成分集合V_iに上組$A_i=V_i\cup V_{i+1}\cup\cdots\cup V_k$、下組$A_i^c=V_1\cup V_2\cup\cdots\cup V_{i-1}$からなる切断$\mu_i=(A_i^c,A_i)$を対応づけると、$A_i\supseteq A_{i+1}$が成り立つことより、関係“$\prec\neq$”に関する切断の列

$$\mu_1\prec\neq\mu_2\prec\neq\cdots\prec\neq\mu_k \tag{4.1}$$

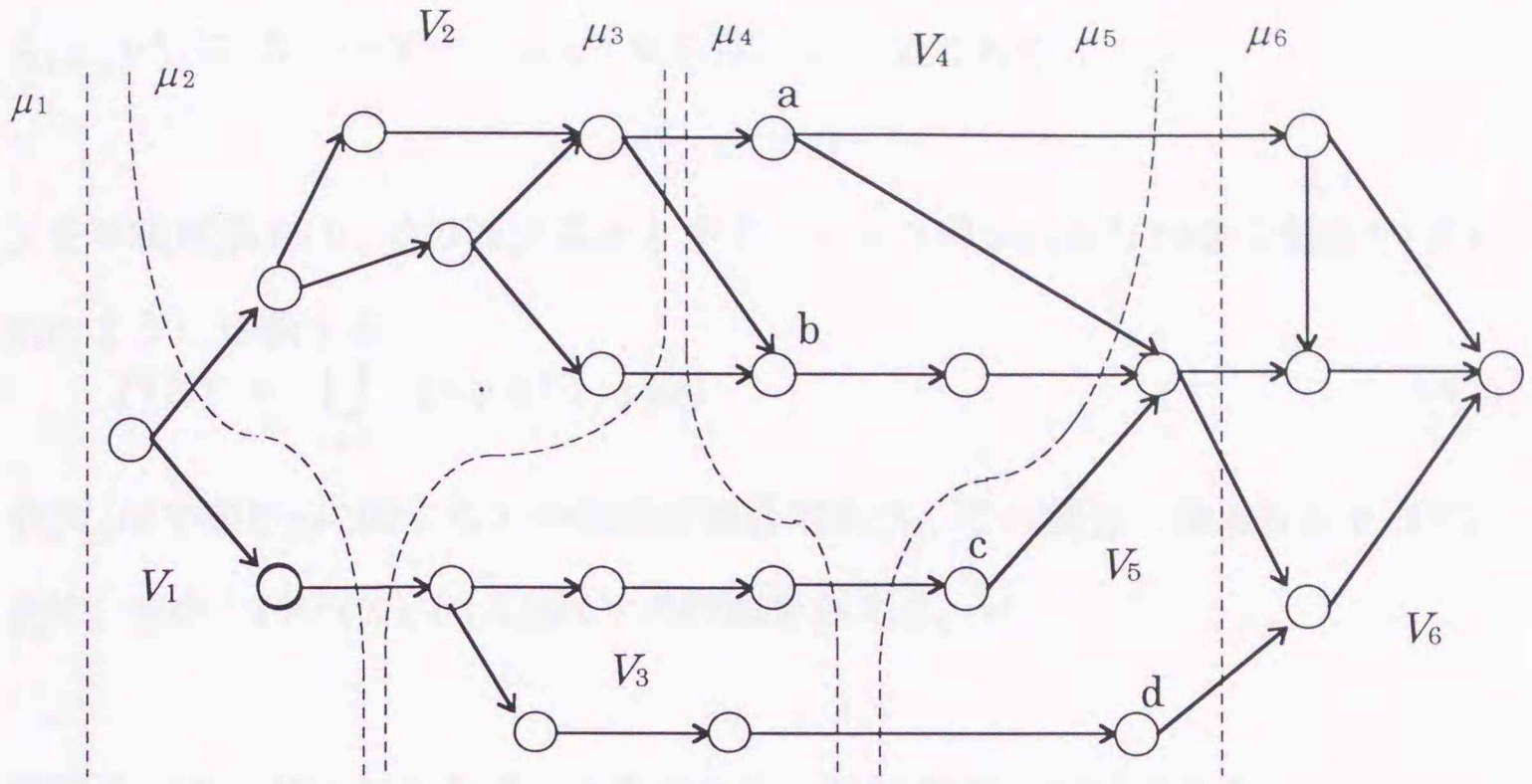


図４．１　無閉路有向グラフの系列分割

が得られる。このような引き続く２つの切断μ_i ,μ_{i+1} が常に関係$\prec\neq$で結ばれている切断の単調列のことを切断の鎖と呼ぶ。逆に、切断の鎖(４.1)が与えられたとき、隣り合う切断の上組をそれぞれA_i ,A_{i+1} として、$V_i = A_i - A_{i+1}$ を求めると、系列分割$V_1|V_2|...|V_k$ が定まる。以上のように、無閉路有向グラフの系列分割と切断の鎖は一対一に対応しており、任意の系列分割は切断の鎖によって一意に定まる。

４．１．２ 系列分割と切断の鎖

本章では、無閉路有向グラフの系列分割が切断の鎖によって一意に表せることを詳細な検討を加え論証する。まず、分離可能な集合に対して次の補題が成り立つ。

補題４．２　ＸとＹが切断により分離可能であるとき、

$\exists(x,y) \in X \times Y$； $x \prec y$ならば、X｜Yである。

Xを半順序集合$(V,\preceq)$の部分集合とする。Xから導かれるVの部分集合P(X)を次のように定義する。

$$P(X) = \bigcup_{x \in X} \{y | y \in V, \ y \succeq x\} \tag{4.2}$$

P(X)は半順序$\preceq$に関するXの後続者集合であり、この集合の補集合を$P^c(X)$で表す。また、$(P^c(X), P(X))$はVの切断を与える。

補題４．３　X｜Yならば、$X \subseteq P^c(Y)$, $Y \subseteq P(Y)$ が成立する。

証明： $Y \subseteq P(Y)$は自明である。$X \subseteq P^c(Y)$が成立しないとすると、$\exists x \in X$； $x \in P(Y)$ が成立し、このxに対して式(４.2)より、$\exists y \in Y$； $x \succeq y$を得る。これはXとYが比較可能な元を持たぬとき矛盾となる。また比較可能のときX｜Yという前提に矛盾する。

□

補題４．４　X,Y,Zは互いに分離可能なVの部分集合であり、またX｜Y,Y｜Z,Z｜Xというサイクルを作らないとする。このとき、X｜Y,Y｜ZからX｜Zが導かれる。

証明： 補題４．３を用いると、X｜Y,Y｜Zからそれぞれ、$X \subseteq P^c(Y), Y \subseteq P(Y)$および$Y \subseteq P^c(Z), Z \subseteq P(Z)$が成立する。P(Y),P(Z)の和集合を上組とする切断$(P^c(Y) \cap P^c(Z), P(Y) \cup P(Z)) = (P^c(Y \cup Z), P(Y \cup Z))$を作り、Xが下組に、Zが上組に含まれることを示すことによって、X｜Zの成立を証明する。まず、$Z \subseteq P(Y) \cup P(Z)$は明らかである。次に$X \subseteq P^c(Y) \cap P^c(Z)$を証明する。$X \subseteq P^c(Y)$は成立しているので、$X \subseteq P^c(Z)$を示すことで十分である。この条件が成立しないと仮定すると、$\exists x \in X$； $x \in P(Z)$で、XとZは互い

に素であるから、$x \in P(Z)-Z$となる。これより、$\exists z \in Z$； $x \succ z$が成立し、補題４．２からＺ｜Ｘが導かれ、補題の前提に矛盾する。

□

補題４．５ V_i を系列分割 $V_1, V_2, \cdots, V_k$ のある成分とするとき、$P(V_i)$に属するｘがV_iに属さないとき、ｘを含む分割成分V_jは$V_i \mid V_j$をみたす。

(証明)前提条件より、$\exists y \in V_i$； $y \prec x$が成立し、補題４．２より、$V_i \mid V_j$が得られる。

□

定理４．１　系列分割$V_1, V_2, \cdots, V_k$から選ばれた任意の３つ組V_i, V_j, V_rについて、$V_i \mid V_j$かつ$V_j \mid V_r$が成立すれば、$V_i \mid V_r$が成り立つ

証明： 上述の性質４．１と補題４．４から導かれる。

□

与えられた系列分割$V_1, V_2, \cdots, V_k$の各要素V_iに対して、Ｖの部分集合Ｑ(V_i)を次のように定義する。

$$Q(V_i) = P(V_i) \cup \left(\bigcup_{V_i \mid V_s} P(V_s) \right) \tag{4.3}$$

また、Ｑ(V_i)の補集合をＱ^c(V_i)で表す。このとき、(Ｑ^c(V_i),Ｑ(V_i))は切断であり、これを$\mu(V_i)$で表す。

補題４．６　系列分割$V_1, V_2, \cdots, V_k$ から取り出されたV_i, V_j に対して、$V_i \mid V_j$ が成立するための必要かつ十分条件はＱ(V_i)$\supset$Ｑ(V_j)である。

証明： 必要性を示す。$V_i \mid V_j$と定理４．１より、$V_j \mid V_r$に対して$V_i \mid V_r$が成立することから、

$$Q(V_j) = P(V_j) \cup (\bigcup_{V_j | V_r} P(V_r)) \subseteq \bigcup_{V_i | V_r} P(V_r) \subset Q(V_i)$$

十分性については、V_i, V_jが切断$(Q^c(V_j), Q(V_j))$により分離されることを示すことによって立証できる。

□

系列分割$V_1, V_2, \cdots, V_k$の各部分集合V_iに切断$\mu(V_i)=(Q^c(V_i), Q(V_i))$を対応づけると、切断の集合$M=\{\mu(V_1), \mu(V_2), \cdots, \mu(V_k)\}$が得られる。この集合に関して次の定理が成り立つ。

定理４．２　系列分割$V_1, V_2, \cdots, V_k$に対応する切断の集合$M=\{\mu(V_1), \mu(V_2), \cdots, \mu(V_k)\}$は切断の前後関係"$\prec=$"に関して全順序集合を形成する。

証明：　まず、Mが半順序集合であることを示す。反射律と反対称律の証明は簡単なので、推移律が成立することのみを証明する。すなわち $\mu(V_i) \prec= \mu(V_j)$、$\mu(V_j) \prec= \mu(V_r)$ が成立すれば、$\mu(V_i) \prec= \mu(V_r)$が成立することを示す。$\mu(V_i) \prec\neq \mu(V_j)$より、$Q(V_i) \supset Q(V_j)$であり、補題４．６より、$V_i \mid V_j$が成立する。$\mu(V_j) \prec\neq \mu(V_r)$より、$Q(V_j) \supset Q(V_r)$であり、$V_j \mid V_r$が成立する。よって、$V_i \mid V_r$が成立し、$Q(V_i) \supset Q(V_r)$となり、$\mu(V_i) \prec\neq \mu(V_r)$がみたされる。

Mが全順序集合となることを示すために、Mの任意の２元$\mu(V_i)$と$\mu(V_j)$の比較可能性を示せばよい。性質４．１の(2)より、V_iとV_j は分離可能である。一般性を失うことなく$V_i \mid V_j$とすれば、補題４．６より、$Q(V_i) \supset Q(V_j)$が導かれ、$\mu(V_i) \prec\neq \mu(V_j)$が成立する。

□

この定理より、M は$\prec=$に関して全順序集合となるので、その要素を大きさの

順に並べかえ単調増加列にすることができる。ここで、この増加列の要素の並びの順に従って系列分割の成分番号を付け替えて、

$$\mu(V_1) \prec\neq \mu(V_2) \prec\neq, \cdots, \prec\neq \mu(V_k) \tag{4.4}$$

が成立するようにする。この番号付けに従う系列分割

$$V_1, V_2, \cdots, V_k \tag{4.5}$$

を正規化された系列分割という。この対応する切断の単調増加列(４.4)が切断の鎖となる。以後、議論はすべて正規化された系列分割について行う。また簡単のために、ｉ＝１,２,･･･,ｋに対して、

$$\mu_i=\mu(V_i),\ Q_i=Q(V_i),\ P_i=P(V_i) \tag{4.6}$$

と置く。$\mu_i=(Q_i{}^c, Q_i)$と補題４．６を使うと、(４.4)式と等価な次の関係式が導かれる。

$$Q_1 \supset Q_2 \supset \cdots \supset Q_k \tag{4.7}$$

$$V_1 \mid V_2 \mid \ . \ . \ . \ \mid V_k \tag{4.8}$$

式(４.8)から明らかなように、$V_i \mid V_j$ なら、ｊ＞ｉである。それゆえ、正規化された系列分割(４.5)に対して、(４.3)式で定義された$Q(V_i)$は

$$Q(V_i) = \bigcup_{i \le j \le k} P(V_j) \tag{4.9}$$

と書き直すことができる。以上から次の諸定理を得る。

補題４．７　正規表現による系列分割(４.5)式において、$V_i \mid V_j$ならば、V_iとV_jは切断$(Q_{i+1}{}^c, Q_{i+1})$により分離できる。

証明： (４.9)式より、$Q_{i+1} = \bigcup_{i<s} P_s$である。またｉ＜ｊであるから、$V_i \subseteq P_i$はQ$_{i+1}$に含まれることなく、一方$V_j \subseteq P_j$はQ$_{i+1}$に含まれる。

□

定理４．３　正規表現による系列分割(４.5)式において、Q$_{k+1}$＝φと置けば、

次の関係式が満足される。

$$Q_i = P_i \cup Q_{i+1} \qquad (i=1,2,\cdots,k) \tag{4.10}$$

$$V_i = Q_i - Q_{i+1} \qquad (i=1,2,\cdots,k) \tag{4.11}$$

証明： まず、(4.10)式を証明する。(4.9)式より、

$$Q_i = \bigcup_{i \le j \le k} P_i = P_i \cup \left(\bigcup_{i+1 \le j \le k} P_j \right) = P_i \cup Q_{i+1}$$

次に、(4.11)式を示す。$P_k = V_k$に注意して、

$$Q_i - Q_{i+1} = (P_i \cup Q_{i+1}) \cap Q_{i+1}{}^c = P_i \cap Q_{i+1}{}^c$$

V_iとV_{i+1}は補題４．７より切断$(Q_{i+1}{}^c, Q_{i+1})$により分離されるので$V_i \subseteq Q_{i+1}{}^c$、また、$V_i \subseteq P_i$であるから、$V_i \subseteq P_i \cap Q_{i+1}{}^c = Q_i - Q_{i+1}$となる。逆に$x \in P_i \cap Q_{i+1}{}^c$が$V_i$に属さないとすると、補題４．５より xを含む分割成分V_mに対して、$m > i$である。それゆえ、(4.9)式より $P_m \subseteq Q_{i+1}$となり、$x \in V_m \subseteq P_m \subseteq Q_{i+1}$が導かれ、$x \in Q_{i+1}{}^c$に矛盾する。

□

一般に、切断の鎖$\mu_1 \prec\neq \mu_2 \prec\neq, \cdots, \prec\neq \mu_k$が与えられたとき、各切断を$\mu_i = (A_i{}^c, A_i)$と置き、さらに$A_{k+1} = \phi$とおけば、次式より $\{V_i\}$ が一意に定まる。

$$V_i = A_i - A_{i+1} \qquad (i=1,2,\cdots,k) \tag{4.12}$$

この$\{V_i\}$は切断の鎖によって定まる分離構造のもとで、性質４．１の(1),(2),(3)を満足し、系列分割を与える。これを切断の鎖により生成された系列分割という。定理４．２は与えられた系列分割から、切断の鎖(4.4)式が一意に定まり、これから生成された系列分割がもとの系列分割と一致することを述べている。以上を以下の定理にまとめる。

定理４．４　任意の無閉路有向グラフの系列分割は切断の鎖によって生成できる。

4．2 総カット値を最小化する系列分割問題の定式化

総カット値を最小化する系列分割問題で考えるネットワークは、多重辺をもたない単一の入口と出口を持つn個の頂点からなる無閉路有向グラフ$D(V,E)$であり、すべての頂点$v\in V$には重み$w(v)$が、各有向辺$(v_1,v_2)\in E$にはコスト$c(v_1,v_2)$が付与されている。これらの値は任意の$v\in V$および$(v_1,v_2)\in E$について、各々条件$0<w(v)\leq B, c(v_1,v_2)\geq 0$を満たす整数であり、また$B$はブロックサイズと呼ばれる問題に固有な正の整数である。

Kernighanのブレーク・ポイントは切断の上組の最小元として定義され、これを与えることによって切断が一意に定まり、無閉路有向グラフの場合にもこの考えを拡張して検討する。無閉路有向グラフ$D(V,E)$から導かれる半順序集合$(V,\preceq)$の切断を$\mu=(A^c,A)$とするとき、Aに制限した半順序集合$(A,\preceq)$の極小元の集合を切断μの切断決定子γと呼ぶ。図４．１における切断μ_4に対応する切断決定子は頂点集合{a,b,c,d}である。切断が与えられたとき切断決定子γは一意に定まり、逆に、γが与えられたとき、$A=\bigcup_{u\in\gamma}\{v|v\succeq u\}$とすることによって切断$(A^c,A)$が復元でき、切断と切断決定子の関係は１対１である。それゆえ、切断の鎖$\mu_1,\mu_2,\ldots,\mu_k$は切断決定子の列$\gamma_1,\gamma_2,\ldots,\gamma_k$によって表現できる。また、この鎖から生成される系列分割の成分集合の一つをV_iとするとき、それから誘導されるDの部分グラフ$\mathcal{D}(V_i)$をブロックと呼ぶ。この切断の鎖で$i<j$のとき、μ_i,μ_jの上組集合A_i,A_jの差集合A_i-A_jを記号$[\gamma_i,\gamma_j)$で表す。この記法によれば、$V_i=[\gamma_i,\gamma_{i+1})$と書ける。また、$V_i$に属するすべての頂点$v$の重み$w(v)$の総和 $|[\gamma_i,\gamma_{i+1})| = \sum_{v\in V_i} w(v)$をブロック長という。一方、どのブロックにも属さないDの辺の集合をカット・セットという。カット・セットに属する辺のコストの総和は

$$f_c(\gamma_1,\gamma_2,\cdots,\gamma_k) = \sum_{i=1}^{k-1} C(\gamma_i,\gamma_{i+1}) \qquad (4.13)$$

と書き表すことができる。ここで、$C(\gamma_i,\gamma_{i+1})$は切断決定子γ_iの後ろにγ_{i+1}を置いたときの増分コストと呼ばれる量であり、始点がV_iの中にあり、終点が$j>i$をみたすV_jの中にあるすべてのカット辺のコストの総和を表している。

$$C(\gamma_i,\gamma_{i+1}) = \sum_{\substack{v\in[\gamma_i,\gamma_{i+1}),\\ u\in P}} c(v,u) \qquad (4.14)$$

ここで、$P=V_{i+1}\cup\ldots\cup V_k$である。

以上より、カット値の総和を最小にする系列分割問題はブロック長がブロックサイズBを越えないという制約のもとで、カット・セットに属する辺のコストの総和を最小にする系列分割を求める問題であり、制約条件$|[\gamma_i,\gamma_{i+1})|\leq B$を満たすすべての切断決定子列の集合の上で、目的関数$f_c(\gamma_1,\gamma_2,\ldots,\gamma_k)$を最小化する問題として定式化される。

4．3 一列化グラフの最適系列分割問題と本問題の関連

無閉路有向グラフの一列化とは半順序を全順序に埋め込むことであり、図表現によるグラフ表現ではVの頂点を直線上で並べ替え、すべての辺の矢線の向きを常に右側に向くようにすることである。このようにして得られたグラフを一列化グラフ、また、左端から右端まで各頂点に並びの順に連続的に１からｎまでの番号を付した番号付けを一列化による番号付けという。無閉路有向グラフはいつでも一列化可能であるが結果は一意ではない。一般に複数の一列化による番号付けが可能である。

ある特定の一列化に限定し頂点の番号付けを行う。番号ｉに対して、ｉ以上の番号ｊ(≧ｉ)を持つ頂点の集合をＡとすれば、(A^c,A)がＶの切断となることは明かである。また同じ条件のもとで番号の単調増加列$1\leq i_1<i_2<\ldots<i_k<n$を与えたとき、列要素$i_j$に対応する切断を$\mu_j$とする切断の列$\mu_1,\mu_2,\ldots,\mu_{k+1}$は正規化された切断の鎖となり、無閉路有向グラフの系列分割が定まる。この系列分割で、そのすべてのブロック長がブロックサイズＢを越えないという制約

のもとで、カット辺のコストの総和を最小にする分割を求める問題はKernighanの最適系列分割問題そのものである。これを与えられた無閉路有向グラフから導かれた一列化グラフにおける最適系列分割問題という。ここで頂点番号の単調増加部分列はKernighanの問題の定式化におけるブレーク・ポイントの列の役割を果たしている。逆に、任意に与えた無閉路有向グラフの系列分割について次の定理が成り立つ。

定理４．５　無閉路有向グラフの任意の系列分割は、ある一列化による頂点の番号付けと、頂点番号の単調増加部分列を指定することによって定まる。

証明：　Ｄの系列分割は正規表現のもとでＶ１｜Ｖ２｜･･･｜Ｖｋをみたす。各Ｖｉによって誘導されるＶの部分グラフを一列化グラフＤｉに書き直し、Ｄｉを添字の順に左から右に一列に並べ、それらをカットセットに属する辺で結びつけると、Ｄの一列化グラフが得られる。ここで、この一列化による頂点の番号付けと、各Ｄｉの最左端の頂点番号からなる単調増加部分列から一列化の系列分割を作れば、これは与えられた無閉路有向グラフの系列分割と一致する。

□

この定理から、無閉路有向グラフＤの最適系列分割問題の解は、Ｄから導かれたある一列化グラフに対する最適系列分割問題の解であることが導かれる。これを利用して、Ｄの最適系列分割問題の解を求めるために、"Ｄのすべての一列化グラフを列挙し、それらのすべてについて一列化グラフに対する最適系列分割問題の解を求め、その中で最良のものを解とする。"という原始的な算法を導くことができる。これを一列化にもとづく算法とよぶ。

４．４　まとめ

本章ではKernighanの一列化グラフの最適系列分割問題をより一般化した無閉

路有向グラフの最適系列分割問題に拡張し、先行順位のある要素をその先行順位の制限を無視することなく、いくつかのグループに配置する問題を考えた。このとき、各グループに課せられた制約条件のもとで、最良の評価値をもつ配置を求めるものとする。典型的な事例としてライン・バランシングの問題[1],[4]、工場配置の問題などがある。例えば、ライン・バランシングの問題は先行順位の制限を無視せずに、要素作業を作業ステーションの数が最小になるように各作業ステーションに割り当てることであり、工場配置問題では工程の先行順位を無視することなく各工場間の中間製品の全移動コストの最小化を達成する配置を求める。この種の問題は各グループに課せられた制約条件、配置の評価関数などを変えることにより、種々の配置問題に適応させることが可能である。これらの問題はすべて無閉路有向グラフの頂点を先行順位の関係を無視せずに各成分に配置する基本的分割問題の範疇に属するものと考えられる。この基本的分割問題を無閉路有向グラフの系列分割問題として提案した。

さらに、系列分割問題の一般的定式化と、その実例として無閉路有向グラフを系列分割するときに生ずるカット・エッジのコストの総和を最小化する問題を取り扱い、この問題を“総カット値を最小化する系列分割問題”と定義した。その事例として、次のような問題が考えられる。先行順位を持ついくつかの生産プロセスからなるシステムを考える。各要素プロセスは、その先行プロセスの生産物を入力として受け取り生産活動を行い、その産出物をそれを必要とする後続プロセスに渡す。また、各プロセスは生産活動のために一定の資源量を必要とするが、１ステーションあたり使用できる資源量は定まっているとする。中間製品の移動には各ステーション間で、ある輸送コストを必要とするが、同一ステーション内の輸送コストは無視できるものとする。このとき、各生産プロセスを輸送コストの総和が最小になるように、先行順位の制限を無視せずに、ステーションごとに利用可能な資源量の範囲内で各ステーションに配置する問

題が挙げられる。

また、このような要素が先行順位関係をもつシステムの群配置問題はコスト関数と制約条件をそれぞれの問題の特性に合わせることによって、多様な問題へ適応可能である。

参考文献

[1] Betts,j. and Mahmoud,k.I.: A Method for Assembly Line Balancing, Engineering Costs and Production Economics, Vol.18,pp55-64(1989).

[2] Kernighan,B.W. and Lin,S. : An Efficient Heuristic Procedure for Partitioning Graphs, Bell System Tech. J., Vol.49, No.2, pp.291-307(1970).

[3] Kernighan,B.W. : Optimal Sequential Partitions of Graphs, J.ACM, Vol.18, No.1, pp.34-40(1971).

[4] Salveson,M.E.: The Assembly Line Balancing Problem, The Journal of Industrial Engineering, May-June, pp.18-25(1955).

[5] 秋山、西関: グラフとダイグラフの理論, 共立出版(1981).

[6] 加地，大内 ：最適系列分割問題に対する効率的分枝限定法の構築と諸特性解析，情報処理学会論文誌，Vol.35, No.3, pp.364-372(1994).

[7] 加地 ： 半順序の最適系列分割問題の構造と算法構成， 商学討究（小樽商科大学）, Vol.45, No.2, pp.185-204(1994).

[8] 加地、大内： 要素間に先行順位をもつシステムの配置問題, 電気学会論文誌 C 分冊, Vol.117-C, No.2, pp.136-142(1997).

[9] 加地 ： 要素間に先行順位をもつシステム要素の配置問題に対する厳密解法と近似解法の提案, 日本経営工学会論文誌, Vol.47-C, No.6, pp.344-350(1997).

第５章　一列化グラフの最適系列分割問題に対する解析と分枝限定法による解法

Kernighan による一列化グラフの最適系列分割問題とそのアルゴリズムについて第３章で述べた。この問題は提案する無閉路有向グラフの最適系列分割問題の中での重要な構成要素となり、また、この問題の解法が今後多数回反復使用される観点から、その解法の性能は重要な問題の一つとなる。そこで本章では第３章で示された Kernighan のアルゴリズムに対して詳細な分析を試み、その特性と問題点について検討を加える。さらに、Kernighan の算法に対して探索法および限定操作の観点から改善の余地があるものと考え、この一列化グラフの最適系列分割問題に対して分枝限定法の考え方による効率的な算法を提案する。また、提案された算法の特性と性能について詳細な検討を加え、その有効性を確かめる。以上により、無閉路有向グラフの最適系列分割問題における効果的な算法の開発の基礎とする。

５．１　動的計画法によるアルゴリズムの特性と評価

Kernighan は動的計画法を適用したＫＤＰアルゴリズムを示し、その計算量についてエッジ数の変化のみからの検討を加えている。しかし、アルゴリズム全体の詳細な計算量および、その特性については大きくふれてはいない。ここで、我々はＫＤＰの詳細な計算時間の特性とそのパフォーマンスについての検討を試みる。初めに、Kernighan によって定義されたアルゴリズムの計算量を理論的に導く。この場合、アルゴリズム中で使われている基本的演算が加算と比較の

みであるので、これらの演算数に着目して計算量を求める。次に、このアルゴリズムの具象化のレベルにおいてデータ構造の詳細化および、それによって生じる負担を考慮に入れた評価を行う。このとき、効果的なアルゴリズムを実現するための具体的な構築法を述べる。さらに、数値実験を行って評価値を確かめ、計算量に関して、ブロックサイズが変動する場合や、エッジ数が疎および密である場合を考慮し考察する。

５．１．１　理論的計算量の算出

Kernighan が与えた算法を解析することにより、計算量を算出する。この算法で使われている基本的演算は加算と比較であり、これらの演算数に着目して計算量を導く。

セクション３．２．１で述べたKernighanのＤＰアルゴリズムにおけるステップ１（Set T(1)=0; L(1)=0;）とステップ３（Set Total_Cost=T(n+1); Set z_0=n+1; Set k=0;）は単なる代入であり、ステップ５（ブレイク・ポイントは大きいほうから $z_1,z_2,\ldots,z_k$ である。）は出力である。ステップ４（while (z>1){z_{k+1}=L(z_k); k=k+1;}）はたかだか n/B 程度の計算量であり、主な計算量が生ずるステップ２（for x=2 to n+1 do {T(x)=$\min_y$[T(y)+C_f(y,x)]}）を解析する。以後、簡単化のために、すべての頂点の重みが１であると仮定する。（すべての重みをそれらの最小値に等しくさせた場合は計算量は上限値となり、すべての重みをそれらの最大値に等しくさせた場合は計算量は下限値となる。したがって重み１の場合は考えられる計算量の上限値である。）　このことによって、$d(y,x-1) \leq B$ は $x-y \leq B$ と置き換えられる。

ステップ2を関数方程式T(x)の計算、増分コストC_f(y,x)の計算、およびC_f(y,x) を計算するために必要な中間量W^x, Δ^xの計算に分解してステップ2の計算量を考察す

る。

（1）　関数方程式Ｔ（ｘ）の計算

ステップ２のＤＰの漸化式による計算過程は表形式を用いると理解し易い、それを表５．１によって示す。この三角表の最上位行はこれから決定すべきＴ（ｘ）；ｘ＝２，３，・・・，n＋１を書き込む欄である。最右端の列は漸化過程ですでに求まったＴ（ｘ）の値を複写しておく欄である。Ｔ（ｘ）を求めるには、Ｔ（ｘ）を含む列の各要素 C_f(x-i, x)；ｉ＝１，２，・・・，ｘ－１　と、対応する最右端列の要素Ｔ（ｘ－ｉ）；ｉ＝１，２，・・・，ｘ－１　の要素ごとの和をつくり、それらの中の最小値を求めて、それをＴ（ｘ）とする。このようにして、与えられた初期値Ｔ（１）＝０から出発して順次Ｔ（２），Ｔ（３），・・・，Ｔ（n＋１）と漸化的に求めていくことができる。

表５．１から明らかなように最小値を求めるためのｙの範囲はＴ（２），・・・，T(B)に対しては階段的に増大しており、一方 T(B+1)，・・・，Ｔ（n＋１）に対しては常に一定幅 B のみに着目すればよい。そこで必要なすべての C_f(y,x)がすでに計算されているという仮定のもとで、Ｔ（２），Ｔ（３），・・・，T(B)を求めるのに必要な計算量は加算と比較に着目して、

$$2(1+2+3+\ldots+(B-1))=B(B-1)$$

となり、一方、T(B+1)，・・・，Ｔ（n＋１）を求めるのに必要な計算量は

$$2B(n-B+1)$$

となる。この両者を加えて、増分コストがすでに計算されているという仮定のもとで、Ｔ（２），・・・，Ｔ（n＋１）を決定するのに要する計算量は

$$B(B-1)+2B(n-B+1)$$

$$=2Bn-B^2+B \qquad (5.1)$$

表５．１ 増分コストの上三角タブロー

$T(2)$	$T(3)$	…	$T(B)$	$T(B+1)$		$T(x\text{-}1)$	$T(x)$		$T(n+1)$	
$C_f(1,2)$	$C_f(1,3)$	…	$C_f(1,B)$	$C_f(1,B+1)$		$C_f(1,x\text{-}1)$	$C_f(1,x)$			$T(1)$
	$C_f(2,3)$		$C_f(2,B)$	$C_f(2,B+1)$		$C_f(2,x\text{-}1)$	$C_f(2,x)$			$T(2)$
						…	…	…		⋮
			…	…	…	$C_f(x\text{-}B+1,\ x\text{-}1)$				
						$C_f(x\text{-}B,\ x\text{-}1)$	$C_f(x\text{-}B,x)$			$T(x\text{-}B)$
			$C_f(B\text{-}1,B)$							⋮
				C_f		…	…	…		$T(B)$
				$(B,B+1)$						
										⋮
						$C_f(x\text{-}i,x\text{-}1)$	$C_f(x\text{-}i,x)$			$T(x\text{-}i)$
										⋮
						$C_f(x\text{-}2,x\text{-}1)$	$C_f(x\text{-}2,x)$			$T(x\text{-}2)$
							$C_f(x\text{-}1,x)$			$T(x\text{-}1)$
										⋮
										$T(n)$

と求まる。

（2）　増分コスト C_f（ y, x ）の計算

次に既知と仮定した増分コスト C_f(y, x)はKernighanが指摘しているように漸化式

$$C_f(y, x) = C_f(y, x\text{-}1) + (c(x\text{-}1,x) + c(x\text{-}1, x\text{+}1) + \ldots + c(x\text{-}1, n)$$

$$- (c(y,x\text{-}1) + c(y\text{+}1, x\text{-}1) + \ldots + c(x\text{-}2, x\text{-}1) \qquad (5.2)$$

を満たすのでこれを用いて計算する。この漸化式を中間量W^xとΔ^xを用いて、

$$C_f(\text{x-i, x}) = C_f(\text{x-i, x-1}) + W^x - \Delta_i^x \quad ,\text{i=1, 2, } \ldots \text{, x-1} \qquad (5.3)$$

と置く。ここで

$$W^x = \text{c(x-1, x)} + \text{c(x-1, x+1)} + \ldots + \text{c(x-1, } n) \qquad (5.4)$$

であり、Δ_i^xは次式である。

$$\Delta_1^x = 0$$

$$\Delta_j^x = \Delta_{j-1}^x + \text{c(x-j, x-1)} \quad ,\text{j=2, } \ldots \text{ ,x-1} \qquad (5.5)$$

必要なすべてのW^xとΔ_i^xがすでに計算されているという過程のもとで(5.3)式からわかるように各々C_f(y, x)を求めるのには２回の演算が必要である。そこで前述の計算過程で必要とされるC_f(y, x)の総数はT（２）からT(B)までの階段的増大部分とT(B+1)からT（n＋１）までの一定幅Bの中にあるCf(y, x)の個数の総和 $(2\,B\,n - B^2 + B) / 2$ となる。したがって(5.3)式の全増分コストの計算量は、２回の演算を考慮すると、

$$2\,(2\,B\,n - B^2 + B) / 2$$

$$= 2\,B\,n - B^2 + B \qquad (5.6)$$

である。

（３）　W^x，Δ_i^x の計算

既知であると仮定したW^{x}とΔ_i^xについての計算量を求める。このとき非零要素の参照は無視するという立場をとる。任意の頂点ｘにおけるW^{x}の加算演算数は(5.4)式の非零要素のみ着目してｏｄ（ｘ－１）－１と算出される。ここでｏｄ（ｘ）は頂点ｘにおける出次数である。すべての頂点に対してW^{x}の演算数を加えると

$$\sum_{x=2}^{n+1} \left(\mathrm{od}\left(\mathrm{x}-1 \right) -1 \right) = |\mathrm{E}| - n \qquad (5.7)$$

となる。

一方Δ_i^xについては任意の頂点ｘにおいて、漸化式(5.5)の計算にあたって非零要素の加算演算のみに着目すると、$id_{x-y<B}(x\text{-}1)$となる。ｉｄ（ｘ）は頂点ｘにおける入次数であり、流入辺の始点をｙとするとき、条件ｘ - ｙ ≤ Ｂを満たす入次数を$id_{x-y<B}(x)$で表わしている。すべての頂点ｘに対してこれらの総和を求めると

$$\sum_{x=2}^{n+1} id_{x-y<B}(x-1) = \sum_{x=2}^{n+1} id(x-1) - \sum_{x=2}^{n+1} id_{x-y>B}(x-1)$$

$$= |E| - |E_{x-y>B}| \qquad (5.8)$$

となる。ここで、$E_{x-y>B}$は$x - y > B$の条件をみたす辺の総数である。

（４）　ステップ２全体の計算量

ステップ２全体の計算量は以上の(5.1), (5.6), (5.7)および(5.8)式で与えられた量の総和であり、

$$(2Bn - B^2 + B) + (2Bn - B + B) + (|E| - n) + (|E| - |E_{x-y>B}|)$$

$$= 4Bn - 2B^2 + 2B + 2|E| - |E_{x-y>B}| - n \qquad (5.9)$$

となる。この(5.9)式から以下のことがわかる。

1) このＤＰアルゴリズムにおける計算量は頂点数と辺数に対しては比例増加となり、ブロックサイズに対してはn（頂点数）＋１／２が最大値となる上に　凸な増加曲線部分となる。

2) 辺数の上限値はn^2であり、辺数が頂点数に依存すると仮定した場合，　(5.9)

式のＥの項のみに着目する計算量は、グラフが疎な場合、すなわち、｜Ｅ｜＝ｏ（n^2）のとき、Ｏ（n）であり、密な場合Ｏ(n^2)である。またブロックサイズの上限値はnであり、ブロックサイズが頂点数に比例すると仮定した場合、Bn項のみに着目する計算量は疎でも密でもＯ（n^2）となる。したがって（５．９）式の計算量はグラフの疎密に関係なくＯ（n^2）である。

すなわち、Kernighan が述べるところの「計算量が辺数に比例する」以外に、頂点数に比例し、ブロックサイズについて上に凸な増加曲線を示し、およびこれらの相乗効果からなることがわかる。

５．１．２　アルゴリズムの具象化

次の２点を考慮することによって（５・９）式の理論計算量を実現する具象化アルゴリズムを構成することが可能である。

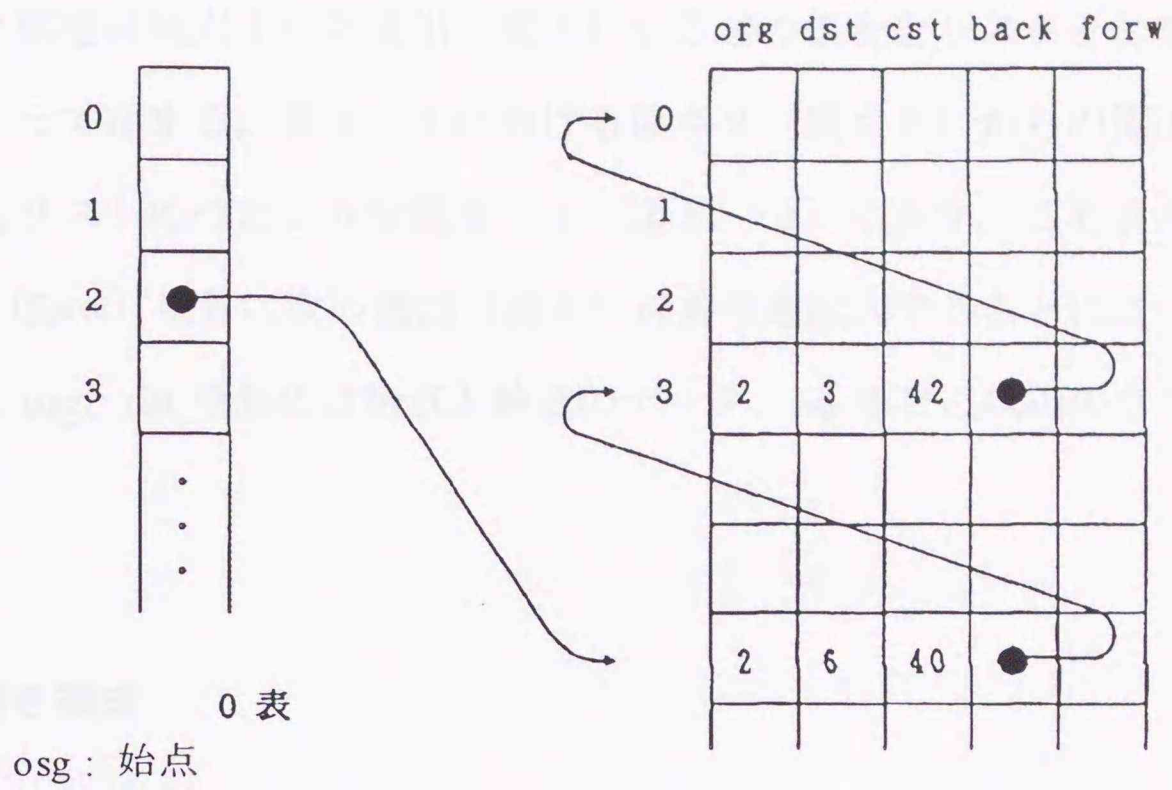
(a) 任意の頂点から流出（流入）する非零コストの辺のみに着目し、これらの辺をリストにつないでおく。

(b) 増分コストの計算にあたっては前に計算したデータを積極的に使用する漸化式方式を採用する。その計算過程で、上述のリスト構造を有効に利用する。

以下、アルゴリズムの具象化について述べる。

（１）　データ構造

処理上ｉとｊを端点としてもつ辺が存在するとき、ｉ＜ｊの順にｉを始点、ｊを終点とする有向グラフを考える。この有向グラフを表わすために、Ｅ表，Ｏ表（Ｄ表）からなるデータ構造を採用する。図５．１（図５．２）に示すようにＯ表（Ｄ表）は１項目のみからなるレコードの集合で、グラフの各頂点番号を保持している。またＥ表は多項目のレコードからなり、辺のデータを保持



osg : 始点
dst : 終点
cst : エッジコスト

図５．１ 流出辺リスト

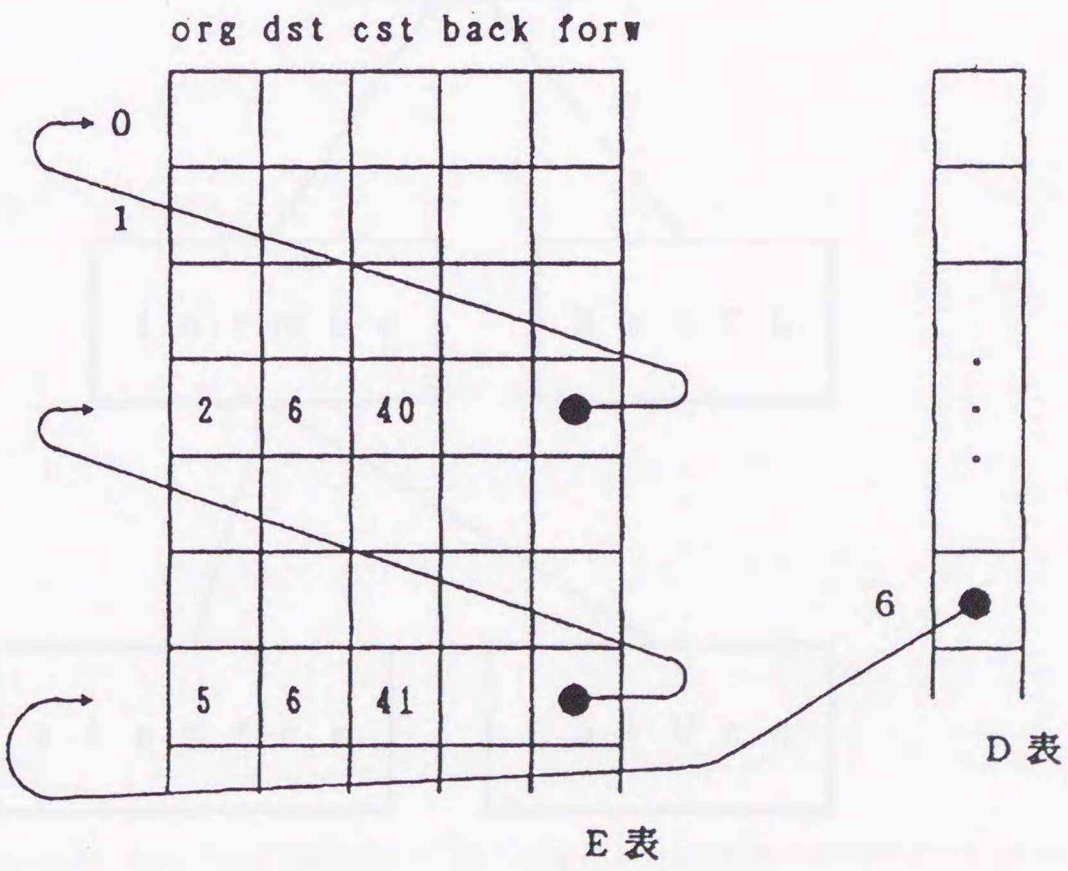


図５．２ 流入辺リスト

する。グラフ構造は始点 i から流出（流入）する辺の集合をリストとして指定することによって定まる。図３．１における頂点２（頂点６）からの流出（流入）に関するリストのつながりを図５．１（図５．２）に示す。このリストはＥ表の back（forw）項目に次の流出（流入）辺番号を記入することによって作成する。また org、dst 項目には始点と終点のデータ、cst 項目には辺のコストを格納する。

（２）　手続き構成

具象化されたプログラムは図５．３に示す手続き構成によって構成される。すべての関数方程式Ｔ（ｘ）；ｘ＝２，・・・，$n+1$ を（３．３）式から求め

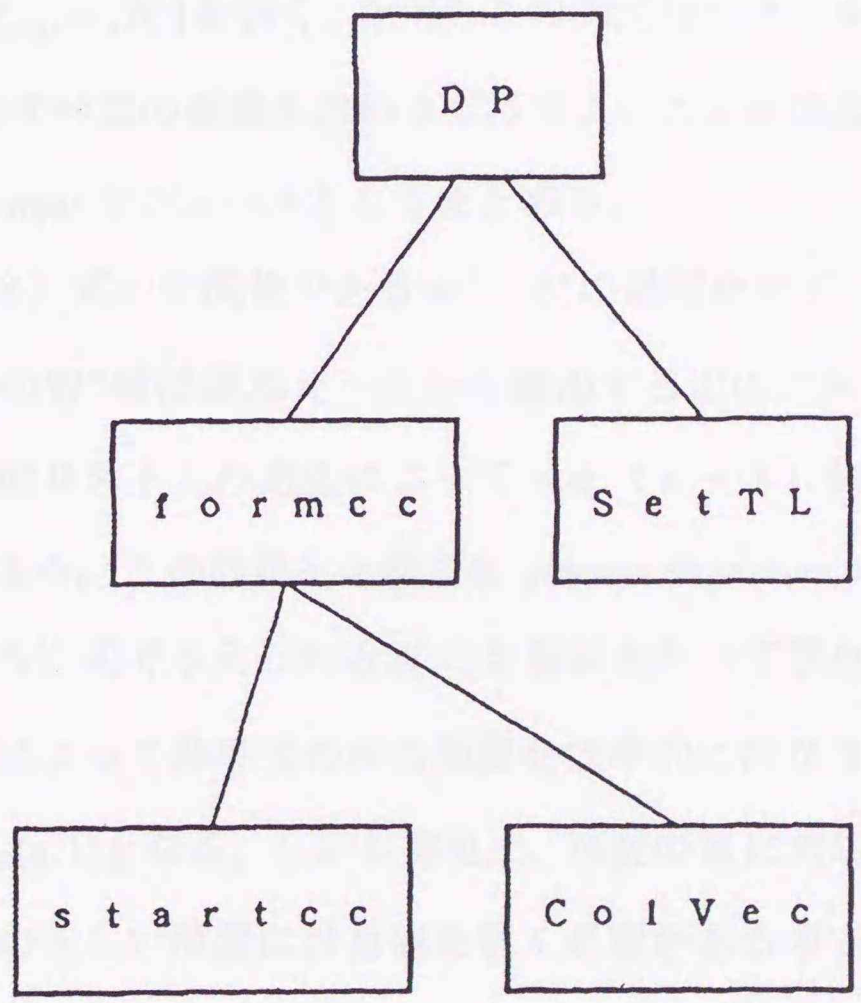


図５．３　手続き構成

る過程は、表５．１において最左端の列から順次、左から右へ最後の列まで計算していくことである。したがって１次元配列のみを用いて更新していくことが可能である。これを図５．３の最上部にあたる手続き名 *DP モジュール*としてまとめる。

次にその各部分となる任意のｘに対するＴ（ｘ）を（３．３）式を用いて求める。計算過程は手続き名 *SetTL モジュール*でまとめる。

また（５．３）式を使って必要な増分コスト計算を行なう過程を表５．１を用いて説明する。Ｔ（ｘ）に対応する（ｘ－１）成分列ベクトル[C_f(1,x), C_f(2,x), ・・・, C_f(x-1,x)] を求めるには、一つ前のＴ（ｘ－１）に対応する（ｘ－２）成分列ベクトルに第（ｘ－１）成分として、０を拡張した（ｘ－１）成分列ベクトル [C_f(1,x-1), C_f(2,x-1), ・・・, C_f(x-2,x-1), ０] の各要素に一定値W^xを加え、 それから $[\Delta^x_{x-1}, \Delta^x_{x-2}, \cdots, \Delta^x_1]$を引く。ただしこの列成分ベクトル要素は最後の成分からＢ個未満のすべての要素を求めるだけでよいことを注意しておく。この過程を手続き名 *formcc モジュール*としてまとめる。

最後に（５．３）式の中間量であるw^x, Δ^x_iの過程を示す。（５．４）式で与えられる頂点ｘのW^x値は頂点ｘ－１から流出する辺のコストの総和であることより単に流出辺リスト上の走査によってｏｄ（ｘ－１）個の項の加算で無駄なく計算可能である。この計算を手続き名 *startcc モジュール*としてまとめている。次に（５．５）式で与えられるΔ^x_iの計算にあたって頂点ｘ－１への流入辺リスト上の走査によって非零項のみの加算を効率的に行なうことができ、 その計算量は$id_{x-y<B}(x-1)$となる。しかし実現上、可変のＢに対応する必要があり、また表５．１上の正しい位置に計算値を置く必要があるので、そのため余分の計算コストが必要である。この計算過程を手続き名 *colvec モジュール*としてまとめた。

以上のプログラムの主要部をＣ言語で実現したものを示す。

{ＤＰ手続き}

```
for (x=2; x<n+2; x++) {
  formcc(x);
  SetTL(x,B);
}
```

{Ｓｅｔ ＴＬ手続き}

```
minval = MAXVAL;
for (y=unitf(x-B); y<x; y++) {
  val = T[y] + CC[y];
  if (val < minval) {
    minval = val;
    miny = y;
  }
}
T[x] = minval;
L[x] = miny;
```

{ｆｏｒｍｃｃ手続き}

```
outd = startcc(x);
cc[x-1] = outd;
ColVec(x-1,V);
ind = 0;
for (y=x-2; y>0 && y>=x-B; y--) {
  ind += V[y];
  CC[y] = CC[y] + outd - ind;
}
```

{ｓｔａｒｔｃｃ手続き}

```
W=0;
for (j=O[x-1]; j>0; j=E[j].back)
  W += E[j].cst;
return W;
```

{ＣｏｌＶｅｃ手続き}

```
for (j=D[y]; j>0; j=E[j].forw)
  col[E[j].org] = E[j].cst;
```

５．１．３ 具象化されたアルゴリズムの計算量

表５．２ プログラムを構成する基本演算処理

基本処理演算	演算時間記号	処理時間（１０$^{-7}$ｓ）
ループ処理	τ（loop）	３７
代入処理	τ（＝）	１６
添字が変数である配列などにアクセスするる処理時間	τ（[val]）	１０
＋＝による加算代入処理	τ（+=）	３６
戻り値の処理時間	τ（return）	２０
・	・	・
・	・	・
・	・	・

第５．１．２章における具象化されたアルゴリズムの計算量を求める。表５．２はプログラムを構成する各基本処理の演算時間を記号で表し、また今回実験で用いた EPSON(PC-286)上での、それらの演算時間を示した。図５．３で示された各手続きの計算量はそれらを構成する基本処理演算をその演算時間記号で置き換えることによって求まる。最終的にＤＰ手続きの計算量の値が本問題におけるアルゴリズムの主要点であり、ステップ２に対応するので、これを求める。

計算の実例として、手続き startcc の導出を示す。startcc プログラムに対して以下のように表５．２の演算時間記号で置き換えると、

ｓｔａｒｔｃｃの計算時間

＝τ（＝）

＋τ（＝）＋τ（[val]）

$$+\tau\ (\text{loop})+\tau\ (\text{処理}_1)$$
$$+\tau\ (\text{loop})+\tau\ (\text{処理}_2)$$
$$\cdot\quad\cdot\quad\cdot$$
$$+\tau\ (\text{loop})+\tau\ (\text{処理}_{\text{od}(x-1)})$$
$$+\tau\ (\text{return}) \tag{5.10}$$

となる。

このループの回数はｏｄ（ｘ－１）である。また、処理$_i$の計算量はτ（+=）＋τ（[val]）となるので、

$$=\text{od}(x-1)\cdot(\tau\ (\text{loop})+\tau\ (+=)+\tau\ ([\text{val}]))$$
$$+2\tau\ (=)+\tau\ ([\text{val}])+\tau\ (\text{return}) \tag{5.11}$$

となる。

以上のような導出法によって、 ColVec, formcc, SetTL, DP の計算量を求めると、最終的にＤＰ手続きの計算量は

$$\text{ＤＰの計算量}=\lambda nB-\beta B^2+\nu B+\eta n+\mu|E|+\Phi \tag{5.12}$$

とまとめることができる。結果として（５．１２）式は（５．９）式とほぼ同様な計算量式となる。ただし、ブロックサイズに対しての変化は（$\lambda n+\nu$）／２βが最大値となる上に凸な曲線である。

さらに（５．１２）式における係数の値を具体的に求める。表５．２のEPSON(PC-286)上での処理時間を用いると、同計算機上において（５．１２）式は以下のようになる。ただし、係数の値には10^{-7}が乗算され、以後この値は省略する。

ＤＰの計算量（10^{-7}ｓ）
$$=292\cdot nB-164.5\cdot B^2+164.5\cdot B$$
$$+497\cdot n+162\cdot|E|+16 \tag{5.13}$$

となる。

しかし本プログラムにおいて初期設定などの必要不可欠なプロシェジャーが存在するため、上記計算量に対して微少な係数であるが$\phi\ n^2$の計算量が現実には存在する。

５．１．４　実験的検証

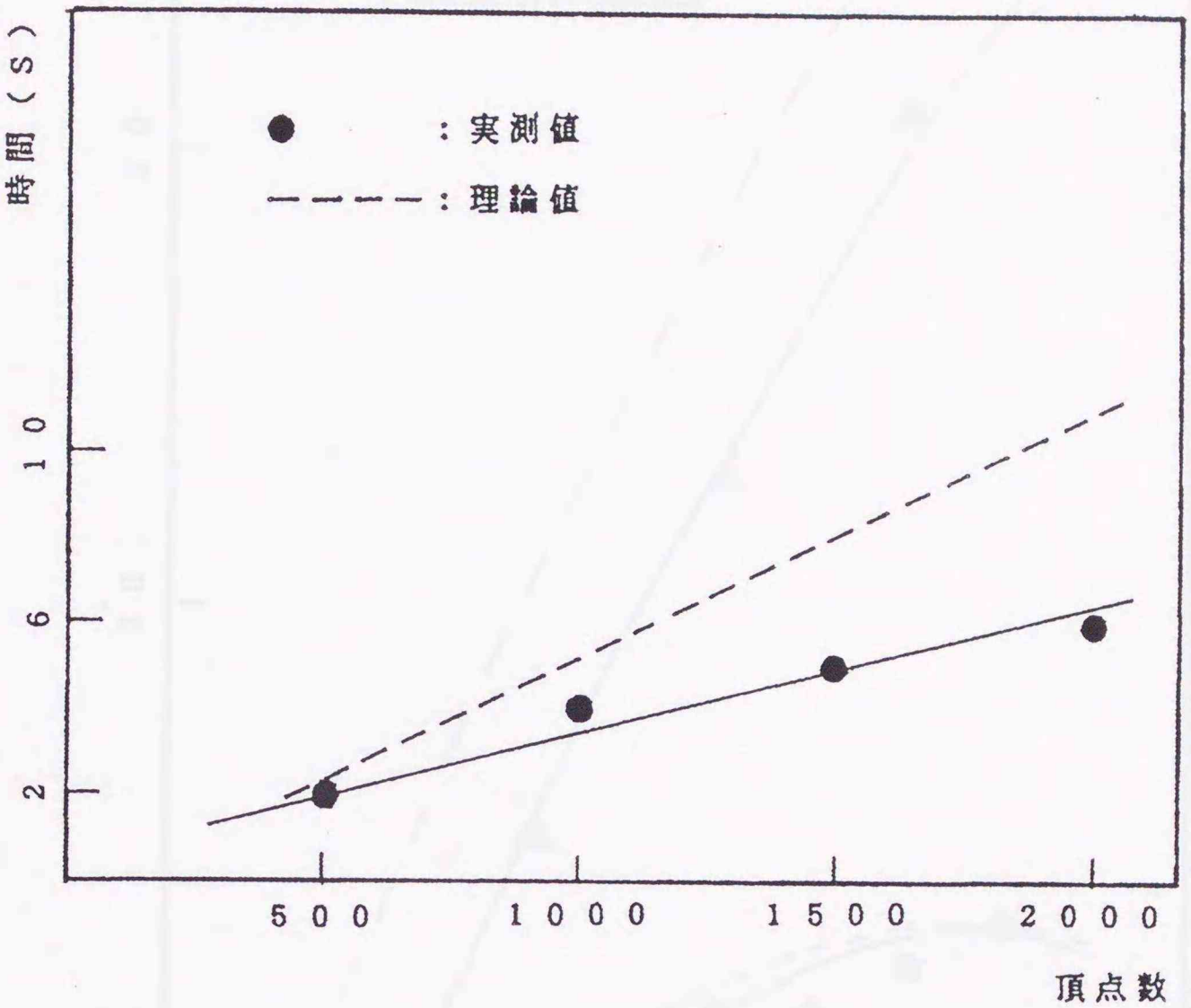
現段階の実験では　EPSON(PC-286)を用い実測を行なった。ただし、ハード的に秒単位以上しか測定できないので秒以下の時間と、実際のプログラムにおける種種の雑損となる計算量を考慮しなければならない。この非本質的部分である計算量は$\phi\ n^2$とする。実測値ではϕ＝８０であり、以下の実験では$\phi\ n^2$を引いた値で考察する。

（１）　頂点数による変化

頂点数を５００から２０００まで増大させたときの計算時間を図５．４に示す。ただしブロックサイズは２００、辺数は頂点数の２倍の割合とする。（５．１３）式からの理論値を破線で示す。実験値は理論値と比べて傾斜はややゆるやかであるが、ほぼ一致したエッジ数に比例した傾向を示す。

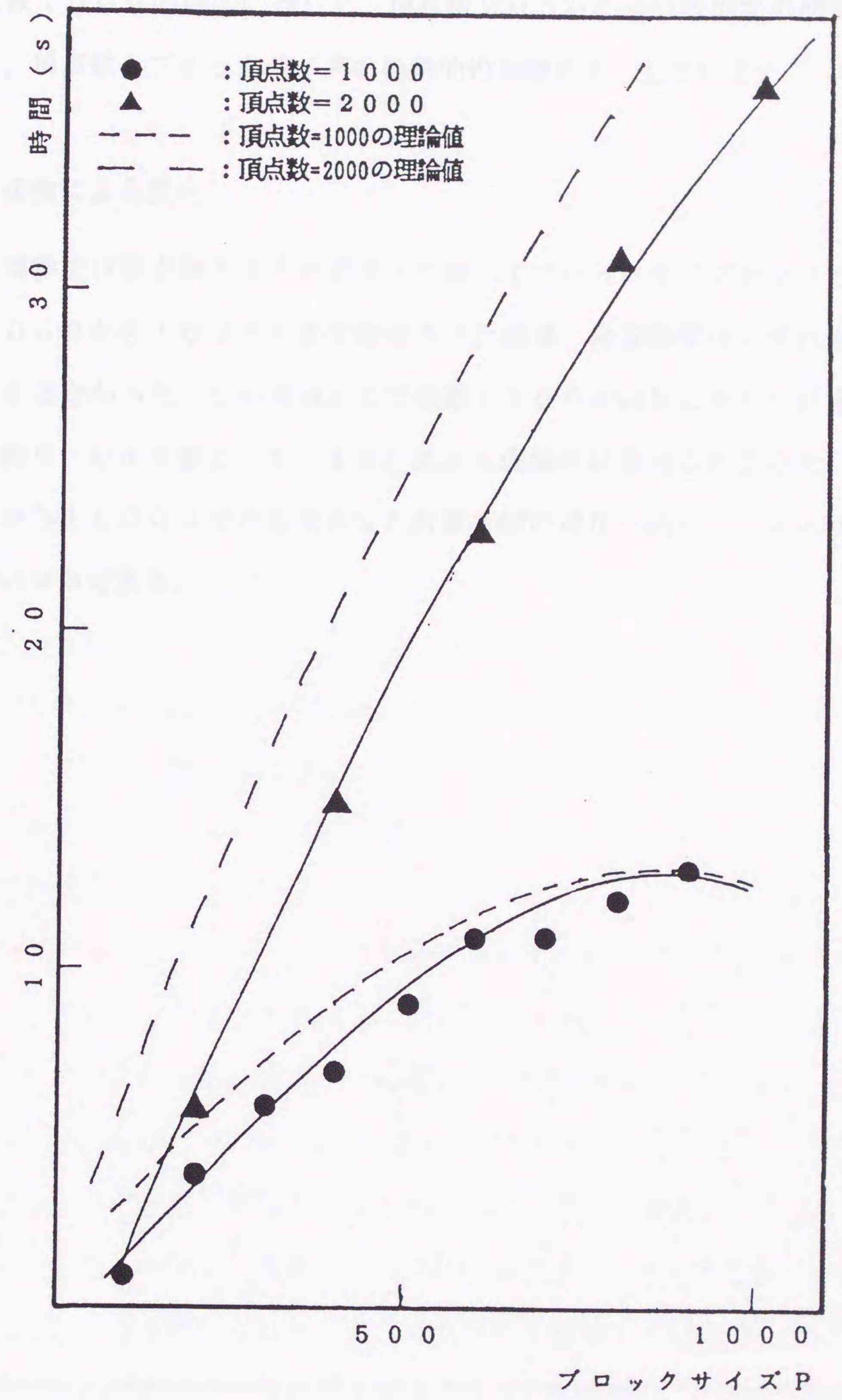
（２）　ブロックサイズによる変化

図５．５に頂点数が１０００と２０００の２つの場合についてブロックサイズの変化が計算時間に与える影響を示す。ただし辺数は頂点数の２倍である。ここで実線は実測された計算時間を示し、破線は（５．１３）式から計算された理論値を示す。この結果、頂点数が１０００の場合から、ブロックサイズ（$\lambda n+\nu$）／$2\beta \fallingdotseq$８８８のとき最大値になる上に凸な曲線となることがわかる。



ブロックサイズ＝２００
変数　＝　２×頂点数

図５．４　頂点数と時間計算量



辺数＝２×頂点数

図５．５ ブロックサイズと時間計算量

また頂点数１０００の場合に対して、頂点数２０００の場合は増加の傾斜が大きくなり、頂点数とブロックサイズの相乗的増加傾向を示している。

（３）　辺数による変化

今回の実験では頂点数５００のグラフに対してブロックサイズを２０とし、辺数を１０００から１００００まで変化させた結果、計算時間はいずれも３秒と変化を示さなかった。この理由として辺数１０００の増加に対して計算時間の増加は約０．０１６秒と（５．１３）式から理論的に算出されるので、辺数１０００から１００００での変化させた計算時間の増加は約０．１４４秒にしかならないからである。

5．2 分枝限定法による一列化グラフの最適系列分割問題に対する解法

第３章で述べたように、一列化グラフの最適系列分割問題に対してKernighan[2]は分割をブレーク・ポイントを用いて表現し、標準的な動的計画法（ＤＰ）の考えにもとずいて、計算過程で必要とされる増分コストの計算時間をも含めて、全体の計算量がグラフの辺数に比例する算法を提示した。また浅野[1]は区間の集合上で定義される問題を考察し、区間木によるデータ構造によって動的計画法を効率的に実行する手法を示した。その中の一例として本問題を扱い、同じ計算量の算法を開発した。

動的計画法の最適性の原理と、分枝限定法における優越関係には大きな共通点があることが知られている[6]。分枝限定法で下界値テストは行わず、探索法として順位を適当に選んだ横型探索法を用い、優越関係による限定操作のみを用いるとき、この分枝限定法は動的計画法の手順と等価になる。したがってこの立場では動的計画法と分枝限定法との間には本質的な差はないといってよく、動的計画法の計算手順は分枝限定法の一種とみなせる。しかし動的計画法をこのように分枝限定法とみなすと、必ずしも最良の探索法とはいえない横型探索法を用いていること、および下界値等による限定操作を全く用いていないという点からさらに改善の余地がある。この視点から標準的な動的計画法の上に組み立てられたKernighanの算法も改善可能と考えられる。

この章では、まず最適系列分割問題の動的計画法による構成を、探索法の選択や部分解の限定操作をより柔軟に行える分枝限定法によって再構成する。その構成にはKohler[3]の論文を参考にし、本問題の特殊性にあわせて分枝限定法の構成要素である探索法、分枝規則、優越関係、下界値関数、上界値、削除規則、停止則を考察し効率のよい算法の構成を目指す。探索法として、計算終了までに分枝される部分解の個数を最小にすることが期待できる最良下界探索法

を用い、限定操作として優越関係テスト、下界値テストに加えて、この問題に固有な特徴を利用した細分化禁止則を取り込んだ分枝限定法による構成を行う。この際、細分化禁止則は分枝規則の中に取り込み分枝数の絞り込みを行い、生成される部分解の大幅な減少と、それにともなう増分コスト計算の減少を計っている。これを最良下界探索構成法とする。また探索法を、順位としてレベル値を用いた横型探索法に換えた構成についても述べる。この構成は下界値テストと細分化禁止則の利用をやめたとき、Kernighan の動的計画法による構成と同等なものになる。さらに、離散的動的計画法の計算量の改善に分枝限定法の考えがどのように利用できるかについて述べた Morin,Marsten[4]の論文の考えにしたがって、これら二つの規則を直接 Kernighan の算法の動的計画法による構成部分に反映させることも可能である。これをレベル順横型探索構成法とする。

さらに、以上の構成による算法の数値実験にもとづいて、予期された算法特性の実証と性能評価を行う。また算法の理論的計算量の推定についても考察する。優越テストはプログラム上では、開リストまたは閉リスト中の同レベルな部分解との比較によって行われる。このとき閉リストによる限定操作が最も有効に働くのは最良下界探索構成法の場合であり、実験によればほぼ３０％の棄却率が得られている。一方、レベル順横形探索構成法ではこの閉リストによる限定操作は全く機能しない。細分化禁止則による削除効果は探索法の選択によらず、どちらの構成に対しても等しく有効であり、約２０％から４５％の棄却率を得ている。算法の性能評価のために、数値実験から決定できる指標として、計算の複雑度を表わす総分枝数、子部分解の同レベル判定による平均棄却率、下界値テストによる棄却数、Ｕカット数および開リスト長を表として提示し、プログラム挙動の理解と分析のためのデータとした。下界値テストはレベル順横形探索構成法では有効に働きプログラムの停止を早める。これに反して最良下界探索構成法ではその効果は期待できない。最良下界探索構成法で停止を早

めるには確定性の利用が有効であり、停止までの反復回数がｎ＋１の約６０から９０％に短縮できることが実証できた。プログラムが停止するまでの総分枝回数については最良下界探索構成法の方が有利である。最後に両構成法による算法の理論的計算量について考察する。

５．２．１ 分枝限定法に関する記法と諸定義

まず、本章で使用する諸定義を以下にまとめる。 部分解を単にその部分系列π＝｛b_1，b_2，・・・，b_m｝，b_m＜ｎ＋１によって表し、最終ブレーク・ポイントb_mをＢＰ（π）で表す。またＢＰ（π）の値を部分解πのレベル値といい、最終ブレーク・ポイントの等しい部分解同志を同レベルな部分解という。π´＝｛b_1，b_2，・・・，b_m、b_{m+1}｝のことをπにブレーク・ポイントb_{m+1}を付加することにより、πより分枝して得られる部分解と言う。πをπ´の親部分解、π´をπの子部分解と言う。部分解のコストは常にそれから分枝により派生されるすべての完成解の目的関数値の下界値を与えている。

また、ある時点において、分枝の対象となる部分解を活性化された部分解と呼び、活性化された部分解の集合を開リストに格納する。同じ時点において、すでに分枝されたか、終端に達した部分解を非活性化された部分解と呼び、非活性化された部分解の集合を閉リストに格納する。

５．２．２ 最良下界探索構成法による分枝限定法の構成

本章では標準的な動的計画法の考えのもとに組み立てられたKernighanの算法に対して、分枝限定法の考え方を導入し改善を試みる。そのとき、最適系列分割問題を戦略の多様性と問題の特性に容易に対応できる柔軟性をもつ分枝限定法で構成する。その概要は、探索法として最良下界探索法、限定操作としては

下界値テストと最適性の原理に対応する優越関係テスト、また分枝操作の中に本問題の特殊性を利用した細分化禁止則を取り込む構成であり、これを最良下界探索構成法という。算法の構成にあたってはKohlerの構成法を参考に、探索法、分枝規則、優越関係、下界値関数、上界値、削除規則、停止則の各要素ごとに考察を進める。

なお、分枝限定法の一般的構成は図５．６に示しておく。

（１）探索法

あらゆる探索法の中で計算終了までに分枝される部分解の個数を最小にする特徴を持つ最良下界探索法を採用する。最良下界探索法は優先順位付き待ち行列のキー値を下界値とすることによって構成される。これによって開リスト中の下界値最小の部分解を取り出し、分枝を行った後、その部分解自身を閉リストに移す。

ある許容部分解が同レベルの部分解の中での最小コストのものになることを部分解が確定されたと言う。探索法に最良下界探索法を用いるこの構成のもとでは開リストの先頭から取り出された部分解は常に確定される性質が以下の定理より導かれる。

定理５．１

閉リストを持ち最良下界探索法を用いた本算法では閉リスト中に存在する節点は常に確定した部分解である。

証明：

これまでに閉リストに入れられた節点の最終ブレーク・ポイントの集合をＶＬとする。開リスト中の各節点はＶＬに含まれる点のみをを中間ブレーク・ポイントとする。開リスト中の各節点の下限値はその節点に達するこのような部

```
(1)   procedure Branch and Bound
(2)   begin
(3)       π_cur := 初期部分解;   U := Maxval;      { U は上界値 }
(4)       repeat
(5)           while ( "π_cur より分枝規則に従って次の子部分解π_son を生成する" )
(6)             begin
(7)               "π_son の下界値を次の式で求める。"
                      lbd(π_son) = lbd(π_cur) + C(BP(π_cur),BP(π_son));
(8)               if ( "π_son と同レベルな部分解π_mk が開リストあるいは閉リスト
                          に含まれている" )
(9)                 if ( lbd(π_mk) > lbd(π_son) )
(10)                    if ( "π_mk が開リストに含まれている" )
(11)                      "開リスト中のπ_mk を π_son で置き換える。";
(12)                   else
(13)                     "π_mk を閉リストから削除し、かわりに π_son を開リス
                          トに挿入する。";
(14)                else
(15)                  "何もしないで、π_son を捨てる。";
(16)            else
(17)              "π_son を開リストへの挿入対象とする。";
(18)            if ( lbd(π_son) > U )
(19)              "挿入対象の π_son を捨てる。";
(20)            else if ( BP(π_son) == n+1 && lbd(π_son) < U )
(21)              begin
(22)                U := lbd(π_son);  π_opt := π_son;
(23)                "開リストから lbd 値が U より大なるものをすべて捨てる。";
(24)              end;
(25)            "挿入対象である π_son を開リストに入れる。";
(26)          end;
(27)      "現在のπ_cur を閉リストに入れる。";
(28)      "開リストの key 値が最小の要素を次のπ_cur とし、その要素を開リストか
          ら取り除く。";
(29)      until (開リスト = 空);
(30)      if (U <> Maxval)
(31)        "π_opt が最適解である。";
(32)      else
(33)        "解は存在しない。";
(34)  end;
```

図５．６　分枝限定法のアルゴリズム

分解の中で最小コストを持つもののコスト値が書き込まれている。これは算法の構成から明かである。

開リストの中から下限値が最小の節点ＣｕｒＮｏｄｅを選びだしたときに、この節点に対応する部分解ＣｕｒＳｏｌの最終ブレーク・ポイントをb_pとすると、ＣｕｒＳｏｌはb_pを最終ブレーク・ポイントとする他のすべての部分解の中で最小コストをもつことを示すことができる。このことから最終ブレーク・ポイントがＶＬに属する節点は常に確定した部分解に対応することがわかる。

上の事実をＶＬのサイズに関する帰納法によって証明する。｜ＶＬ｜＝０に対する正当性は自明である。次に集合ＶＬによって特徴ずけられたある段階で上記命題が正しく成立しているものと仮定する。ここで算法は開リストの中から下限値が最小の節点としてＣｕｒＮｏｄｅを選びだしたとする。このＣｕｒＮｏｄｅの最終ブレーク・ポイントをb_pとすれば、これは次にＶＬに含めるべき最終ブレーク・ポイントである。節点ＣｕｒＮｏｄｅに対応する部分解ＣｕｒＳｏｌは、その最終ブレーク・ポイントb_pと同じ最終ブレーク・ポイントをもつ他の全ての部分解の中で最小コストを与えるものであることを背理法によって証明する。

上の部分解ＣｕｒＳｏｌのコスト値より小さなコスト値をもつ他の許容部分解が存在すると仮定する。この部分解を$\{b_1, b_2, \cdots, b_m = b_p\}$とすると、$b_1, b_2, \cdots, b_{m-1}$の中に集合ＶＬに属さないブレーク・ポイントが少なくとも一つは存在しなければならない。このような点の中で最も番号の小さいものをb_kとすると、つぎの不等式が成立する。

$$\begin{aligned} &Cost(\{b_1, b_2, \cdots, b_k\}) \\ &\quad \leqq Cost(\{b_1, b_2, \cdots, b_k, \cdots, b_m\}) \\ &\quad < Cost(CurSol) \end{aligned} \tag{5.14}$$

b_{k-1}はＶＬに属するブレーク・ポイントであるから、b_{k-1}を最終ブレーク・

ポイントとする確定した節点が存在する。それをＰｒｅＮｏｄｅとすると、

$$\begin{aligned} &\mathrm{PreNode.lbd}+\mathrm{IncCost}([b_{k-1}, b_k)) \\ &\leqq \mathrm{Cost}(\{b_1, b_2, \ldots, b_{k-1}\}) \\ &\quad +\mathrm{IncCost}([b_{k-1}, b_k)) \\ &= \mathrm{Cost}(\{b_1, b_2, \ldots, b_k\}) \end{aligned} \tag{5.15}$$

となり、上の二つの不等式から

$$\begin{aligned} &\mathrm{PreNode.lbd}+\mathrm{IncCost}([b_{k-1}, b_k)) \\ &< \mathrm{Cost}(\mathrm{CurSol}) \end{aligned} \tag{5.16}$$

を得る。この不等式は、節点ＰｒｅＮｏｄｅに対応する部分解をＰｒｅＳｏｌとするとき、算法は次の確定節点の選択にあたって、ＣｕｒＳｏｌではなくて算法の計算過程でより下限値が改善された部分解$\mathrm{PreSol}+[b_{k-1}, b_k)$に対応する節点の子孫のうちの一つを選ぶであろうことを述べている。これは明らかに矛盾であり、主張の正当性が示される。

□

以上より閉リストに移された部分解が再び開リストに戻らず、許容部分解はコスト値の大きさの順に一つずつ確定していくことによって、解を得るまでの反復回数の減少がもたらされる。

（２）分枝規則

分枝規則により選ばれた部分解の集合を候補者集合と言う。本問題の特性を利用し、分枝による絞り込みを行い、限定操作の一部を候補者集合の縮小で実現する。

本問題に対する分枝規則は各ブロックの頂点の重みの和が B 以下であることを保証する容量制約則と、次に述べる細分化禁止則を用いて構成する。

ここで$d(y)=\max\{x|\sum_{i=y}^{x-1} w_i \leq B\}$とする。与えられた単調増大な任意のブレーク・ポイントの部分列

$$b_i,\ b_{i+1},\ \cdots,\ b_j \tag{5.17}$$

に対して［b_i、b_j）が容量制約を満たすとき、すなわち$b_j \leqq d(b_i)$であるとき、細分点b_{i+1}、・・・、b_{j-1}を設けないことによって、明らかに、目的関数の値を改善できるか、また少なくとも改悪することはない。よって最適解を求める立場では$b_j \leqq d(b_i)$をみたすブレーク・ポイントの列(5.17)による細分化された分割を考慮する必要はない。これを細分化禁止則という。

すでに細分化禁止則をみたしている部分解$\pi_i=\{b_1,\ b_2,\ \cdots,\ b_i\}$に$b_{i+1}$を付加し、部分解$\pi_{i+1}$を派生するとき、容量制約則と細分化禁止則の性質をみたすように$b_{i+1}=BP(\pi_{i+1})$を決めるには次の不等式を分枝規則として採用すればよい。

$$BP(\pi_{i+1}) \leqq d(BP(\pi_i))$$

かつ

$$BP(\pi_{i+1}) > d(BP(\pi_{i-1})) \tag{5.18}$$

ここで、π_{i-1}はπ_iの親部分解とする。

とくに、境界条件として、$BP(\pi_i)$が1のときは

$$d(BP(\pi_i)) \geqq BP(\pi_{i+1}) \tag{5.19}$$

とし、$d(BP(\pi_i)) \geqq n+1$ ならば

$$BP(\pi_{i+1})=n+1 \tag{5.20}$$

とする。

（3）優越関係

優越関係は部分解の集合で定義された２項関係で、２つの部分解の優劣を比

較する。部分解πとπ´において、それぞれを親とする可能解の集合の中でコストが最小なものを比較し、２つの部分解の優劣性を決定する。πから派生するすべての可能解の中で最小のコスト値がπ´から派生される同様なコスト値より低い値を示すなら、πはπ´に対して優性であり、逆に後者は前者に対して劣性であると言う。劣性のπ´からは最適解は分枝されないので削除可能である。これを優越テストと呼ぶ。

本問題では同レベルにおける優越関係が成立する。同レベルにおける優越関係とは、同レベルな任意の２つの部分解をπ，ρとした場合、Cost（π）＜Cost（ρ）ならば、どちらの同レベルな部分解π，ρにおいても以後派生するブレーク・ポイントのパターンは等しいので、コストの単調増加性よりπはρに対して優性であることが成り立つ。

この同レベルにおける優越関係により各レベルでの最良な値を決定できる。これは動的計画法における最適性の原理に対応するものである。

（4）下界値関数と上界値

与えられた部分解πに対してπから派生するすべての完成解からなる集合をＳとする。任意の部分解πに対して下界値決定関数ｌ（π）は

$$0 \leqq l(\pi) \leqq Min\{cost(\pi') - cost(\pi) \mid \pi' \in S\} \tag{5.21}$$

を満たすものとする。

$$lbd(\pi) = cost(\pi) + l(\pi) \tag{5.22}$$

は親πから派生されたすべての部分解のコストの下界値をあたえる。このとき下界値テストは上界値Ｕを用いて

$$cost(\pi) + l(\pi) \geqq U \tag{5.23}$$

をみたすπを削除するものである[4]。上界値Ｕはすべての可能解の目的関数の

上界であり、現在知られている最良値すなわち暫定値をとる。

なおｃｏｓｔ（π）の計算は増分コスト関数 $C_f(y, x)$ を用いて、式（３．２）により再帰的に計算できる。（３．２）式中の増分コスト計算は、その定義式である（３．１）式を直接用いる方法に対して、以下の Kernighan の漸化式を採用することによって、本問題のコストの全計算量はグラフの辺の数に比例する[2]。

$$C_f(y,x) = C_f(y\text{-}1,x) + (c(y\text{-}1,x) + c(y\text{-}1,y+1) + \ldots + c(y\text{-}1,n))$$
$$- (c(x,y\text{-}1) + c(x+1,y\text{-}1) + \ldots + c(y\text{-}2,y\text{-}1)) \qquad (5.24)$$

（５）削除規則

削除規則は優越関係や下界値テストなどを使って、候補者の削除、現在の活性化部分解、および非活性化部分解の中から不用のものを除去する操作を開リスト、閉リストに関する操作として記述したものである。

確定性を利用すると削除規則は次のように分枝限定法の一般的構成に対して簡略化でき、それによって、オーバーヘッドの減少が見込まれる。

R1: πの同レベルな部分解が開リストまたは閉リスト中に見いだされないなら、πを新規要素として開リストに挿入する。

R2: πの同レベルな部分解が閉リスト中に見いだされるなら、πは棄却する。

R3: πの同レベルな部分解が開リスト中に見いだされ、lbd（π）が開リスト中の同レベルな部分解の下界値よりも改善されているとき、πによる更新を行う。改善されていないとき、πは棄却する。

（６）停止則

分枝限定法では通常開リストが空になったとき、算法を停止し、この時点で

上界値が初期設定から改善されていれば最適解であると判明する。この方式による算法の停止を正規停止則と言う。これに対して最良下界探索構成法では正規停止則以前に最終ブレーク・ポイントがｎ＋１の可能完成解に達したときに停止する特殊化された停止則を採用し、停止過程を早める。

５．２．３　レベル順横型探索構成法による分枝限定法の構成

最良下界探索法に対し、探索法をレベル順による横型探索法に換えた構成を考える。これは動的計画法に対応するもので、レベル順横型探索構成法という。この構成での探索法は、次の分枝を行う対象として開リスト中の最小レベル値をもつ部分解を取り出すことによって、レベル順による横型探索法を用いる。

分枝規則、優越関係、下界値関数は前記の最良下界探索構成法で使用したのと同様な構成要素を用い、細分化禁止則を初めとして、同レベルにおける優越関係および増分コストの漸化計算式を算法構成にそのまま取り込む。探索法としてはレベル順に横型探索法を構築するために、優先順位付き待ち行列のキー値を部分解のレベル値に設定する。また下界値テストを積極的に利用することによって、正規停止則のもとで解を得るまでの反復回数がｎ＋１以下にできる。削除規則を構成するにあたって、ある対象部分解πから分枝された任意の部分解はＢＰ（π）より大きなレベル値をもち、閉リストにより同レベル要素が見いだされることがない性質に着目すると、以下のような規則の使用が効果的である。対象とする子部分解ρに対して

RL1: ρの同レベルな部分解が開リスト中に見いだされないなら、ρを新規要素として開リストに挿入する。

RL2: ρの同レベルな部分解が開リスト中に見いだされ、lbd（ρ）が開リスト中の同レベルな部分解の下界値よりも改善されているなら、ρによる更新を

行う。改善されていないなら、ρは棄却する。

RL3: 部分解が可能解に到達したとき、下界値の値が上界値Uより改善されていれば、上界値を改善した値に変更し、この上界値より大なる下界値をもつ開リスト中の部分解をすべて削除する。これをUカットと言う。Uカットとは別に通常の下界値テストによる限定操作は常に行う。

すでに知られているように、横型探索法のもとで削除規則として優越関係を用いれば、最適性の原理と同じ効果が実現され、動的計画法の手順と等価になる[4],[6]。これを、算法の第（m＋1）段の開始直前の閉リストと開リストの内容を詳細に示すことにより明らかにする。

閉リストは

$$(i_1^{(1)},\cdots,i_{m-B+1}^{(m-B+1)},\cdots,i_{m-1}^{(m-1)})$$

一方、開リストは

$$(i_m^{(m)},i_{m+1}^{(m)},\cdots,i_{m+B-1}^{(m)})$$

$$i_m^{(m)} = \min_{lbd}\{j_m^{(m-B)},j_m^{(m-B+1)},\cdots,j_m^{(m-1)}\}$$

$$i_{m+1}^{(m)} = \min_{lbd}\{\quad\quad j_{m+1}^{(m-B+1)},\cdots,j_{m+1}^{(m-1)}\}$$

$$\cdots$$

$$i_{m+B-1}^{(m)} = \min_{lbd}\{\quad\quad\quad\quad j_{m+B-1}^{(m-1)}\}$$

となる。ここで上付の添字は算法の反復回数を、下付の添字はその節点のレベル数を表す。またｍｉｎ$_{lbd}$｛・・・｝は下限値最小の節点を選ぶことを意味し、その各要素の下限値は次の式で計算される。

$$j_k^{(m)}.lbd = i_m^{(m)}.lbd + c_{m,k}$$

第m段で決まる開リストの要素$i_m^{(m)}$は算法の次の展開の対象となり閉リストへ

確定され移動する節点である。上述の$i_m^{(m)}$の決定方法はKernighanのＤＰの漸化式と一致することを示すことができる。ただし上に示した下界値テストと細分化禁止則による限定の効果によって、この構成ではさらに動的計画法の計算効率を高めている。

５．２．４ 高速化のためのデータ構造

おおまかに以下の５つの点においてアルゴリズムおよびデータ構造の工夫をほどこし高速化に対処した。

（１）グラフデータの表現

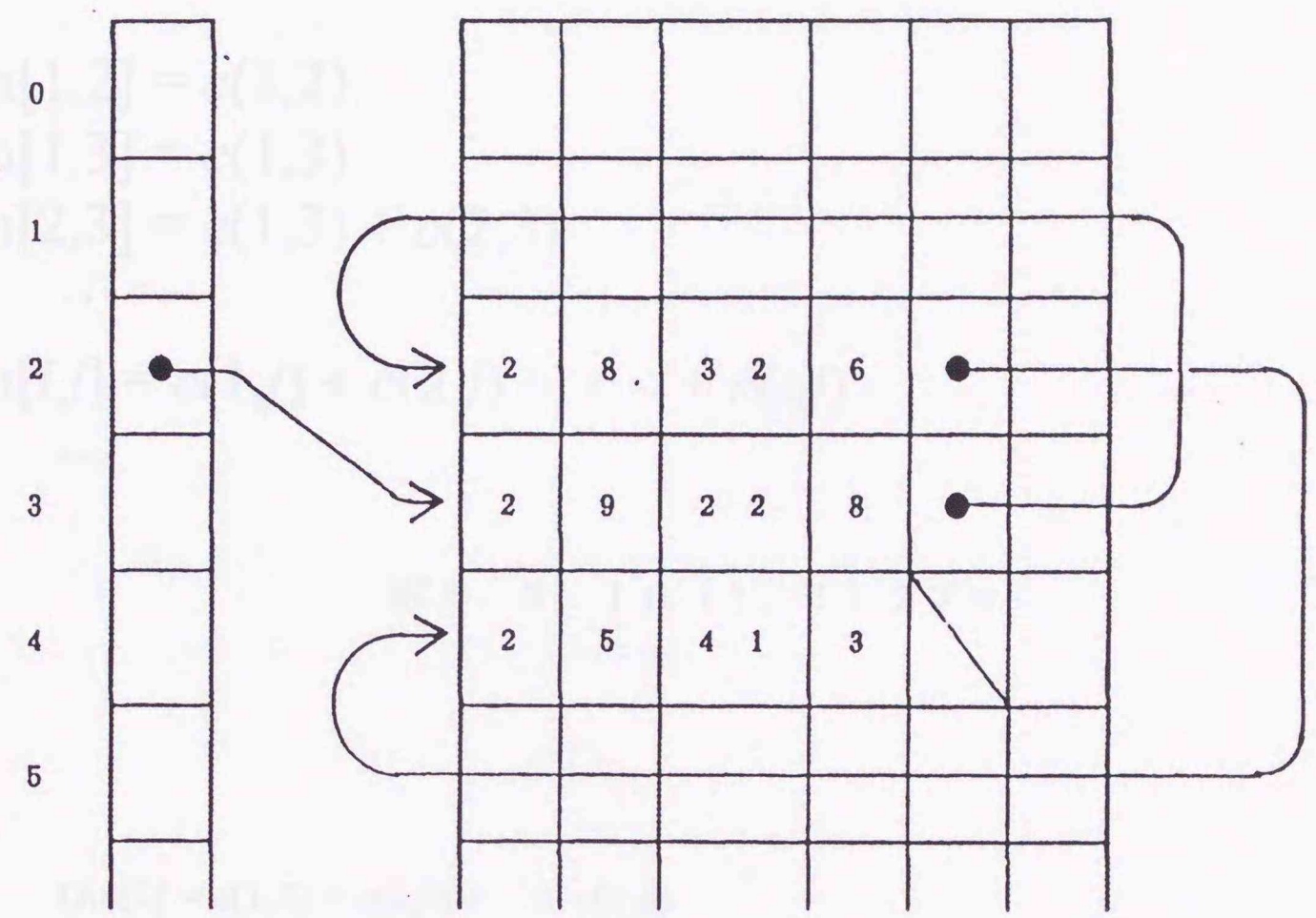
任意のノードから距離が遠い順に整列された流出辺をリストにつなぎ、同様に任意のノードへ距離が遠い順に整列された流入辺をリストにつなぐ。（図５．７参照）これによって、増分コスト計算におけるスパース部への非参照による効率化が行われる。

（２）増分コスト計算

これまでの増分コスト計算はその定義式である$C_f(y,x)=\sum_{\substack{y\le i<x\\ j\ge x}} c(i,j)$を直接用いる方法と以下の（５．２）式で示された漸化式

$$C_f(y,x)=C_f(y,x\text{-}1)+(c(x\text{-}1,x)+c(x\text{-}1,x\text{+}1)+\ldots+c(x\text{-}1,n)$$
$$-(c(y,x\text{-}1)+c(y\text{+}1,x\text{-}1)+\ldots+c(x\text{-}2,x\text{-}1)$$

を直接用いる方式によってプログラムを構成していた。これに対して、今回は次に述べる２つの補助データ表を用意することによって積極的に効率化の改善を計る。１つめの補助データ表として、任意の頂点を始点とするエッジのコストの総和を以下のＯｕｔ配列に格納したものを用いる。



org : 辺の始点

dst : 辺の終点

cst : 辺のコスト

diff : 辺の長さ

back : 始点を共有する次の流出辺へのつなぎ

prio : 始点を共有する次の流入辺へのつなぎ

図５．７ グラフデータの表現

In[1,2]	In[1,3]	In[1,4]	…
	In[2,3]	In[2,4]	…
		In[3,4]	…
			…

In[1,2] = c(1,2)
In[1,3] = c(1,3)
In[2,3] = c(1,3) + c(2,3)
…
In[i,j] = c(1,j) + c(2,j) + … + c(i,j)
…

図５．８　Ｉｎ［ｉ，ｊ］タブロー

Out[1] = c(1,2) + c(1,3) + ... + c(1,n)

Out[2] = c(2,3) + c(2,4) + ... + c(2,n)

…

Out[i] = c(i,i+1) + c(i,i+2) + ... + c(i,n)

…

Out[n-1] = c(n-1,n)

第２の補助データ表として、１からｉの範囲の頂点を始点とし、頂点ｊを終点とするエッジのコストの総和をＩｎ［ｉ，ｊ］に格納した表をつくる。図５．８にこれによって構成されたタブローを示す。

この補助データ表を用いると、コスト計算を示す先の（５．２）式は

C_f(y,x) = Cf(y,x-1) ＋頂点ｘ－１を始点とするエッジコストの総和

－（頂点ｙから頂点x-2の範囲を始点とし、頂点x-1を終点とする
エッジのコストの総和）

$=C_f$(y,x-1) ＋Ｏｕｔ［ｘ－１］

－（Ｉｎ［ｘ－２，ｘ－１］－Ｉｎ［ｙ－１，ｘ－１］）

となる。これによって、エッジの参照部分を補助段階に分解でき、複数の同一参照を避けることができる。

（３）探索空間のノード状態の表現

ＢＢで展開する探索空間におけるノード状態の表現はブレイクポイントの後ろの方から逆順リストで表現する。ブレイクポイントが｛１、３、７、９｝である部分解を図５．９に示す。これによって、探索ノードのブレイクポイントのデータへのアクセスの迅速化が計れる。

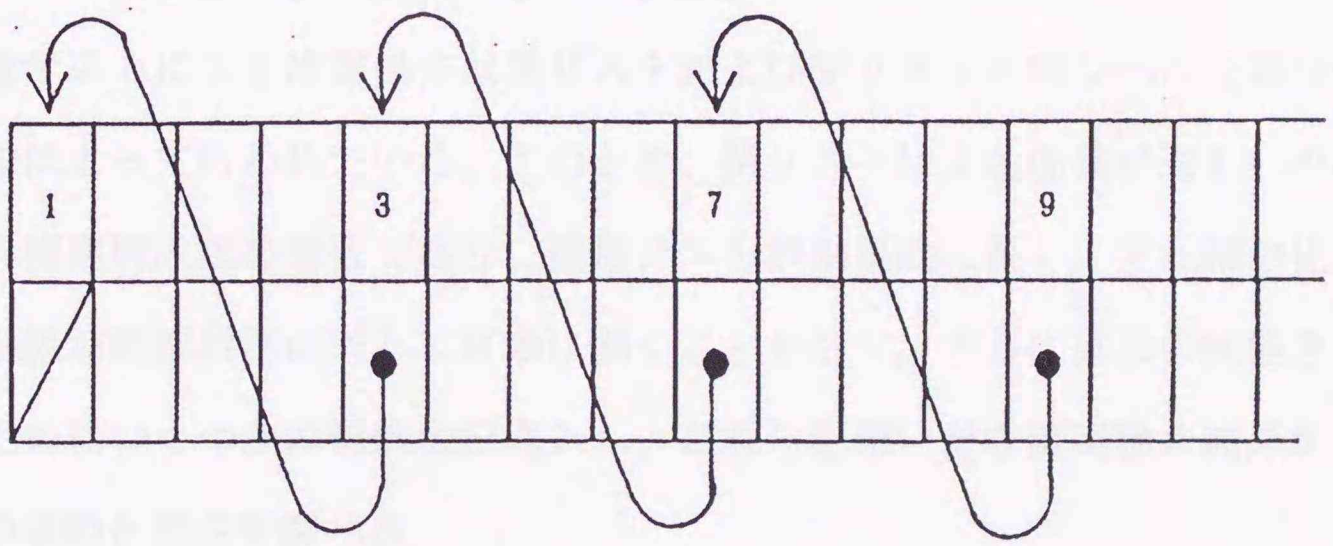


図５．９ 探索空間のノード状態の表現

（4）開リストと閉リスト処理の高速化

開リストおよび閉リストを管理するためのデータ構造を図５．１０で表わす。展開されたノードを開リストに格納するために、、キー値の低い順にリストを構成させる。ｌｉｎｋ部には探索ノードの状態へのポインター、最良下界探索構成法でのキー値である下限値はｌｂｄ部に格納する。このとき、双方向整列リストを構成することにより項目の出し入れの高速化が計られる。また同レベル要素を高速に参照するためにＸ付表を作成し、開リストと閉リストの判別のためにＦｌａｇ付表をおく。

５．２．５ 数値実験による性能評価

本章では以上の構成によって得られた算法の数値実験にもとずいて、算法の特性と性能評価を行う。また算法の理論的計算量についても論じる。なお、数値実験では上界値の初期値は十分大きな値とする。

優越テストによる限定操作は開リストおよび閉リストの同レベルな部分解との比較によって行われている。このとき、閉リストによる効果が著しいのは最良下界探索構成法の場合であり、優越テストが効果的に働く。また細分化禁止則の効果が両探索法に対して有効に働くことを示す。さらに算法の性能を評価するためにいくつかの特性数を導入し、これらを用いた数値実験の結果から、算法の挙動と特性を調べる。

（1） 探索法の選択による削除効果

レベル順横型探索構成法の場合、レベル値の大きさの順に解が順次決定していくため、閉リスト中に同レベル要素を見いだすことがないので、閉リストの使用が限定操作に影響を及ぼすことはない。したがって、この構成では閉リス

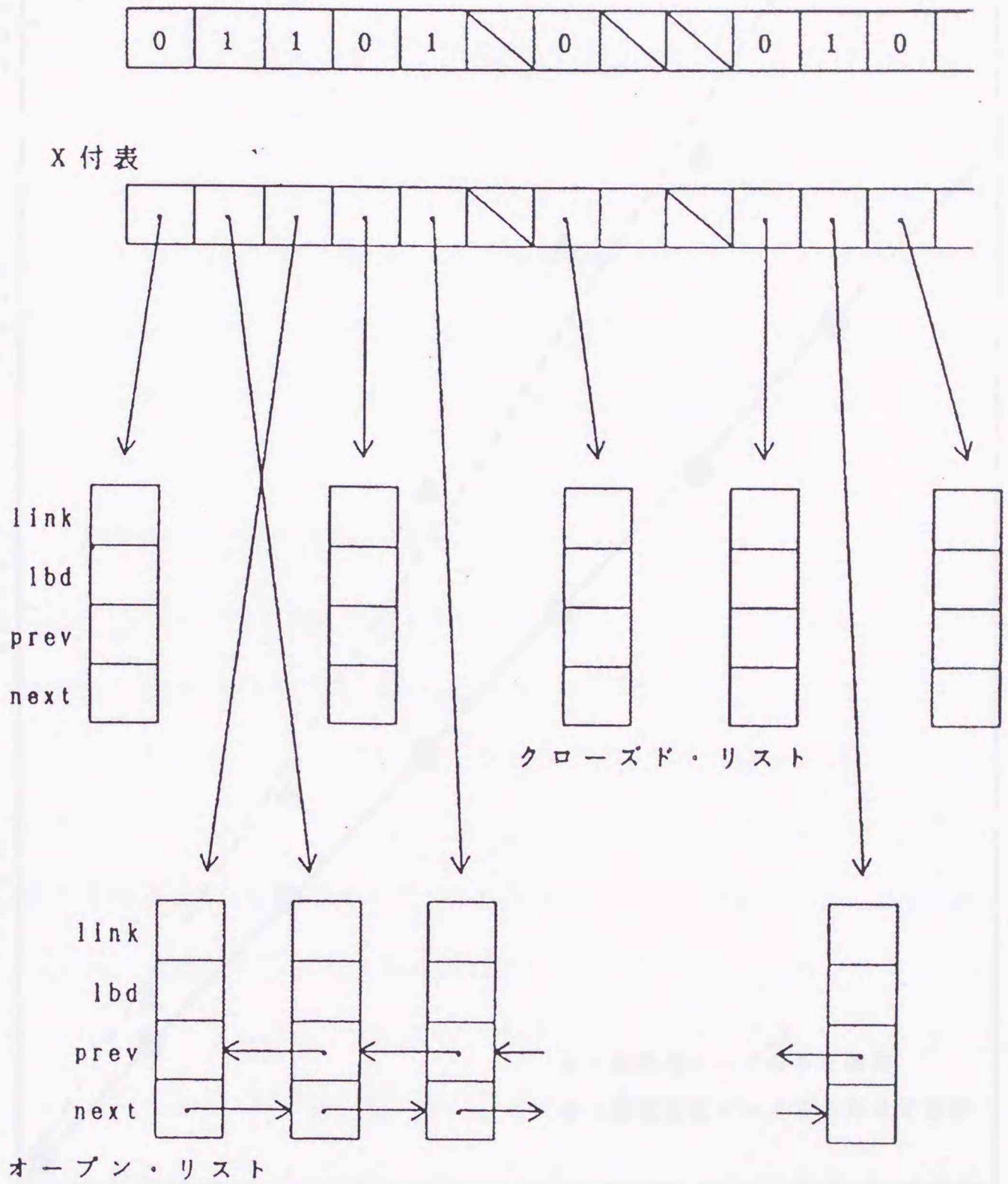


図5．１０ 開リストと閉リストを管理するデータ構造

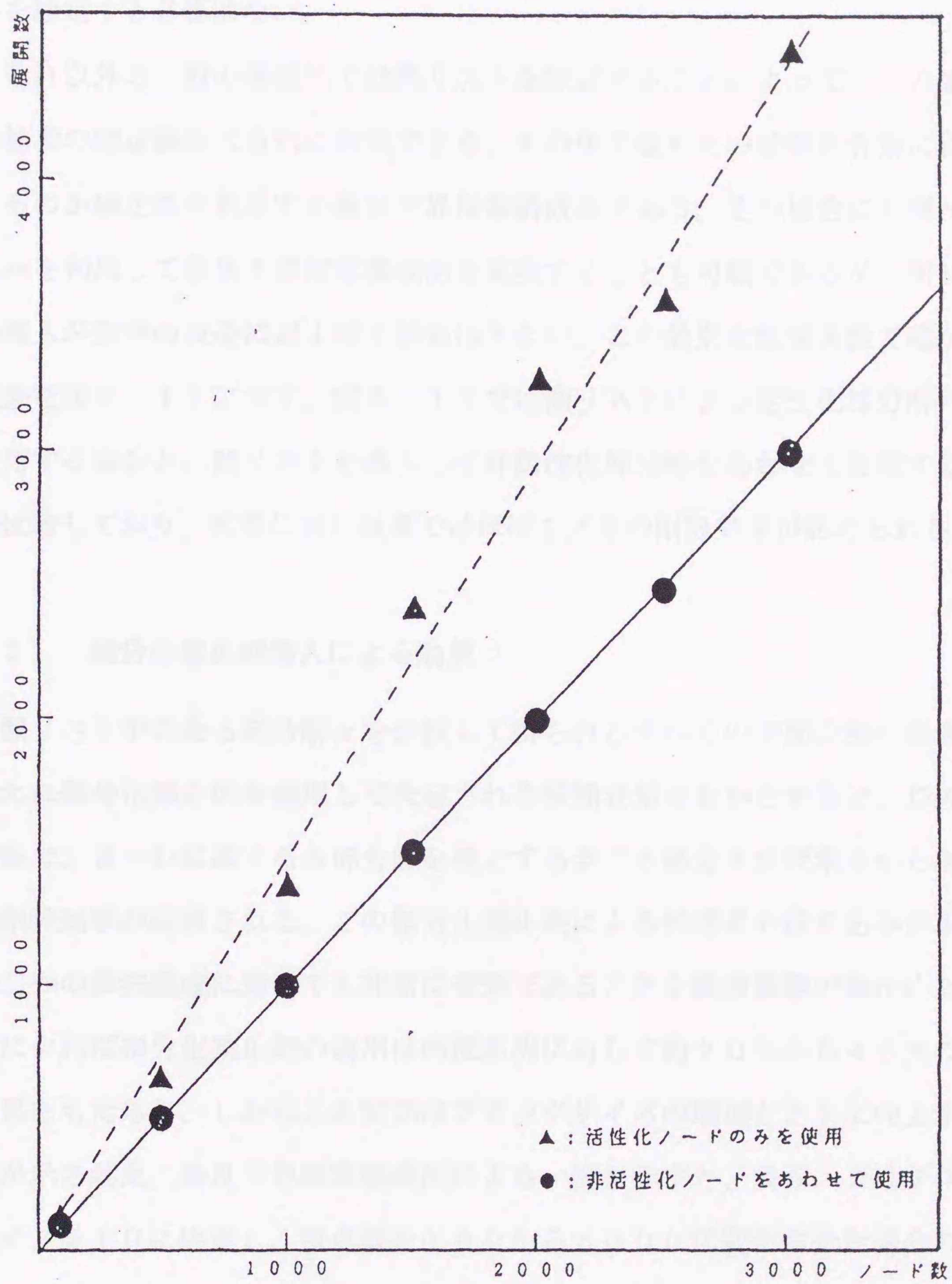


図5．11 頂点数と展開数

トを設定する必要はない。

それ以外の一般の構成法では閉リストを設定することによって、その情報を候補者の限定操作に有利に利用できる。その中で最もその情報を有効に利用できるのが確定性を利用する最良下界探索構成法である。その場合にも開リストのみを利用して最良下界探索構成法を実現することも可能であるが、閉リストの導入が効率の改善におよぼす効果は大きい。この効果を数値実験で確かめた結果を図５．１１に示す。図５．１１では開リストにより活性化部分解のみを使用する場合と、閉リストを導入して非活性化部分解をあわせて使用する場合を比較しており、前者に対し後者ではほぼ１／３の削除効果が認められる。

（２）　細分化禁止則導入による効果

開リスト中のある部分解πを分枝して得られるすべての子部分解の集合をＢ、それに細分化禁止則を適用して決定される候補者集合をＤとすると、Ｄの生成過程で、Ｂ－Ｄに属する各部分解を根とする全ての部分木が探索木から削除され削除効率が改善される。この細分化禁止則による候補者の絞り込みが本問題の二つの算法構成に対しても非常に有効であることを数値実験で確かめた。それによれば細分化禁止則の適用は両探索法に対して約２０％から４５％の削除効果をもたらし、しかもこの効果はブロックサイズの増加とともに向上することが示された。最良下界探索構成法による一例を示すと、表５．３はブロックサイズを１０に固定し、頂点数を１００から３０００に変化させた場合の発生した子部分解数とそのうち細分化禁止則により削除された子部分解数を示している。表５．４は頂点数１００のグラフでブロックサイズを変化させた場合の同様な事項について表示している。

表５．３ 頂点数に対しての細分化禁止則の評価（ブロックサイズ＝１０）

頂点数	100	500	1000	1500	2000	2500	3000
子部分解の発生数	858	4769	9750	14748	19678	24717	29596
細分化禁止則による削除数	258	1412	2644	4017	5326	6716	7830

表５．４ ブロックサイズに対しての細分化禁止則の評価（頂点数＝１００）

ブロックサイズ	4	8	12	16	20	24	28
子部分解の発生数	385	693	1027	1333	1518	1581	1792
細分化禁止則による削除数	86	182	324	555	690	692	885

（３） 動作特性と評価

分枝限定法の枠組みの中で各算法特性を比較検討するための指標として以下の特性値を用いる。

計算の複雑度として解を得るまでの反復回数を用いる。すなわちこの値は算法の開始から開リストが空になるまでに開リストから取り出された分枝対象となる部分解の総数である。

子部分解棄却率を、算法の全過程で分枝規則により親部分解から分枝された子部分解の総数に対する開または閉リストによる削除規則（最良下界探索構成法では§５．２．２の（５）の規則R2,R3またレベル順横型探索構成法では§５．２．３のRL2）によって棄却された子部分解の総数の比率として定義する。

また下界値関数による削除効果を下界値テストによって棄却する回数とUカット数で表す。下界値テストによる棄却数は分枝された子部分解を暫定値である上界値によって棄却する回数である。Uカット数は探索木の葉に達したときに変更された上界値との比較によって開リスト中の部分解を削除する数である。

表５．５ レベル順横型探索構成法による数値実験結果
（頂点数＝１００、 正規停止則使用）

ブロックサイズ	反復回数	子部分解棄却率 (%)	下界値テストによる棄却数	Uカット数
4	99	37	1	2
8	95	60	3	2
12	95	63	3	6
16	88	65	5	3
20	86	65	15	4
24	91	76	0	4

表５．６ 最良下界探索構成法による数値実験
（頂点数＝１００、 特殊化された停止則使用）

ブロックサイズ	待ち行列長			反復回数	子部分解棄却率(%)
	min	max	mean		
4	4	9	4.65	93	49
8	8	16	9.05	83	65
12	12	29	12.92	71	69
16	16	26	16.65	69	67
20	20	32	20.59	59	69
24	24	43	24.38	58	79

これらは§５．２．３のRL3の規則に対応する。

これらの特性数を数値実験で計測した一例を表５．５、表５．６に示す。数値実験から判明した結果について以下にまとめる。なお、この数値実験では下界値決定関数はすべてのπに対してｌ（π）＝０としている。

レベル順横型探索構成法では開リスト長は定常的にはブロックサイズと一致し、Uカットが起きた時点から急速な減少を示す。最初のUカットが生ずるの

は第（ｎ＋１）－Ｂ段においてであり、それ以前に完成解に達することはない。Ｕの更新にともない、この段以後初めて下界値テストは有効となり、最適解への到達を早める。実際、正規停止則のもとで、下界値テストとその先取りであるＵカットの作用でｎ＋１段より早い停止が起こることを表５．５が示してる。

一方、最良下界探索構成法では開リスト長は反復回数ごとに変化するが、その傾向を理論的に推定するのは困難である。しかし多くの数値実験によれば、ほとんどの反復回数でブロックサイズの長さをとり、それよりきわだって長いものの出現は稀であり、平均長はブロックサイズ長をやや上まわる程度であることが判明した。

算法構成から分かるように、部分解は下界値の大きさの順に一つづつ確定していく。それゆえ、最悪でもｎ＋１回の反復ですべての部分解の確定が行われる。実際には最終ブレーク・ポイントがｎ＋１である部分解はそれ以前に確定されている。したがって正規停止則を最終ブレーク・ポイントがｎ＋１である部分解に達したときに停止するように変更すれば、より効率が改善できる。これを実験的に確かめた結果、停止するまでの反復回数はｎ＋１に対して６０～９０％の値であり、（ｎ＋１）－Ｂより小さな数となる。

最良下界探索構成法は反復回数は（ｎ＋１）－Ｂより小でレベル順横型探索構成法ではこれより大となるので、前者が有利である。また最良下界探索構成法の場合、優越関係による削除効率はレベル順横型探索構成法に対してやや上回った数値を与えている。この意味で分枝限定法の構成の枠組みの中では最良下界探索構成法はレベル順横型探索構成法より優位にあると考えられる。実際、数値実験においてもこの傾向は確かめられている。両者の反復回数が頂点数以下となるのに対してKernighanの動的計画法ではｎ＋１回の反復回数が必要である。

また下界値による限定操作は最良下界探索構成法では最終ブレーク・ポイン

トがｎ＋１である部分解に達したとき、すなわちはじめて完成解に達したときに停止するため働かない。しかしレベル順横型探索構成法の数値実験では１（π）の値を０としたにもかかわらず、下界値テスト、Ｕカットによって効果的な限定操作が行われている。したがって１（π）の値を緩和問題などを利用して合理的な値とすることによって、大きなＢに対しては効果が期待できる。

５．２．６　分枝限定法の計算量

レベル順横型探索構成法と最良下界探索構成法について、漸近的計算量を求め、それが優先順位付き待ち行列に依存することを示す。ただし本問題の応用上の立場から、疎（辺数＝Ｏ（ｎ））であることが保証されている場合に計算量を導く。

両探索法共通の計算量に寄与する主な要因は増分コスト計算と優先順位付き待ち行列に関する計算量である。このうち増分コストの計算量は式(3.2)を用いることによって、計算終了までに必要な総計算量はＢが一定のときＯ（｜Ｅ｜）となり[2]、疎なグラフではＯ（ｎ）となる。次に、優先順位付き待ち行列に関する総計算量は解が得られるまでに、この優先順位付き待ち行列に対する基本操作を反復する回数と、操作に要する計算量との積となる。反復回数は両構成法とも高々ｎ＋１回である。一回の反復において、優先順位付き待ち行列から要素の取り出しは一回、また挿入、削除、更新の操作は高々ブロックサイズ分行われる。優先順位付き待ち行列にヒープ構造を用いることにすれば上記の各基本操作に要する計算量はＯ（log（行列長））である。以上をまとめると優先順位付き待ち行列に関する総計算量はＯ（（ｎ＋１）×（１＋ブロックサイズ）log ｎ）≒Ｏ（ｎ log ｎ）となる。したがって、総計算量はＯ（ｎ log ｎ）と考えられる。

この導出は優先順位付き待ち行列長が最大ｎに達することを考慮して導いて

いるが、レベル順横型探索構成法では優先順位付き待ち行列長はブロックサイズを越えない。これよりBが一定のとき計算量はO（n logB）＝O（n）となる。漸近的計算量はKernighanと同じ結果になる。

一方、最良下界探索構成法では前に述べたように行列長は平均としてややブロックサイズを上回る程度で長いものの出現頻度はきわめて少ない。この数値実験の結果から、計算時間は実用上O（n）と考えてよい。また優先順位付き待ち行列にVan Emde-Boas priority queues[5]を用いると、最小値の取り出し、要素の値の挿入、書換操作はO（log log N）で行える。ここでNはキー値の上界である。Nとして下界値集合の上界を取り、現問題でコスト値が有界であるとするとNはnに比例すると考えて良いので計算量は理論上O（n log log n）に改善できる。

５．３ まとめ

本章では、提案する無閉路有向グラフの最適系列分割問題の解法の重要な構成部品となる一列化グラフの最適系列分割問題の特性を明らかにし、さらに、解法の性能を改善することにより、提案する問題への寄与を試みた。

まずKernighanのＤＰアルゴリズムの計算量について詳細な検討を行ない、この研究で不足している評価検討を補った。計算量はアルゴリズムより理論的に導きだしたものと、具象化したプログラムより導きだした２つを示した。以上の２つにおいてはほぼ同様な計算量となる。これらよりKernighanが述べるところの「計算量が辺数に比例する」以外に、頂点数に比例し、ブロックサイズについて上に凸な増加曲線を示し、およびこれらの相乗効果からなることがわかる。これらの傾向は数値実験でも確かめられた。

次に、KernighanのＤＰアルゴリズムに対して探索法および限定操作の観点から改善の余地が見出されるという観点から、分枝限定法の考え方のもとで、効

率的算法を提案した。探索法に最良下界探索法、分枝規則に細分化禁止則を導入した最良下界探索構成法と、細分化禁止則と下界値による限定操作を導入した動的計画法に相応するレベル順横型探索構成法の特性と性能について詳細な検討を行い、その有効性を確かめ、その比較検討を行なった。

特に、反復回数に関しては最良下界探索構成法では（n＋1）－B 以下であり、レベル順横形探索構成法ではそれ以上になる。下界値による限定操作は最良下界探索構成法では不要であり、レベル順横形探索構成法では（n＋1）－B 段以上で初めて有効に機能する。したがってブロック長 B が短いとき、下界値決定関数 l （π）をどのように選んでもその効果には限りがある。一方、細分化禁止則による限定操作は両構成法ともに非常に有効であるが、最良下界探索構成法に対してより良い結果が得られている。

また、計算量に関しては疎（辺数＝O（n））なグラフに対して、レベル順横形探索構成法ではO（n）であり、最良下界探索構成法では開リスト上のデータの統計的性質から、実用上はO（n）と考えてよく、理論上は開リストの優先順位付き待ち行列の構成法に強く依存する。これらの諸点を総合的に見れば最良下界探索構成法がより優れている。

プログラム実装上から見れば、開リストの優先順位付き待ち行列構成はレベル順横形探索構成法に対してはレベル値をインデックスとする一次元配列を用いて容易に実現できるが、最良下界探索構成法では高機能の優先順位付き待ち行列の使用を検討しなければならない。

参考文献

[1] Asano, T. : Dynamic Programming on Intervals, Technical Report, 91-AL-20, Information Processing Society of Japan, pp.1-8(1991).

[2] Kernighan,B.W. : Optimal Sequential Partitions of Graphs, J.ACM, Vol.18, No.1, pp.34-40(1971).

[3] Kohler,W.H.: Characterization and Theoret-ical Comparison of Branch-and-Bound Algorithms for Permutation Problems, J.ACM,Vol.21,No.1,pp.140-156(1974).

[4] Morin, T. L. and Marsten, R. E. : Branch-and-Bound Strategies for Dynamic Programming, Operations Research, Vol.24, No.4, pp.611-627(1976).

[5] Van Emde Boas, P., Kaas, R. and Zijlstra, E. : Design and Implementation of an Efficient Priority Queue, Mathmatical Systems Theory, No.10, pp.99-127 (1977).

[6] 茨木 : 組合せ最適化－分枝限定法を中心として－、産業図書(1983).

[7] 加地、大内：最適系列分割問題の基礎的研究, 北海道情報大学紀要, Vol.1, No.1, pp.115-126(1990).

[8] 加地、大内: 分枝限定法による最適系列分割問題、情報処理学会研究報告、90-AL-16,pp.69-76(1990).

[9] 加地、大内 : ダイナミックプログラミングによる最適系列グラフ分割問題の計算量, 北海道情報大学紀要, 第 2 巻, 第 2 号, pp.25-45(1991).

[10]加地、大内 : 動的計画法による最適系列グラフ分割アルゴリズムの計算量解析, 北海道大学工学部研究報告, 第 157 号, pp.59-70(1991).

[11]加地、大内： 分枝限定法による系列グラフ分割問題の高速化と拡張, 情報処理学会研究報告（アルゴリズム研究会）, Vol.91,No.48 pp.1-8(1991).

[12]加地、大内： 分枝限定法による系列分割問題の算法構成と効率, 情報処理学会研究報告（アルゴリズム研究会）, Vol.92,No.58 pp.25-32(1992).

[13]加地、大内： 最適系列分割問題に対する効率的分枝限定法の構築と諸特性

解析, 情報処理学会論文誌, Vol.35, No.3, pp.364-372(1994).

第６章　無閉路有向グラフの最適系列分割問題に対する効果的厳密解法

本研究で提案する無閉路有向グラフの最適系列分割問題に対する解を与えることを目標に、まず単一の入口と出口を持つ無閉路有向グラフ$D(V,E)$に対して、系列を保持する部分グラフ、頂点集合Vの切断、Vの互いに素な部分集合の切断による分離などの定義と記法を説明し、系列分割の概念を導入し、さらに、任意の系列分割が切断の鎖によって一意に定まることを示した。それによって、単一の入口と出口を持つ無閉路有向グラフの各頂点に重み、各辺にコストを付与したネットワークを構成することによって総カット値を最小化する系列分割問題を考察し、さらに、問題の定式化を容易にするために、分割を一意に表す切断決定子を導入し、切断の鎖を切断決定子列で一意に表現できることを示すことによって問題の定式化を試みた。

本章では、動的計画法を適用し、無閉路有向グラフの構造に並列構造が認められるとき、多項式オーダーの実用時間内で計算可能な有効な厳密解法を提案する。まず、すべての切断を決定するために切断を表す切断決定子を節点とする木を生成する。このための分枝規則を与える基本操作について述べ、基本操作を高速に実現するための関数を導入する。次に無駄のない探索方法を実現するために木を縮約し既約グラフとする。この既約グラフ上で動的計画法にもとづき、この問題の性質を加味した最適化算法を構成する。最後に細分化禁止則による効果を示し、さらに並列に近い構造のグラフに対して、算法の計算量が多項式オーダーになることと、中間エッジの挿入がこの計算量の負担を軽減す

ることについて述べる。

６．１ 厳密解法の概略と諸定義

ここでは、本章で用いる記号等について以下に述べる。　点vを終点（始点）とする辺(u,v)（辺(v,u)）をvの流入辺（流出辺）と呼ぶ。Vの切断を$\mu=(A^c,A)$とすると、点$v\in A$に対してDでのvの入り次数を$d^-(v)$で、A^cに始点をもつvへの流入辺の個数を$d^-_\mu(v)$で表す。$d^-_\mu(v)>0$を満たす点vをAの境界点といい、その集合を$\partial A=\{v|v\in A,d^-_\mu(v)>0\}$とする。また、$A-\partial A$を$A$の内点の集合という。$A$を定義域とする関数$h_\mu(v)=d^-(v)-d^-_\mu(v)$を特性値関数と定義し、その値を$v$の特性値と呼ぶ。その値は上組に始点をもつ$v$への流入辺の個数を表す。半順序集合$(A,\preceq)$の極小元$v$は$h_\mu(v)=0$を満たす点であり、切断決定子は定義より$\gamma=\{v|v\in A,h_\mu(v)=0\}$と書き表される。$A$と$\partial A$および$\gamma$の間には包含関係$\gamma\subseteq\partial A\subseteq A$が成立する。また、境界条件に関する記述の統一性とプログラムの便宜上、入口の前にダミーの頂点を一個加え、切断$\mu_{k+1}=(V,\varnothing)$を表す$\gamma_{k+1}$を切断決定子列の末尾に常に加える。

６．２ 厳密解法の算法構成

本章では前章で論じた無閉路有向グラフの総カット値を最小化する系列分割問題についての効果的厳密解法の構成と実現について論じる。まず、切断の鎖の生成過程を可能解に至る部分解の列挙のプロセスと考えて、すべての切断からなる構造木を構成し、これより同一の切断決定子が重複しない縮約した既約グラフを導出する。この上で問題固有の性質を取り入れた動的計画法による算法について論じる。

６．２．１　切断決定子の生成

切断$\mu=(A^c,A)$に対応する切断決定子をγとする。このとき、下組A^cの頂点数をμまたはγのレベルと呼ぶ。前後関係 $\prec=$ についてμより後にあり、かつレベルが１だけ大であるような切断をμ'とする。μ'は$(A,\preceq)$の極小元の一つを選び、それを下組A^cに移すことによって得られる。すなわち、$v\in\gamma$をピボットとして選び、

$$A'=A-\{v\} \tag{6.1}$$

と置けば、

$$\mu'=((A')^c,A') \tag{6.2}$$

となる。この操作をvをピボットとする切断μに対する基本操作という。$(A,\preceq)$において、ピボットvを始点とするすべての辺の終点の集合をWとする。すなわち、$v\prec u$であるが$v\prec x\prec u$をみたすxが存在しないことを$v\ll u$で表すと、Wは次のように書ける。

$$W=\{u|u\in A, v\ll u\} \tag{6.3}$$

基本操作を行うと関数$h_\mu()$は定義域をA'とする関数$h_{\mu'}()$として新たに定義され、

$u\in A'-W$のとき、$h_{\mu'}(u)=h_\mu(u)$

$u\in W$のとき、$h_{\mu'}(u)=h_\mu(u)-1$ (6.4)

となる。

補題６．１　切断μのレベルが１以上ならば、次式が成り立つ。

$$\gamma=\{w|w\in\partial A, h_\mu(w)=0\}$$

証明：　$\partial A\subseteq A$より、$\gamma=\{w|w\in A, h_\mu(w)=0\}\supseteq\{w|w\in\partial A, h_\mu(w)=0\}$である。逆に$v\in\gamma$とすれば$h_\mu(v)=0$であり、また$D$が単一の入口を持ち、$\mu$のレベルが１以上であることから、$d^-(v)>0$が成立し、$d^-_\mu(v)=d^-(v)>0$となる。これから、$v\in\partial A$が導かれ、逆の包含関係が満たされる。

□

この補題と、関係$\gamma \subseteq \partial A \subseteq A$から、$(A,\preceq)$および$(\partial A,\preceq)$の極小元集合は$\gamma$と一致する。次に基本操作により変換された境界点の集合と切断決定子は以下の２つの定理によって導かれる。

定理６．１　vをピボットとする切断μに対する基本操作により、μがμ'に変換されるとき、∂Aは$\partial A'=(\partial A-\{v\})\cup W$に変換される。

証明：　まず、$\partial A' \supseteq (\partial A-\{v\})\cup W$を示す。右辺を次のように互いに素な部分集合に分ける。

$$(\partial A-\{v\})\cup W=((\partial A-\{v\})-W)\cup W$$

$u\in(\partial A-\{v\})-W$のとき、基本操作でピボットvを下組に移しても、点uへの下組からの流入辺の個数は変わらない、それゆえ$d^-_{\mu'}(u)=d^-_{\mu}(u)>0$となる。また、$u\in W$のとき$u$への流入辺として$(v,u)$が常に存在するので、$d^-_{\mu'}(u)\geq 1>0$となる。いずれにしても$u$は$\partial A'$に含まれる。

逆に、$\partial A' \subseteq (\partial A-\{v\})\cup W$を示す。“$u\in\partial A'$に対して、$u\notin(\partial A-\{v\})\cup W$”と仮定すると、この仮定は“$u\in A-\{v\}$かつ$d^-_{\mu'}(u)>0$に対して、$u\notin(\partial A-\{v\})$かつ$u\notin W$”と書き直される。$u\in A-\{v\}$と$u\notin(\partial A-\{v\})$から$u\in A-\partial A$となり、$d^-_{\mu}(u)=0$が導かれる。また$u\notin W$であるから、$d^-_{\mu'}(u)=d^-_{\mu}(u)=0$であり、$d^-_{\mu'}(u)>0$に矛盾する。

□

この定理６．１と上記の（６．４）式から、vをピボットとする基本操作によりμがμ'に変換されるとき、$h_\mu()$の定義域をAから∂Aに縮小して得られる制限関数$h_\mu(u)|\partial A$は次の（６．５）式で与えられる$h_{\mu'}(u)|\partial A'$に変換されることを

容易に示すことができる。

$u\in(\partial A-\{v\})-W$に対して、$h_{\mu'}(u)|\partial A'=h_{\mu}(u)|\partial A$

$u\in W$のときは

$u\in\partial A$ならば、$h_{\mu'}(u)|\partial A'=h_{\mu}(u)|\partial A-1$

$u\in A-\partial A$ならば、$h_{\mu'}(u)|\partial A'=d^{-}(u)-1$ (6.5)

ここで、μの上組からの流入辺の個数が1であるWの点の集合W^{*}を

$$W^{*}=\{u|u\in W,h_{\mu}(u)=1\} \quad (6.6)$$

と定義すると、次の定理が成立する。

定理６．２． ｖをピボットとする切断μに対する基本操作によりμがμ'に変換されるとき、γがγ'に変換されるならば、

$$\begin{aligned}\gamma'&=\{u|u\in A',h_{\mu'}(u)=0\}\\&=\{u|u\in\partial A',h_{\mu'}(u)=0\} \quad (6.7)\\&=(\gamma-\{v\})\cup W^{*}\end{aligned}$$

となる。

証明： 定理６．１で与えられた式の右辺を次のように互いに素な集合に分ける。

$$\begin{aligned}\partial A'&=(\partial A-\{v\})\cup W\\&=((\partial A-\{v\})-W)\cup W\end{aligned}$$

ここで、関数$h_{\mu}()$の変換式（６．４）と補題６．１を用いると次の計算式が成り立つ。

$$\begin{aligned}&\{u|u\in\partial A',h_{\mu'}(u)=0\}\\&=\{u|u\in(\partial A-\{v\})-W,h_{\mu'}(u)=0\}\cup\{u|u\in W,h_{\mu'}(u)=0\}\\&=\{u|u\in(\partial A-\{v\})-W,h_{\mu}(u)=0\}\cup\{u|u\in W,h_{\mu}(u)=1\}\\&=(\gamma-\{v\})\cup W^{*}\end{aligned}$$

□

（6．7）式より、∂Aに制限した情報とそれに対応する関数$h_\mu()$を表す情報のみを持つことによって、γ'はγから、vからの出次数に線形な計算負担で計算可能である。また∂Aをもつことによりコスト計算も効果的に行うことが可能になる。

次に、すべての切断を決定するために切断を節点とする木を生成する。切断μに対応する切断決定子γの点vをピボットとして基本操作を行うと、vの選択に応じて、1レベル高い切断$\mu_1,\mu_2,\ldots,\mu_r$が作り出される。これらを親節点μを展開して得られる子節点と考える。親から子への展開操作を繰り返すと高さ$n+1$の多分岐平衡木Tが生成できる。ただし、nは頂点数である。図6．2に図6．1のグラフから作られた木Tを示す。ここでは切断のかわりに切断決定子によって節点を表している。この木の葉節点はレベル値がnの切断$(V,\varnothing)$を表しており、また各節点にその節点を得るために用いたピボットをラベルとして付けると、根から葉に至る経路上の節点のピボットの列は頂点集合Vのある特定の一列化に対応する。

6．2．2　既約グラフの生成

上で述べた切断を節点とする木Tでは、同じ切断が多数重複して生成される。しかし、同一の切断を根とする部分木はすべて同じ構造を持つので、これらを重ねて一つの節点とし、既約グラフとして表現できる。図6．3に図6．2に対する既約グラフを示す。既約グラフは、辺の向きが基本操作における親から子への向きであり、入口と出口がそれぞれ1個で、 レベル間隔1のレベル構造を持つ無閉路有向グラフである。

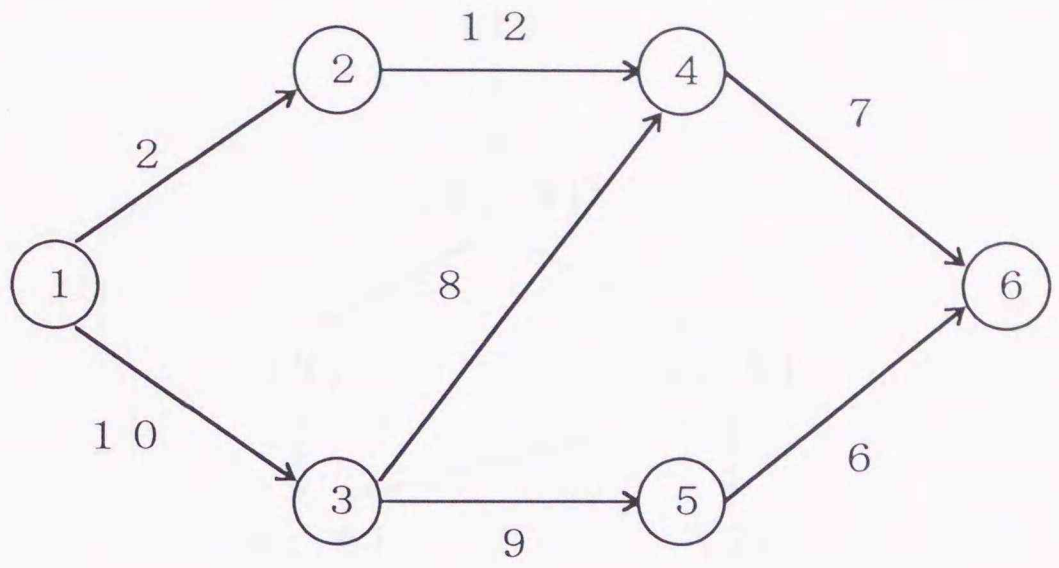


図６．１ 無閉路有向グラフ

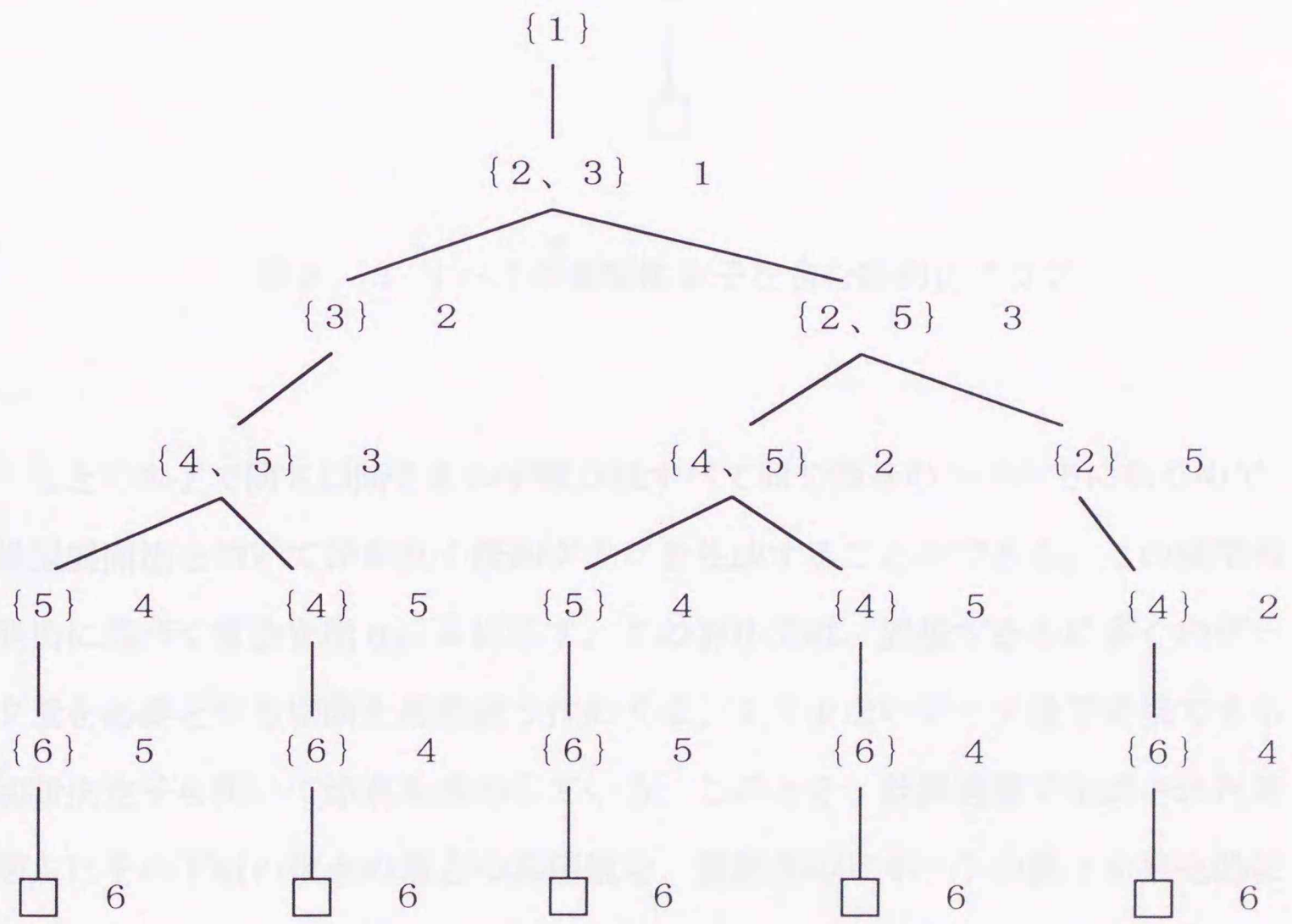


図６．２ すべての切断決定子を生成する多分岐平衡木

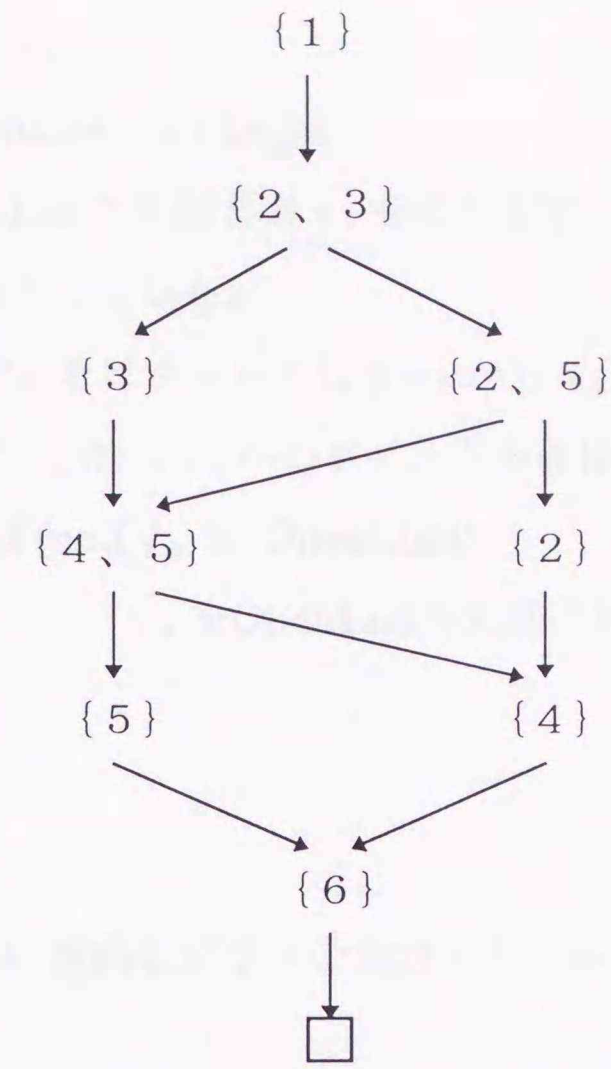


図６．３ すべての切断決定子を含む既約化グラフ

もとの木Tで同じ切断を表わす節点はすべて同じ深さのレベル上にあるので、横型展開法を用いて効率良く既約グラフを生成することができる。この横型展開法に基づく算法を図６．４に示す。この算法では、表現するのに多くのデータ量を必要とする切断を直接使う代わりに、より少ないデータ量で表現できる切断決定子を用いて節点を表わしている。このとき、計算過程で生成された各節点にその下組の頂点の重さの累積値を、親節点のピボットの重さを漸化的に加算していくことによって求め、それを記憶しておく。また横型展開の過程を制御するために、レベル値をキーとする優先順位待ち行列をオープンリスト OpenList として構成する。この算法の第５行 "α をピボットとしてγ_tからγ_gを生成する" 、第７行の"$\gamma_g \in$ OpenList を判断する"という二つの操作を効率良く実

```
1:  OpenList:=γ_s;
2:  While (OpenList≠φ) begin
3:      "OpenListの先頭要素γ_tを取り出す";
4:      for (α∈γ_t) begin
5:          "αをピボットとしてγ_tからγ_gを生成する";
6:          "γ_gからγ_tへのポインタを確保する";
7:          if (not(γ_g∈ OpenList))
8:              "γ_gをOpenListの末尾に加える";
9:      end
10: end;
```

図６．４ 既約化グラフを生成するアルゴリズム

行することはプログラム実装上重要である。前者による計算負担の総量は前章の議論より無閉路有向グラフのエッジ数に比例する。すなわち、切断決定子γは補題６．１からわかるように、特性値関数ｈ$_\mu$()の値が∂Ａ上で既知であれば決定できる。そこで、ｈ$_\mu$()の定義域をＡから∂Ａに縮小した制限関数 ｈ$_\mu$｜∂Ａを導入し、それを各節点μへ付与情報として与えておく。切断μに対応する節点を展開するとき、基本演算にともなう定義域の変換は定理６．１の変換式により、また制限関数ｈ$_\mu$()｜∂Ａの変換は（６．４）式から導かれる（６．５）の計算式を用いて効率的に行うことができ、また各節点に対応する切断決定子γは必要に応じていつでもｈ$_\mu$()｜∂Ａから高速に計算できる。このようにすることによって節点を展開し子節点を生成する計算を高速化し、また必要な記憶容量の減少を計る。

後者のOpenList の判断については、次のようなOpenList 上での節点の分布の性質を利用して効率的な判断が可能である。横型展開の構成法より、OpenList 上

で各活性化節点はレベル値について常に昇順に並び、現在のγ_tから生成されたγ_gと同レベルの切断決定子はOpenListの末尾に連続的にまとまって存在する性質を利用すると、$\gamma_g \in$ OpenList の判断はOpenList を末尾から見ていくことによって効率的に行うことができる。さらに、この性質を積極的に利用してより効率のよい判断も可能である。たとえば、ハッシュ法等を用いて計算効率を$O(1)$まで減ずることが可能である。

６．２．３ 既約グラフ上の最適化算法

切断$(\varnothing, V)$をμ_sとおき、切断の単調列$\mu_s \prec= \ldots \prec= \mu_i$ に対応する切断決定子の列$\{\gamma_s, \ldots, \gamma_i\}$を系列分割問題の部分解と考える。最初の要素が$\gamma_s$、最後の要素が$\gamma_i$であるようなすべての可能部分解$\{\gamma_s, \ldots, \gamma_i\}$の集合を$\Xi(\gamma_i)$で表す。ここで、部分解$\{\gamma_s, \ldots, \gamma_i\}$は既約グラフ上で、節点$\gamma_s$を入口とし節点$\gamma_i$を出口とする部分グラフの一つの系列分割を与える。部分解$\{\gamma_s, \ldots, \gamma_j, \ldots, \gamma_i\}$が$\Xi(\gamma_i)$の中での最小コストを実現するならば、部分解$\{\gamma_s, \ldots, \gamma_j\}$が同時に$\Xi(\gamma_j)$の中での最小コストを実現しなければならない。すなわち、最適性の原理が成立する。したがって、$\Xi(\gamma_i)$を状態と考えた動的計画法、もしくは最適性の原理を取り入れた分枝限定法[4],[5],[6]によって効率のよい厳密解法を構築することができる。

図６．５に動的計画法による算法を示す。この算法に従い、順次最良部分解を確定していくことによって最適解が求まる。ここで$f(\gamma_i)$は$\Xi(\gamma_i)$に属する部分解の最小コストを示す関数であり、またΩは既約グラフ上でレベルの小さい順に並べた(同レベル間においては順不同)切断決定子のリストを表現するものであり、既約グラフより容易に取り出せる。このとき、第４行の$\gamma_j(\prec= \gamma_i)$に関する最小値は、既約グラフ上で親節点を辿ることによって得られる節点γ_iから根γ_sに至るすべての経路上で$|[\gamma_j, \gamma_i)| \leq B$により制約された範囲内で計算する。すなわち$\gamma_j$と$\gamma_i$のすでに既知である、それぞれの頂点に付与した“下組の頂点の重さ

1: $f(\gamma_s) = 0$;

2: while ($\Omega \neq \phi$) begin

3: "Ωの先頭要素γ_iを取り出す";

4: $f(\gamma_i) = \min_{\gamma_j} \{f(\gamma_j) + C(\gamma_j, \gamma_i)\}$;

5: "γ_iから最小値を与えるγ_jへのポインターを確保する";

6: end;

図６．５ 動的計画法によるアルゴリズム

の累積値”の差がＢを越えない範囲内で、節点γ_iを出発点として縦型グラフ探索などを用い探索することとなる。プログラム全体の流れでは、既約グラフの切断決定子を表す各頂点に$\{\gamma_s, \ldots, \gamma_i\}$を表す部分解を割付け、根よりレベル順に$f(\gamma_i)$を確定していく。無閉路有向グラフの出口を表す最後の切断決定子γ_{k+1}が確定されたなら、最適解のコストが求り、計算過程で定めたポインターを逆順に辿ることによって最適解の切断決定子列による分割が求まる。

また、増分コストの定義式は（５．１４）式であるが、この定義式を以下の漸化式で代替し、図６．５の第４行の計算過程で漸化的に使用することによって、計算効率を上げることが可能である。ピボットをvとする基本操作により切断決定子γ_jがγ_mに後退したとき、増分コスト$C(\gamma_i, \gamma_m)$は次式のように書ける。

$$C(\gamma_i, \gamma_m) = C(\gamma_i, \gamma_j) + \sum_{t \in [\gamma_m, \gamma_{k+1})} c(v,t) - \sum_{s \in [\gamma_i, \gamma_j)} c(s,v) \tag{6.8}$$

ここで、$\sum_{t \in [\gamma_m, \gamma_{k+1})} c(v,t)$ はピボットｖからの流出辺の総コストであり、$\sum_{s \in [\gamma_i, \gamma_j)} c(s,v)$ は$[\gamma_i, \gamma_j)$内の要素を始点とするピボットvへの流入辺の総コストである。また、$[\gamma_i, \gamma_j)$内の頂点の認識は、無閉路有向グラフの隣接行列の推移的閉包を始めに

一度求めておくことによって容易に実現できる。これによって、（６．８）式または定義式（５．１４）においても効果的な計算が可能となり、その計算の総量はエッジ数に比例する[1]。

また、全カット値を最小化する本問題では、以下に述べるように図６．５の４行目の計算式における最小値を求める過程で制約として、細分化禁止則[7]を組み込むことによって計算効率の向上が可能である。すなわち、部分解を表す単調増大な切断決定子列

$$\gamma_1, \gamma_2, \ldots, \gamma_i, \gamma_j, \gamma_m$$

に対して制約式

$$|[\gamma_i, \gamma_m)| \leq B$$

が満たされるならば、この部分解より、γ_jを無視した

$$\gamma_1, \gamma_2, \ldots, \gamma_i, \gamma_m$$

に対応する部分解のコストが低い。したがって、はじめの分割を考慮する必要はない。すなわち、細分化禁止則が構成される。この規則を取り入れることによって漸近的計算量は等しいが探索空間の実際の絞り込みによって最適化の計算の負担を軽減する算法を構成する。

６．３ 厳密解法の算法の特性評価

一列化にもとづく算法の計算量は、無閉路有向グラフの頂点数をｎ、辺数をｅとすると、一列化されたグラフの系列分割の計算量がO(ｎ＋ｅ)で求まることを考慮し、生成されるすべての一列化グラフの数をａとすると、O(ａ×(ｎ＋ｅ))と考えられる。ただし、ａの値はグラフの構造に大きく依存する値である。本問題はO(ａ×(ｎ＋ｅ))の計算量で必ず解ける問題である。ただし、この場合はすべての可能解を列挙する計算となる。

本章で提案した算法の計算量に寄与する主な要因は既約グラフの生成と、そ

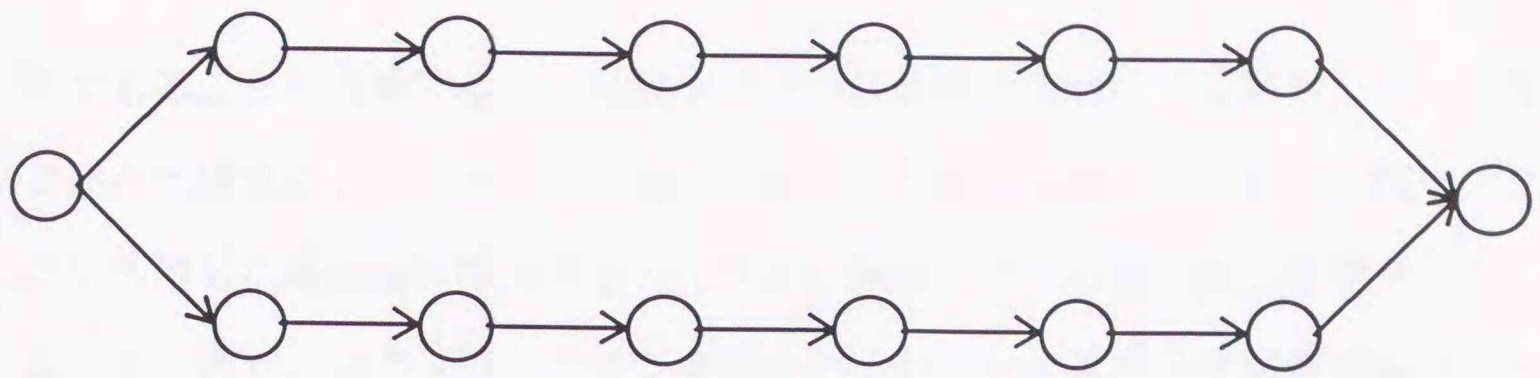


図６．６　２並列なグラフ

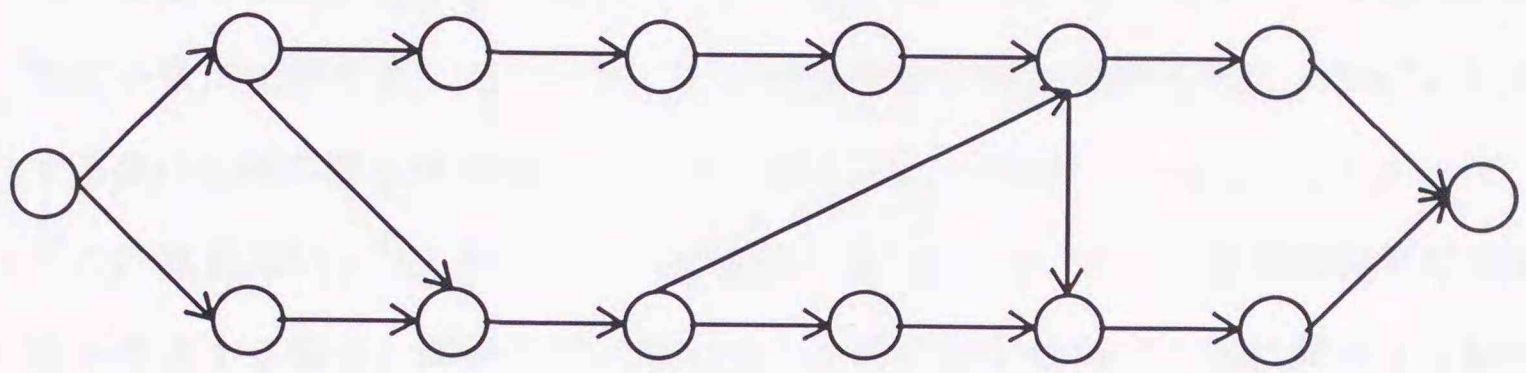


図６．７　中間エッジをもつ２並列なグラフ

のグラフを利用した最適解の導出過程である。両過程における計算量は切断決定子の生成数に大きく依存する。したがって、この切断決定子の生成数に着目し考察する。

ここで着目する切断決定子数は無閉路有向グラフの構造によって大きく影響され、理論的に切断決定子の生成数を求めることは困難である。しかし、無閉路有向グラフをある特定な構造に限定することによって推定は可能である。一例として単一の入口と出口を持つm並列な構造をもち、並列線路間の接続辺（中間エッジ）が存在しないグラフ(図６．６参照)を対象として考察する。一列化にもとづく算法の場合、ａの値は少なくともｍ$^{n/m}$以上となるので指数的増大を示すこととなる。これに対して既約化による場合の考察を以下に示す。

まず、このとき切断決定子の要素数$|\gamma|$は並列数m以下となり、また前章で述べたエッジ数に比例する計算の負担は高々$O(n^2)$で、考えられる切断決定子数以

下であることを考慮すると、既約グラフの生成過程においては第６．２．２章で述べた構成により、その計算量は切断決定子数に比例する。次に、既約グラフを利用した最適解の導出過程では無閉路有向グラフが並列構造を持つことによって、図６．５の４行目の計算過程のある段における最小値探索の数はブロックサイズBのみに依存した定数以下となり、Bが定数であることより同様にその計算量は切断決定子数に比例する。すなわち、本算法の全体の計算量は切断決定子数に比例する。このとき、切断決定子数は組合せ的考察によりn^mに比例する数(2並列の場合は正確に$n^2/4+2$を越えることはない)となる。したがって、その計算量は$O(n^m)$と考えられ、多項式的増大を示す。さらに並列線路間に接続辺が存在する場合、切断決定子数が減少し計算量の負担はさらに減少する傾向が示せる。特に、入口に近い中間エッジは大きな減少をもたらす。図６．７に示すような中間エッジが頂点数の10%の割合でランダムに生成する２並列グラフに対して数値実験を試みた結果、切断決定子数の40から65%の減少が見られた。図６．８に示すように中間エッジが存在しない場合$O(n^2)$となるのに対して上記の多数の数値実験の平均値は$O(n^{1.58})$の特性を示している。

また、細分化禁止則による候補者の絞り込みが最適解の導出過程に対して非常に有効であることを確かめた。それによると細分化禁止則の適用により約10%から30%の削除効果がもたらされ、しかもこの効果はブロックサイズの増加とともに向上する傾向が判明した。ここでは２並列グラフ（中間エッジ数＝0.2×頂点数）に対する一例を表６．１に示す。このときブロックサイズを10に固定し、頂点数を100から400に変化させた場合に対して図６．５の４行目における最小値探索数の総量（全最小値探索数）と、そのうち細分化禁止則により削除された最小値探索数を示している。表６．２は頂点数100の2並列グラフ（中間エッジ数＝0.2×頂点数）でブロックサイズを変化させた場合の同様な事項について表示している。これより、実質的な計算時間の軽減のために細分化禁止則の導入

が薦めるられる。

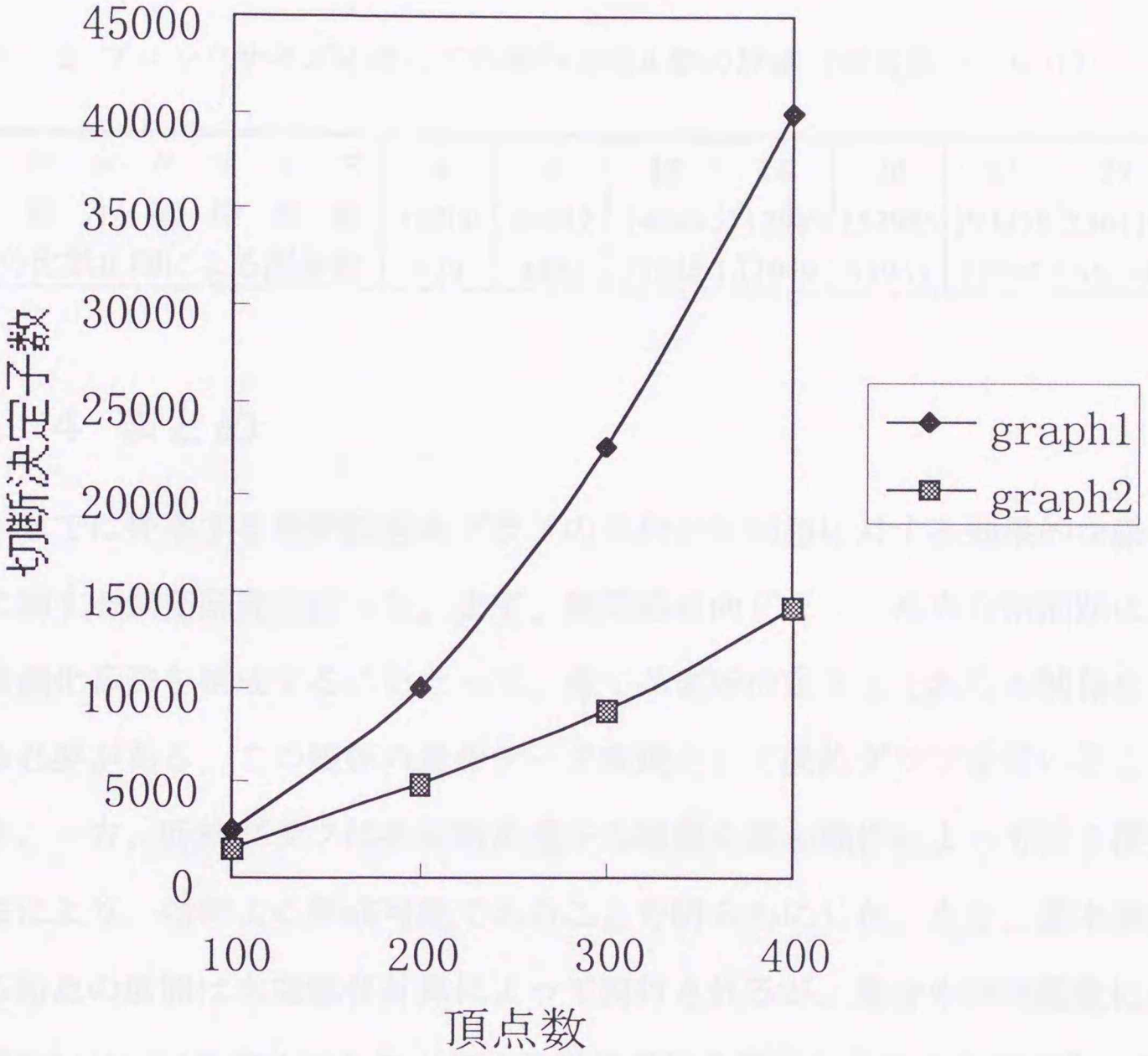


図６．８　頂点数と切断決定子数
graph1：中間エッジがない２並列グラフ
graph2：中間エッジをもつ２並列グラフ

表６．１ 頂点数に対しての細分化禁止則の評価（ブロックサイズ＝１０）

頂点数	100	150	200	250	300	350	400
全最小値探索数	57027	105591	162242	265119	281770	371671	598996
細分化禁止則による削除数	5848	11589	14925	24833	27457	30565	52341

表６．２ ブロックサイズに対しての細分化禁止則の評価（頂点数＝１００）

ブロックサイズ	4	8	12	16	20	24	28
全最小値探索数	14500	40812	74689	112849	152985	194358	236110
細分化禁止則による削除数	624	4431	11648	17949	33963	52796	65231

６．４ まとめ

本章では提案する無閉路有向グラフの系列分割問題に対する効果的な厳密解法に関する開発研究を行った。まず、無閉路有向グラフの系列分割問題に対する最適化算法を構成するにあたって、全ての切断決定子とそれらの関係を表現する必要がある。この関係の最少データ表現として既約グラフを用いることができ、一方、既約グラフは各切断決定子の展開を基本操作によって行う横型展開法により、効率よく形成可能であることを明らかにした。また、基本操作による節点の展開は本来集合計算によって実行されるが、集合の特性関数にあたる関数$h_{\mu}()|\partial A$を導入することにより計算過程を簡素化することが可能である。

要素間に先行順位をもつシステムにおいて並列構造が認められないとき、本来、本問題は指数的計算時間を要するが、これに対して並列構造が顕著に認められるシステム構造に対しては十分実用に耐える算法が実現できることを示した。

参考文献

[1] Kernighan,B.W. : Optimal Sequential Partitions of Graphs, J.ACM, Vol.18, No.1, pp.34-40(1971).

[2] Liu,C.L. : Elements of Discrete Mathematics, McGraw-Hill,Inc(1985).

[3] Moret,B.M.E. and Shapiro, H.D. : Algorithms from P to NP, The Benjamin / Cummings Publishing Company,Inc(1990).

[4] Morin,T.L. and Marsten,R.E. : Branch- and- Bound Strategies for Dynamic Programming, Operations Research, Vol.24, No.4, pp.611-627(1976).

[5] 茨木：組合わせ最適化－分枝限定法を中心として－，産業図書(1983).

[6] 加地，大内：最適系列分割問題に対する効率的分枝限定法の構築と諸特性解析，情報処理学会論文誌，Vol.35, No.3, pp.364-372(1994).

[7] 加地 ： 半順序の最適系列分割問題の構造と算法構成， 商学討究（小樽商科大学）, Vol.45, No.2, pp.185-204(1994).

[8] 加地、大内： 要素間に先行順位をもつシステムの配置問題，電気学会論文誌 C 分冊, Vol.117-C, No.2, pp.136-142(1997).

[9] 加地 ： 要素間に先行順位をもつシステム要素の配置問題に対する厳密解法と近似解法の提案，日本経営工学会論文誌, Vol.47-C, No.6, pp.344-350(1997).

第７章　無閉路有向グラフの系列最適分割問題に対するメタヒューリスティック算法

無閉路有向グラフの系列分割問題に対しての有効な厳密解法を前章で提案した。この厳密解法では並列構造をもつ無閉路有向グラフに対しては実用時間内で計算が可能である。しかし無閉路有向グラフがランダムな構造を有するとき指数的オーダーとなり実質的に計算が困難となる。そこで、ランダムな無閉路有向グラフにも対応できるよう、最近多くの成果を上げているパラダイムであるメタヒューリスティックによる近似解の導出を試みる。

メタヒューリスティックは従来の組合わせ最適化問題を解くための種々の戦略を有機的に結合させたり、あるいは反復させたものであり、従来の近似解法を超えたパラダイムとして注目を浴びている。このとき、各手法においていくつかのパラメータを制御することによって、様々な戦略を構成できる。すなわち、メタヒューリスティックとはヒューリスティックにパラメータを追加し、そこで生まれた自由度を用いて問題を巧く解くテクニックであり、それを工夫することによって、組合わせ最適化問題に対する実用的なツールとなり得る。

ここでは、人間の記憶過程にアナロジーをもつ Tabu Search 法と物理現象の焼き鈍しのアナロジーから構成される Simulated Annealing 法を適用し構築される近似解法の構成について、それぞれ第７．１章と第７．２章で詳細に述べる。さらに、本問題で使用する効果的な近傍構造について一般化し、広くメタヒューリスティック戦略に適合可能となることを第７．３章で示す。

７．１　複合移動による Tabu Search 法

この総カット値を最小化する系列分割問題に対して、前章で示したように動的計画法を適用することによって厳密解法[22]が構成できる。この解法は無閉路有向グラフの構造に強く依存しており、並列構造が認められるとき多項式オーダーであるが、それでも頂点数、および並列数の増加に対して計算、および記憶容量に重い負担がかかることは否めない。また、ランダムな構造を持つ無閉路有向グラフでは指数的オーダーの計算時間を必要とし、この厳密解法では頂点数の多いランダムグラフに対して実用的な時間内での計算は不可能である。これに対して我々は昨今、種々の問題で優れた成果を示しているTabu Search法[7],[8],[9],[16],[23],[24],[25],[22],[28]を用い、本問題の近似解法の構成を試みる。Tabu Search法はFred Gloverによって提案された局所探索法の変形である。その大まかな戦略は探索過程で以前に探索した解に再び戻る解のサイクリングをタブーリストを設けることによって禁止する処置をとることである。

すでに、Tabu Search法により優れた効果を示したグラフの２分割問題に関する研究[28]があるが、本問題の場合、系列性を保存する多分割問題であり、また、解の成分集合の個数、各成分集合の要素数が不定であり、より複雑化したグラフ分割問題となる。したがって、標準的なTabu Search法（近傍構造の設定等）の適用では効果的な結果は得られにくい。この複雑な構造に対して、本論文では、無閉路有向グラフの一列化グラフとその上で与えた分割を表現するブレイク・ポイントの集合によって解である系列分割を表現し、一列化グラフ上での系列分割を維持する頂点の移動、およびブレイク・ポイントの移動、付加、削除により、新しい解への移動を表現し、これらを複合統括して本問題に効果的な独自な近傍移動を実現し、Tabu Search法による算法を構成する。この複合的な近傍移動により解に大きな変化をほどこしTabu Search法の性能を引き出す。また、解の近似度を増すために、本問題の特徴を利用したヒューリスティックな戦略を組み込む。特に、ヒューリスティックな知識を取り入れたコスト変化

量の計算式、また求められた最良解の情報を利用することによって、さらに良好な解を探索する戦略などに効果が認められた。最後に本算法について数値実験を行った結果を示し、その性質、性能、および戦略の効果を明らかにするとともに、Tabu Search法の本問題への有効性と、より複雑なグラフ分割問題への利用可能性について述べる。

7．1．1 Tabu Search法の特徴

Tabu Search 法は局所探索法の変形である。通常の局所探索法の場合、探索の過程で現在の解 x から再びもとの解 x に戻るサイクリングが生じる可能性がある。このサイクリングを避けるためにタブーリストという記憶域を設け、一部の探索を禁止する処置をとる。すなわち、Tabu Search 法は最近の s 個の探索解をタブーリストに記憶しておき、それらを候補から除く、あるいは最近の s 個の探索解で生じた属性の変化をタブーリストに記憶し、これらの属性の逆方向への変化を禁止する処置をとる[16]。ここで、タブーリストをα とすると、Tabu Search 法では現在の解 x^{now} の近傍集合 $N(x^{now})$ からα を除いた集合 $N(\alpha, x^{now}) = N(x^{now}) - \alpha$ を構成し、その中から最小コストとなる解 x^{next} を選択し探索を進める。ただし通常は、α の要素として解そのものを扱わず解の移動に関与した属性を用いる。Tabu Search 法の構造上、解の近傍の構成法、およびタブーリストの要素、リストの構築等が、この算法にとって重要な役割[24],[28]を担い、大きな影響を及ぼす。この構成によりサイクリングをともなう無駄な探索を避け、$N(\alpha, x^{now})$ 中の最良な解を選択することにより近似度をより高める方向へと解を移動させる。また、算法の構造が単純に構成できるため実際の計算時間が少なくてすむ。および、Tabu Search 法は非常に柔軟性の高い枠組みであり、その特性を失うことなく種々の戦略を組込み性能を上げやすい等などの利点を持っている。

7．1．2 近傍構造とタブーリストの基本的考察

Tabu Search法の算法を構成するにあたって重要な構成要素は解の近傍構造とタブー要素の構成である。

無閉路有向グラフ$D(V,E)$の系列分割を$V_1,V_2,\ \cdots\ ,V_k$として、問題の解を$x=(V_1,V_2,\ \cdots\ ,V_k)$で表す。このとき、$V_i$によって誘導された$D(V,E)$の部分グラフを$\mathcal{D}(V_i)$とすると、$\mathcal{D}(V_i)$の有向辺が定める$V_i$の順序関係において、極大元または極小元となる頂点$v\in V_i$は系列分割の性質を変えることなく、他のある成分$V_j$に移動できる。図７．１の部分グラフ$\mathcal{D}(V_2)$に対して、極大元は頂点$b,c$にあたり、極小元は頂点$a$にあたる。まず説明のために記号$d^+(v,V_i)$と$d^-(v,V_i)$を導入する。$v\in V$とし、$v$を始点とする$D$の有向辺で成分集合$V_i$中に終点を持つ辺

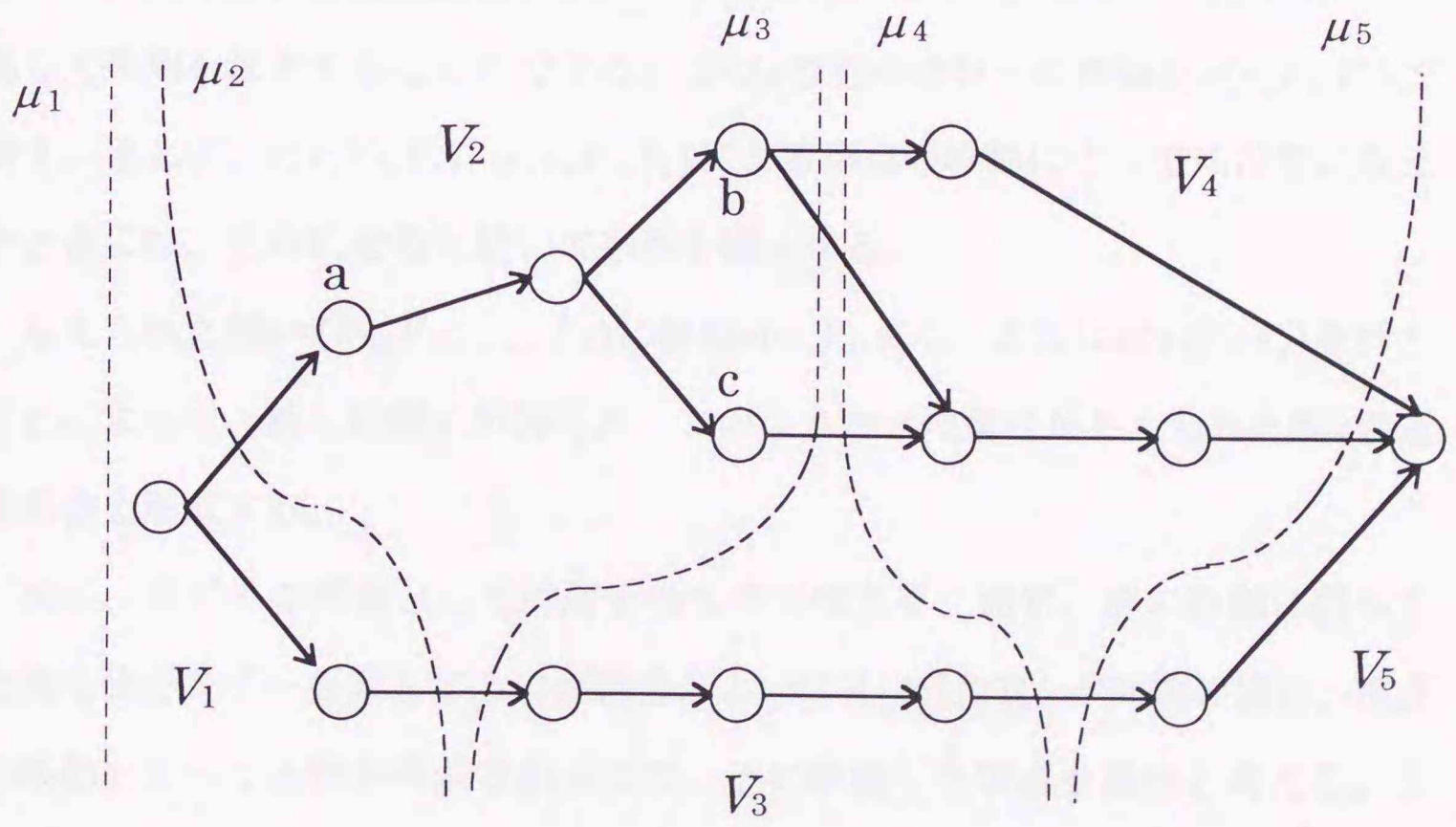


図７．１ 無閉路有向グラフの系列分割

の個数を$d^+(v,V_i)$で、またV_i中に始点を持ちvを終点とする有向辺の個数を$d^-(v,V_i)$で示す。$v\in V_i$が$(V_i,\preceq)$の極大元または極小元である条件は、それぞれ$d^+(v,V_i)=0$、また$d^-(v,V_i)=0$で書き表すことができる。vが極大元のとき、$i<m$に対して、$d^+(v,V_i)=d^+(v,V_{i+1})=\ldots=d^+(v,V_{m-1})=0$かつ$d^+(v,V_m)>0$が成立するならば、$v$は$V_j\in\{V_{i+1},\ldots,V_m\}$に移動可能である。これは、$D(V,E)$での$v$からの流出辺が$V_i,V_{i+1},\ldots,V_{m-1}$;$i<m$内の頂点に流入せず、$V_m$に初めて流入する状況を表している。図７．１の部分グラフ$\mathcal{D}(V_2)$に対する場合、極大元b,cはともにV_3とV_4へ移動可能である。同様に、vが極小元のときは$m<i$として$d^-(v,V_i)=d^-(v,V_{i-1})=\ldots=d^-(v,V_{m+1})=0$かつ$d^-(v,V_m)>0$のとき、$v$は$V_j\in\{V_m,\ldots,V_{i-1}\}$に移動可能である。これは、$v$への流入辺が$V_{m+1},\ldots,V_{i-1},V_i$;$m<i$内の頂点から流出していない状況に対応している。このV_iからV_jへのvの移動を$e(v;V_i,V_j)$で表す。また、同様な条件のもとで、vが極大または極小元のとき、V_jの直前または直後の位置までに新しい空の成分を作り、そこにvを移動して系列を拡大することができる。このvの空の成分への移動を$a(v;V_i,V_j)$で表す。さらに、$e(v;V_i,V_j)$, $a(v;V_i,V_j)$による頂点の移動によってV_iが空になったときには、このV_iを取り除いて系列を縮小する。

与えられた解$x=(V_1,V_2,\ldots,V_k)$に移動$e(v;V_i,V_j)$、または$a(v;V_i,V_j)$を行うことによって、新しい解x'が得られ、このようなx'の集合が与えられた解xの近傍集合を形成する。

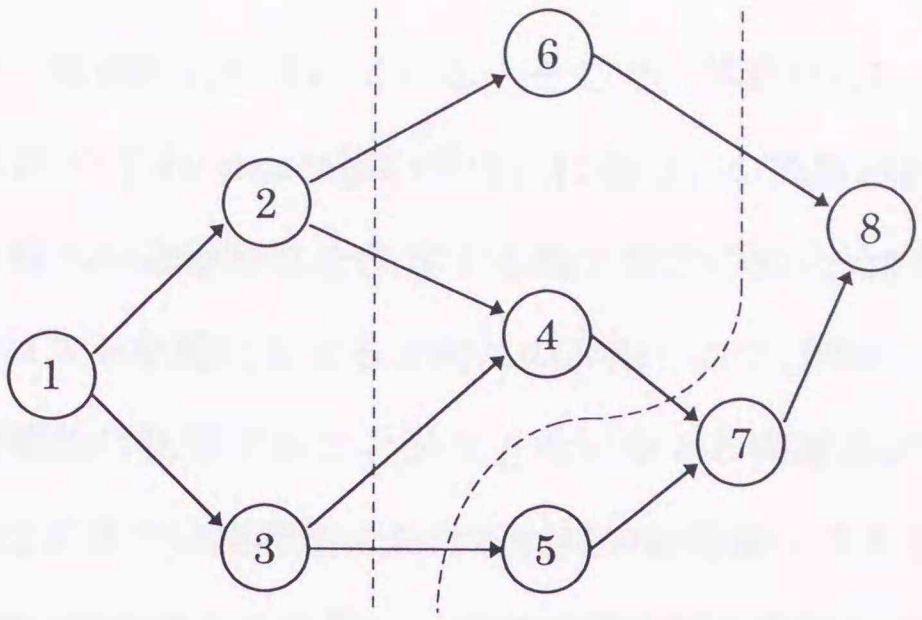
次に、タブーの要素としては解そのものではなく、通常、解の移動に関与した属性値をタブー要素とするのが都合がよい[24],[25],[22]。本問題の場合、頂点の移動によって近傍が構成されるので、その移動した頂点を属性と考える。ここで、$V_i|V_j$であるV_iのある頂点がV_jに移動したとき、その頂点をタブーリストtabu-to-leftに格納する。tabu-to-leftに格納されたこの頂点はV_jからV_iへの移動が禁止されることとなる。同様に、V_jの任意の頂点がV_iへ移動したとき、その頂

点をタブーリストtabu-to-rightに保存し、V_iからV_jの移動を禁断する。

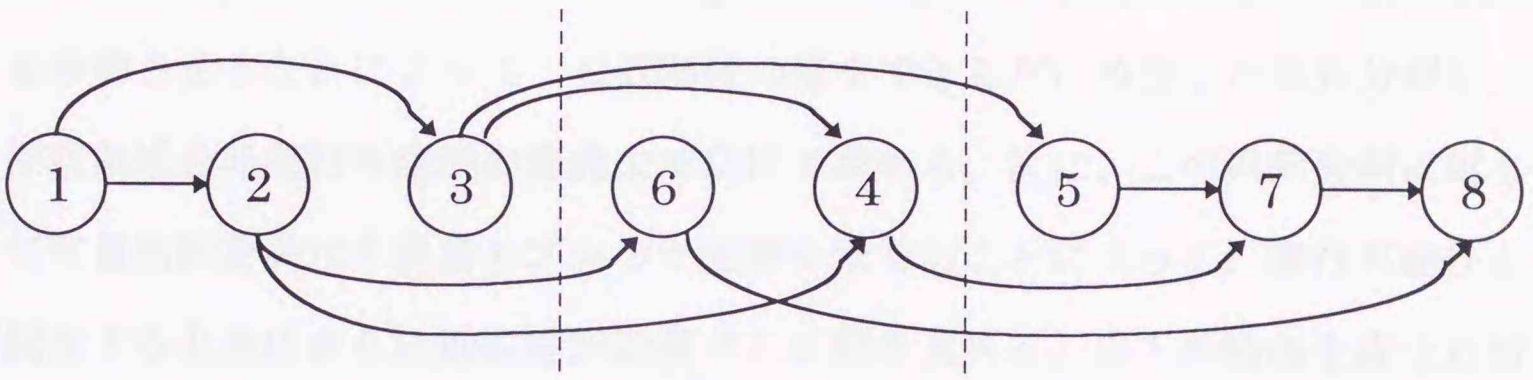
７．１．３ 無閉路有向グラフの一列化グラフによる表現

まず、系列分割問題で無閉路有向グラフを直接利用した算法構成をとると、集合演算を主体とする複雑な処理が要求され、プログラム的にも計算時間に無駄が多い。そこで、最適系列分割問題の探索法による近似解法を構成するにあたって、無閉路有向グラフを一列化グラフで表現し、系列分割を一列化された頂点列とブレイク・ポイントの列で表現する。この表現をとることによって、問題自体の理解と、系列分割の計算処理が容易になるばかりでなく、次章で提案する算法中に使用する最良のブロック分割を求める部分問題の最適化算法の構成を容易にする。

無閉路有向グラフ $D(V,E)$の頂点集合 V はいつでも一列化（トポロジカルソート）が可能である。頂点をこの一列化の順に左から右に並べ換えて得られる同型なグラフを一列化グラフと呼ぶ。一列化された頂点列を$\{v_1,v_2,\ldots,v_n\}$とすると、一列化グラフの各頂点はこの順に左から右に並んでおり、すべての辺の向きは常に左から右に向かっている。また連続した頂点の並びからなる部分列$\{v_p,v_{p+1},\ldots,v_q\}$は系列を保持する$V$の部分集合となることを容易に示すことができる。それゆえ、第５．３章で述べたように、無閉路有向グラフの系列分割は一列化グラフとその頂点列上の分割点（ブレイク・ポイント[6]）の並びを与えることによって生成できる（図７．２参照）。ここで部分列$\{v_p,v_{p+1},\ldots,v_q\}$をブロックと呼び、$[v_p,v_q]$,$[v_p,v_{q+1})$などの記法を用いて表わす。ブレイク・ポイントの挿入位置を示すインデックスを$1\le p_1<p_2<\ldots<p_k\le n$とし、ブレイク・ポイントの並びを$b_i=v_{p_i}$;$i=1,2,\ldots,k$とすると、ブロック$[b_i,b_{i+1})$;$i=1,2,\ldots,k$の系列はVの系列分割を生成



(a)　無閉路有向グラフの系列分割



(b)　(a)に対する一列化グラフの系列分割

図７．２　無閉路有向グラフと一列化グラフの関係

し、$[b_i,b_{i+1})$はその成分集合 V_i となる。したがって、無閉路有向グラフの一列化された頂点列とブレイク・ポイントの列のデータを管理することによって解とその移動を表現し、近傍移動の処理を効果的に構成することが可能になる。

７．１．４ 複合移動によるTabu Search法の実現

本問題はその特徴として、解の成分集合の個数、各成分集合の要素数がともに不定であり、また各成分集合は頂点の重みの和がブロックサイズBを越えては

ならないという容量制約を受けている。そこで、移動$e(v;V_i,V_j)$, $a(v;V_i,V_j)$を用いた標準的なTabu Search法の構成には幾つかの問題点がある。まず、現在の解から次の解への最適移動を決定する際の探索の組合せ的複雑さ、特に移動$a(v;V_i,V_j)$はコストを悪化させる方向にのみ働くので、何等かの先見がなければこの移動を積極的に利用することができないなどの問題点がある。これに対して、本論文ではグラフ分割問題における従来の近傍操作であるleft-to-right移動、right-to-left移動[22]の考えを拡張し、近傍移動の複合処理をほどこすことによって解に大きな変化を与え、問題の解決を計っている。まず、現在の解から容量制約を無視して、left-to-right移動、right-to-left移動の複合移動により複数の頂点を移動させることによって、目的関数の値をできるだけ改善した系列分割を、探索領域を非実行可能解の範囲まで広げて求める。次に、この系列分割に対して容量制約を満たす最良のブロック分割を求めることによって、実行可能性を回復すると共にさらに近似度が改善された解を求める。以上の過程を複合近傍移動として構築し、この独自の複合近傍移動をTabu Search法の枠組みのもとで、十分近似度の良い解が得られるまで反復するという方策を採用する。

（1） 複合近傍移動の実現

本問題の特性から、移動$e(v;V_i,V_j)$, $a(v;V_i,V_j)$の単純な適用による構成は処理の複雑化、解の近似度の向上の観点から問題が多い。そこで、一列化グラフの頂点列とブレイク・ポイントの集合によるデータ表現を用い、その上でブロック間の複合的な頂点の移動、あるいはブレイク・ポイントの移動、およびブレイク・ポイントの付加、削除などによって、複合的な近傍移動を実現し問題の解決をはかる。

まず、頂点の移動は本問題の特色に合わせて構成したleft-to-right移動、right-to-left移動を順次隣接する２つのブロック間に定義する。直接隣接する２つのブ

ロックV_i=[b_i,b_{i+1})、V_{i+1}=[b_{i+1},b_{i+2})に対してV_iに含まれている移動可能な頂点の中で、タブーリストtabu-to-right上にあるものを除き、移動によるコスト変化量が最小となる頂点v_0を見出し、これをV_{i+1}に移動する。このとき移動可能な頂点が存在しなければ、V_i、V_{i+1}はそのままとする。この処理を解x=(V_1,V_2, … ,V_k)に対してある演算子を作用させた結果とみなし、$x' = \bar{N}_i(\text{tabu - to - right}) \cdot x$で表わす。この演算子の操作で、解$x$の第$i$成分と第$i$+1成分は$V_i' = V_i - \{v_0\}, V_{i+1}' = V_{i+1} \cup \{v_0\}$と変化するが、その他の成分は全く変化を受けない。また、V_iからV_{i-1}への左移動に関しても、tabu-to-leftの考慮のもとで、最小移動コスト変化量をあたえる頂点 v_0をV_{i-1}へ移動する同様な演算子$\bar{N}_i(\text{tabu - to - left})$を用いて表わす。

次に、解の近似度をできるだけ早く高めるために、ブロックの容量制約を無視して上記のleft-to-right移動による改善を次のように反復して行う。

$$x' = \bar{N}_{k-1}(\text{tabu - to - right}) \cdot \bar{N}_{k-2}(\text{tabu - to - right}) \cdots \bar{N}_1(\text{tabu - to - right}) \cdot x$$

これを多重left-to-right移動と呼び、

$$x' = \overrightarrow{multiN}(\text{tabu - to - right}) \cdot x$$

で表わす。その結果に次のright-to-left移動の反復による改善を行う。

$$x' = \bar{N}_2(\text{tabu - to - left}) \cdot \bar{N}_3(\text{tabu - to - left}) \cdots \bar{N}_k(\text{tabu - to - left}) \cdot x$$

これを多重right-to-left移動と呼び、

$$x' = \overleftarrow{multiN}(\text{tabu - to - left}) \cdot x$$

で表す。

また、多重left-to-right移動, 多重right-to-left移動だけでは、解の容量制約に関する実行可能性が失われるばかりでなく、成分集合の個数、大きさに大きな変化をもたらすことは望めない。そこで、実行可能性を回復し、さらに近似度を高めるためにブレイク・ポイントの移動を試みる。多重移動によって得られた解x=(V_1,V_2, … ,V_k) がある一列化グラフとブレイク・ポイントの単調増加列

$\{b_1,b_2,\cdots,b_k\}$により表されているとする。このとき、ブレイク・ポイントの移動、新たな付加、および削除などによって実行可能性を回復して得られる実行可能解x'の集合を新たに解xの近傍解の集合λと解釈し、xからλの中で最も低いコストを示す解x'への移行を示す関数を$dp(x)$とする。ここで$dp(x)$は一列化グラフの最適系列分割問題を解く算法[6],[17]をそのまま利用可能である。この算法の計算量は$O(n)$であり、与えられた一列化グラフのもとで最適なブレイク・ポイント列$\{b_1',b_2',\cdots,b_{k'}'\}$を見出す。その結果、頂点のブロック間での移動によるコストの改善と、実行可能性の回復が行われる。

これらの処理を一反復過程中において

$$x' = dp(\overleftarrow{multiN}(\text{tabu - to - left}) \cdot \overrightarrow{multiN}(\text{tabu - to - right}) \cdot x)$$

の順序で行い、局所的に最良な近似効果を持つ近傍移動を達成する。

以上の処理は、複数回の基本操作$e(v;V_i,V_j)$, $a(v;V_i,V_j)$によって、現在の実行可能解から近似度のより改善された可能解に短時間で導く操作を一反復過程ごとに行っている。また、基本操作による隣接していないブロックへの頂点移動も可能である。これらより本処理の反復が、頂点の移動可能性を高めてより多様性のある組合せを生成することとなり、良質の実行可能解の多くを生みだすことになる。この考えにもとづくTabu Search法の算法を図７．３に示す。

（2） タブーリストの実現

解移動(x,x')に対しての逆向き移動を禁止する属性として、(x,x')のとき移動する頂点をあてることとし、多重left-to-right移動と多重right-to-left移動の２つの移動形式に対してそれぞれ２つのタブーリストtabu-to-leftとtabu-to-rightを設け、そのとき関与する頂点の集合をタブーリストに記憶する。タブーリストは待ち行列構造によって構成されるが、プログラム上では頂点集合に対応する一次元

Procedure Tabu_Search
begin
　　“無閉路有向グラフを一列化する。”
　　“実行可能解を与えるブレイク・ポイントの列を設定し、
　　　　初期解をxとする。”
　tabu-to-left := ∅ ;
　tabu-to-right := ∅ ;
　$x^* = x$;
　tabulength := positive integer ;
　t :=1;
　while stop-criterion <> yes do begin
　　$x' := \overrightarrow{multiN}(\text{tabu - to - right}) \cdot x$;
　　L_t := “解移動(x,x')により移動した頂点の集合” ;
　　tabu-to-left := tabu-to-left ∪ L_t ;
　　$x'' := \overleftarrow{multiN}(\text{tabu - to - left}) \cdot x'$;
　　R_t := “解移動(x',x'')により移動した頂点の集合” ;
　　tabu-to-right := tabu-to-right ∪ R_t ;
　　$x''' := dp(x'')$;
　　if $f(x''') < f(x^*)$ then $x^* := x'''$;
　　$x := x'''$;
　　tabu-to-right := tabu-to-right $-(R_{t-\text{tabulength}} - \bigcup_{i=t-\text{tabulength}+1}^{t} R_i)$;
　　tabu-to-left := tabu-to-left $-(L_{t-\text{tabulength}} - \bigcup_{i=t-\text{tabulength}+1}^{t} L_i)$;
　　$t := t + 1$;
　end;
　best solution := x^*
end;

図７．３ Tabu Search 法のアルゴリズム

配列を用意し、配列の添え字は頂点に対応するものとする[28]。このとき、初期化の段階で配列のすべての要素を0にしておき、移動対象となる頂点の要素にある正の数を入れ（すでに要素に値が入っていても、ある正の数を入れなおすこととする）、反復毎に1以上の要素の値に対して1減ずることによって、1以上の頂点を禁断対象とする。

ある正の数としてはV_iからV_{i+1}(V_{i+1}からV_i)へ、頂点vを移動したときのコストの変化量が負および正の２つの場合に対して異なる値を用意する。すなわち、変化量が改善されたならばtabulength1の値、そうでないのならばtabulength2の値を与える。図７．３ではtabulength1とtabulength2の値を同一なtabulengthとしている。tabulengthの長さにより禁断の期間を限定しているのは、探索が無駄に終わった場合、ふたたび元の探索解に戻ってそこから探索をやり直すという可能性を残すためである。この長さは理論的に推察することが困難であるため、通常、事前の数値実験の傾向によりその禁断の期間であるタブーリストの長さを判定する。

（３）　頂点移動にともなうコスト変化量の計算

解に$\bar{N}_i(\text{tabu - to - right})$を作用させた過程で生ずる頂点$v \in V_i$の$V_i$から$V_{i+1}$への移動に伴うコスト変化量$\bar{\delta}(v,V_i,V_{i+1})$、および$\bar{N}_i(\text{tabu - to - left})$における頂点$v \in V_i$に関する同様なコストの変化量$\bar{\delta}(v,V_i,V_{i-1})$は以下の計算式を用いることによって高速に計算することが可能である。ただし、この計算において、無閉路有向グラフ上に存在しない辺については考慮しないものとする。

$$\bar{\delta}(v,V_i,V_{i+1}) = \sum_{s \in V_i} c(s,v) - \sum_{t \in V_{i+1}} c(v,t) \tag{7.1}$$

$$\bar{\delta}(v,V_i,V_{i-1}) = \sum_{t \in V_i} c(v,t) - \sum_{s \in V_{i-1}} c(s,v) \tag{7.2}$$

これらの式の使用は、グラフが並列構造を持つとき良好な結果に導くが、全く無秩序なランダムグラフに対しては不満足なことが多い。本問題の場合、一列化グラフ上の頂点間の辺の長さ（頂点の添字の差）をより短くするというヒューリスティックな戦略が効果をもつことが経験により示される。これを反映するために、コスト変化量式を以下のように変形しその効果を確かめる。一列化グラフ上で、vからの流出辺（への流入辺）の端点となる頂点の中で最も左側（右側）に存在する頂点の直前（直後）にvを移動すると考えたとき、このvを左端点（右端点）とするブロックで、頂点の重さの総和がブロックサイズBを越えないものの中で、重さの総和が最大であるものを$V^+(V^-)$で表わす。このとき、$V^+(V^-)$によって表される以下の新たなコスト変化量を用いることによって、ランダムグラフにも、構造を持ったグラフにも対応できる高い近似度を達成することが可能となる。

$$\bar{\varepsilon}(v,V_i,V_{i+1}) = \sum_{s\in V_i} c(s,v) - \sum_{t\in V^+} c(v,t) \tag{7.3}$$

$$\bar{\varepsilon}(v,V_i,V_{i-1}) = \sum_{t\in V_i} c(v,t) - \sum_{s\in V^-} c(s,v) \tag{7.4}$$

（4） 終了判定基準

終了判定基準としては、ある反復回数iterateを繰り返す方法、または探索解が更新された段階から、この値が更新されなくなってからの経過時間がパラメーターstopを越えたときに、算法が終了する方法などが考えられる。求まる最良解が2000前後の反復回数以内で決定される場合が多いことから、前者の方法ではiterateを2000前後とし（並列グラフの場合、1000以内で求まることが多いので1000前後とする。）、後者の方法ではパラメータstopを1000前後とするのが望ま

しい。さらに、精度を考慮した場合、上記の値の2倍程度に設定する。

（５）　局所最適解からの再出発による近似度の改善

本問題で得られる局所最適解の頂点列と最適解の頂点列の構造は部分的な配置が異なるのみで、解の構造としては近い関係にある。すなわち、分割されたブロックの部分的一致が見られる。この性質より、以前に探索した良好な解の情報を積極的に使い、それを出発点とし、さらに良好な解を探索することによって解を改善する戦略が考えられる。この考えを取り入れ、現在得られている最良解とその暫定値を初期値として、タブーリストをすべて空として再び探索を進めることは有効な結果をもたらす可能性が強い[28]。また、再出発による解の決定は反復回数の初期の段階で決定される傾向がみられるので、この過程での終了判定基準に用いる反復回数iterate、およびパラメータstopの値は前記で示した数の1/2の値を用いてる。

（６）　２頂点交換法による近似度の改善

ある一列化グラフとブレイク・ポイントの単調増加列により表現された系列分割において、この一列化グラフ上に有向辺で結ばれていない隣接する２頂点v_i, v_{i+1}が存在するとき、一列化の性質を変えることなく、これらの２頂点を交換することができ、この交換によって新しい一列化グラフが得られる。v_iとv_{i+1}の間に切断を考慮したときに、もとの一列化グラフから得られるこの切断のカットエッジのコストをc、２頂点を交換した新しい一列化グラフから得られる同様なカットエッジのコストをc'とし、もし$c' < c$が成立するならばv_iとv_{i+1}の交換をおこなうという操作を考える。ブレイク・ポイントの移動により一層のコスト改善を行う次の段階に進む前に、この操作をすべての交換可能な２頂

点に対して行っておくと、次の改善過程で２頂点の交換の位置に丁度ブレイク・ポイントの移動または生成が生じたとき、よりコストが改善された新たな系列分割が作り出される。この予知にもとづく方法を２頂点交換法と呼ぶ。本論文では現在得られている最良解に２頂点交換法を適用し、それに続いて$dp()$操作で解の近似度を上げる方策を試みている。

７．１．５　数値実験による特性評価

本章では上記で構成したTabu Search法の算法に対して特性評価と挙動解析を行う。なお、以下の数値実験は小樽商科大学情報処理センターSun SparcStation 2およびDEC3000上で行い、計算時間に関する数値実験はSun SparcStation 2を使用した。まず、Tabu Search法を効果的に動作させるための最も重要なパラメータは禁止リストの長さである。本問題においては２つの禁止リストtabu-to-leftおよびtabu-to-rightを設けているが禁止リストの長さのパラメータは両禁止リストとも共通とする。このとき、任意の頂点の移動に関与するコスト変化量が改善された場合tabulength1、そうでない場合tabulength2と２つのパラメータを用いる方法が考えられる。しかし、tabulength1の値に対して多少低めにtabulength2の値を設定することにより若干の効果はあるものの、著しい効果は認められずその効果は低い。したがって、両者とも同じ値であるtabulengthとし、算法を構成している。このtabulengthの適正値を判定するため、tabulengthの値の変化に対してコストの値の変化を数値実験で確かめる。

図７．４、７．５、７．６は頂点数500のランダムグラフに対しての数値実験の一例であり、それぞれブロックサイズが10、30、50の場合を扱う（コスト変化量式は（７．１）、（７．２）を使用、パラメータiterateは4000とし、近似度の改善は行わない）。この実験より、ブロックサイズ10の場合、tabulengthの長さは1、または2が適正であり、ブロックサイズ30の場合、6前後であり、ブロック

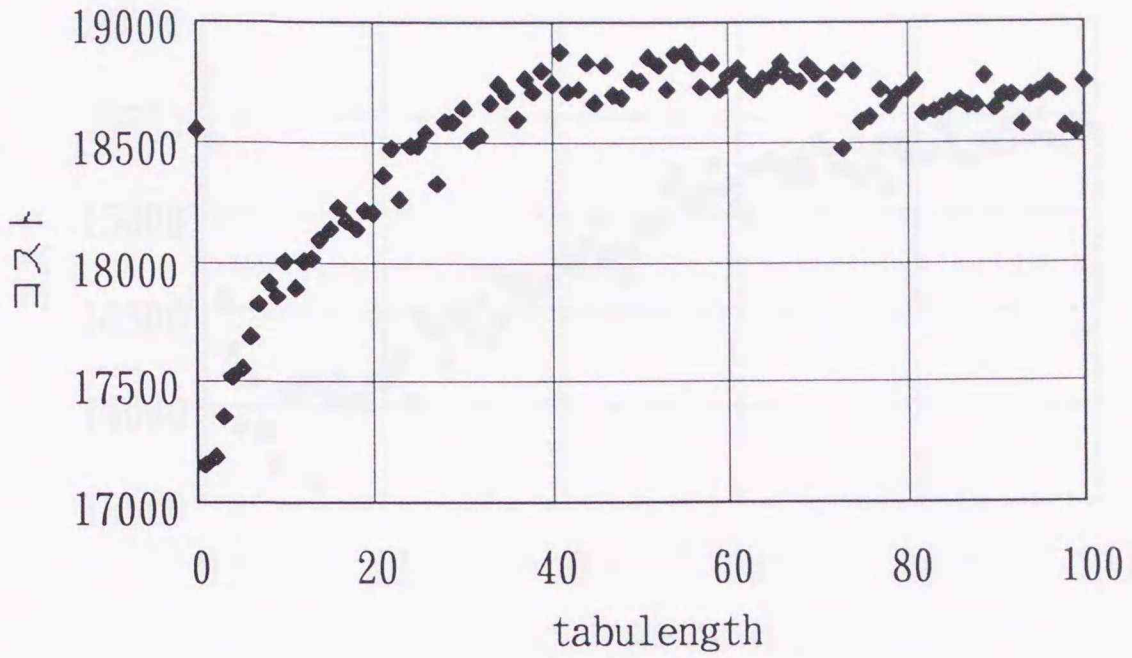


図７．４　ブロックサイズ 10 に対するタブーリストの長さとコスト

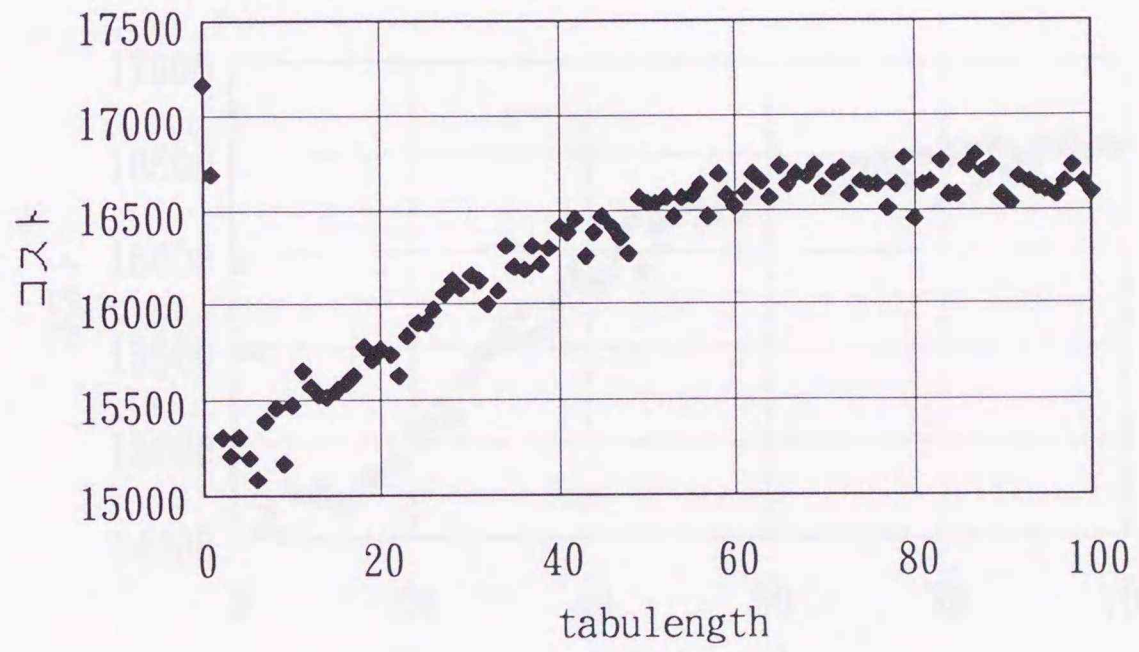


図７．５　ブロックサイズ 30 に対するタブーリストの長さとコスト

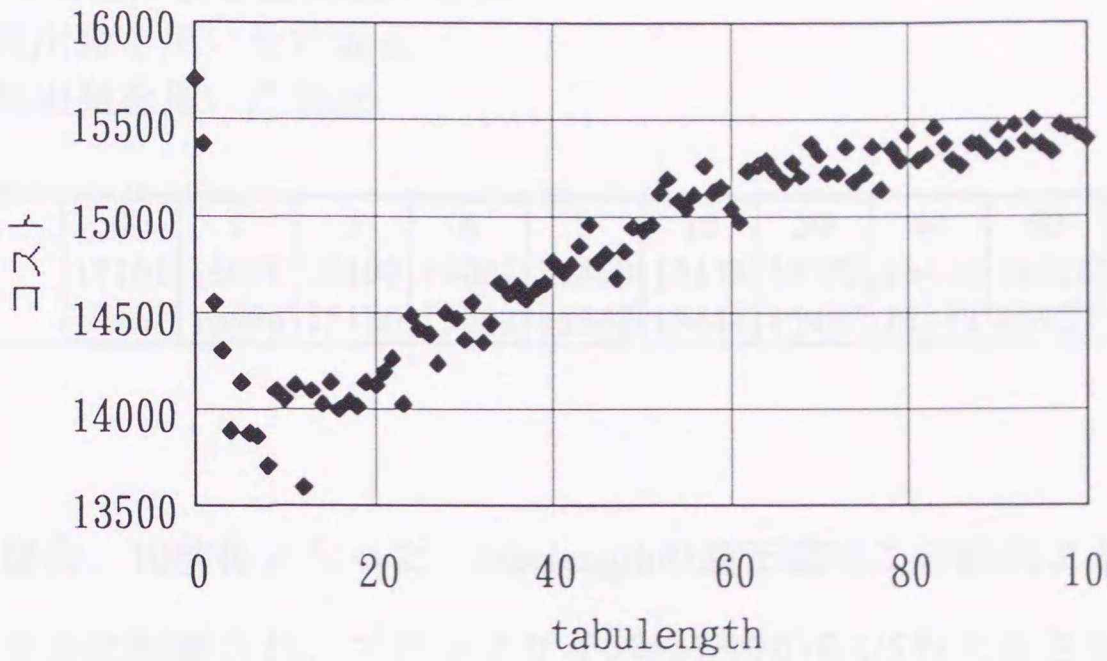


図７．６　ブロックサイズ50に対するタブーリストの長さとコスト

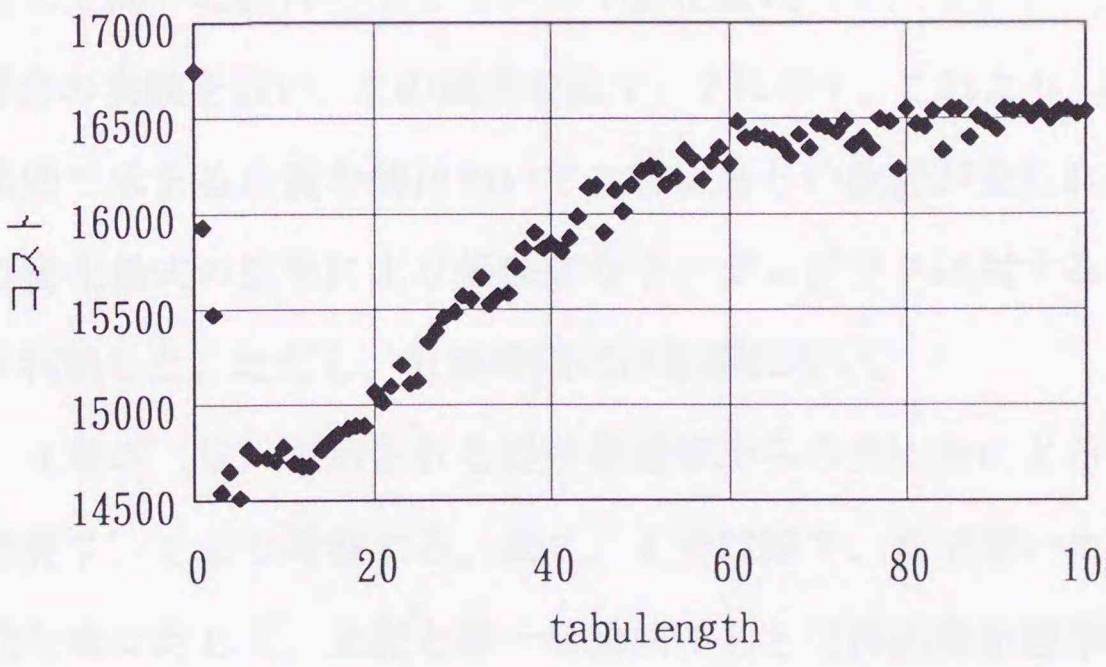


図７．７　$\vec{\varepsilon}(\bar{\varepsilon})$に対するタブーリストの長さとコスト

表７．１　再出発による近似度の評価
Algo1: 再出発を用いない場合、
Algo2: 再出発を用いた場合

tabulength	0	1	5	6	8	10	20	40	60	80	100
Algo1	17161	16696	15190	15081	15460	15478	15772	16416	16528	16449	16598
Algo2	17161	16696	15180	15042	15342	15441	15487	16173	16127	16036	16103

サイズ50の場合、10前後となった。tabulengthの適正値はこの結果よりブロックサイズの大きさに影響され、ブロックサイズの1/10から1/5の大きさをtabulengthの適正値とすることが望ましい。このとき、グラフの構造に対するこの適正値の影響は少ない。

次に、第７．１．４章の（３）で述べた改良型のコスト変化量の計算式（７．３）、（７．４）を用いることによる効果について確かめる。図７．５で用いたグラフに対して同一の条件のもとでコスト変化量式（７．３）、（７．４）を適用した場合の実験を行い、その結果を図７．７に示す。これより、tabulengthが20以内の範囲で求まる良質な解において、特に著しい改善が見られた。したがって、この変化量式の採用により無秩序なランダムグラフに対する効果が得られることが判明した。ただし、計算時間には影響はない。

第７．１．４章の（５）で示される局所最適解からの再出発による近似度の改善の効果は表７．１より考察する。表７．１では図７．５で用いたグラフに対して代表的な値に対して、上記と同一な条件のもとで再出発を適用しない場合のコスト値と、適用した場合のコスト値との比較を示している。これより、再出発の効果により解が改良されていることがわかる。ただし、解の近似度が悪い場合、著しい効果をもたらすが、良質な解の場合、改善率は低い傾向が見出される。しかし、２頂点交換法を適用した場合、これによる改善が示される例は少なく、また改善される場合もその改善率は低い傾向にある。

表７．２　長期メモリーによる近似度の評価
Algo1: 長期メモリーを使用しない場合、
Algo2: 長期メモリーを使用した場合

tabulength	0	1	5	6	8	10	20	40	60	80	100
Algo1	17161	16696	15190	15081	15460	15478	15772	16416	16528	16449	16598
Algo2	16007	15708	15637	15848	16058	16002	16378	16942	17160	17141	17129

論文[28]で述べられている長期メモリーの効果について実験の結果を加えておく。長期メモリーは長期的な巡回を防ぐため、頻繁に移動を繰返す頂点に対して移動しにくくさせるものであり、任意の頂点の移動回数とパラメーターBIASの積をペナルティ関数としてコスト変化量に加算し実現する。上記と同様な条件で図７．５で用いられたグラフを使用した一例を表７．２に示すと、明らかに長期メモリーによる改悪の現象が見られる。tabulengthが0、1のときは改善されているが、これはtabulengthの長さが0あるいは短いため、タブー効果が存在しない算法となり、これに対して長期メモリーを設けることよりタブー効果が付加され、タブーサーチの戦略と同一の構成となるものと思われる。また、BIASを上げることによりさらに改悪される傾向が示されている。この結果、少なくとも本問題に対しては長期メモリーは近似度の悪化の要素となる。

この問題に対して、2並列グラフに対しては厳密解を求めることが可能であるので、Tabu Search法との比較を試み、解の近似度について論じる。この厳密解法は探索領域の削除を考慮した動的計画法にもとづく算法である。表７．３はブロックサイズ、エッジコスト、中間エッジを変化させた場合の頂点数200の2並列グラフに対するコストの比較である。コスト変化量式は（７．１）、（７．２）を使用し、パラメータiterateは2000とする。さらに、近似度の改善のアルゴリズムを加えている。しかし、グラフf以外は最良解が反復回数1000以内で発見され、また、fとhに対してのみ２頂点交換法が行われている。表７．３よりエ

表７．３　２並列グラフに対する近似解と厳密解

グラフ	a	b	c	d	e	f	g	h
エッジコスト	1				1から10			
ブロックサイズ	10	10	40	40	10	10	40	40
中間エッジ	0	20	0	20	0	20	0	20
厳密解法	20	40	5	19	51	146	6	87
Tabu Search	20	41	5	20	58	168	6	89

表７．４　中間エッジ数による影響と計算時間

中間エッジ数	0	50	100	150	200	250	300
厳密解法	20	69	115	162	203	253	298
厳密解法の時間(ms)	23820	20352	25689	22075	24747	26843	25812
Tabu Search	20	70	119	167	211	260	307
Tabu Search の時間(ms)	7023	9699	8443	7687	8493	12014	8083

ッジコストがすべて1の場合、Tabu Search法の結果は最適解にほとんど近い値か最適解そのものとなる。また、エッジコストがすべて同一な値に対しての実験においても同様な傾向にあることが示されている。しかし、エッジコストの値が幅広く分散する場合、近似度は前記に比べて悪くなる傾向がある。これはカットされる辺のコスト値が大きな範囲を持つため、カットの選びかたにより、解の値に大きな影響を及ぼす結果となるためである。しかし、ブロックサイズが大であると、そのカットの選びかたに多様性があるので最良な値を選びやすく、上記の影響は現れにくい。3並列等の同様な実験でも同じ傾向が示された。しかし、エッジコストの分散による影響は強くでる傾向がある。

表７．５ 頂点数による影響と計算時間

頂点数	50	100	150	200	250	300
厳密解法	5	10	15	20	25	30
厳密解法の時間(ms)	699	3808	11729	23638	45947	76949
Tabu Search	6	11	15	20	25	31
Tabu Searchの時間(ms)	1419	3049	4709	6449	8228	9898

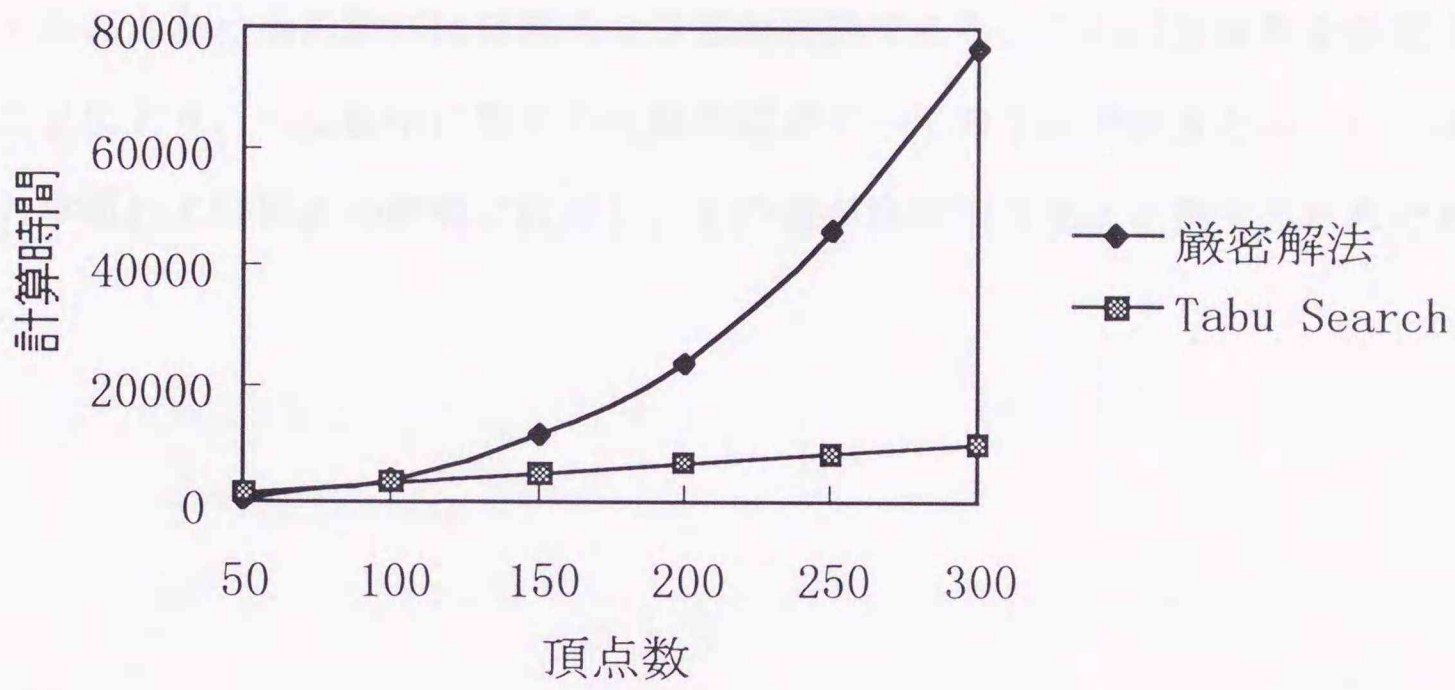


図７．８　頂点数と計算時間（ms）

また、表７．４で並列路の間を結ぶ中間エッジが入ることによるグラフの複雑性に対して、厳密解法と比較したコスト、ならびに計算時間を併記し、近似度と計算の負担への影響を示す。本実験で使用する終了規則はパラメータiterateの値を1000とする。また、コスト変化量式は（７．１）、（７．２）を使用し、解の改善は行っていない。これより中間エッジが増加することによるグラフの複雑性に本算法は影響を受けず、近似度（相対誤差）5%以内を維持する。

最後に、２並列グラフに対しての頂点数に対する計算時間とコストの変化を表７．５で示し、特に時間の増加傾向を図７．８で表す。このときのパラメー

タiterateは1000であり、コスト変化量式は（７．１）、（７．２）を採用する。また、近似度の改善は行っていない。この実験より、本算法では最適解かほぼそれに近いコストの値が求まることが示される。さらに、厳密解法はべき乗で計算時間を要するのに対して、Tabu Search法はほぼ線形の計算時間で求まることが示される。また厳密解法では頂点数の大きいランダムグラフに対しては実質的に計算が困難であるが、本算法ではグラフの構造およびその他の戦略に依存することなく頂点数にほぼ線形な計算が可能である。これは反復数を限定することにより、一反復中に要する計算時間がすべての頂点の探査とエッジコストの参照および頂点の移動に依存し、その値がほぼ頂点数に比例するためである。

7．2 確率的複合移動によるSimulated Annealing法

無閉路有向グラフの系列分割問題は一般に指数的オーダーの計算時間を必要とし、実用的な時間内での厳密解の導出は困難である。これに対して、前章でTabu Search法[7],[8]による近似解法を提案した[21]。この解法では並列構造をもつ無閉路有向グラフに対して、相対誤差5%以内の精度で解を得ることができた。しかし、ランダムなグラフに対しては局所解に落ち込みやすい傾向なども示された。

本章では、局所解からの脱出性をより強めるため、Simulated Annealing法[2]の考えにもとづく解法を提案する。本問題はその特徴として、解の成分集合の個数、各成分集合の要素数がともに不定であり、これがTabu Search法と同様に、解の近傍移動などの処理を複雑にし、標準的なSimulated Annealing法による解法の構成を難しくしている。本章では、前章で用いいた複合移動の考えを用い、これをSimulated Annealing法に効果的な複合移動に改良する。すなわち、頂点の移動法に修正を加え、また、この頂点移動の過程において独自の基準を用いてSimulated Annealing法の考えを取入れている。さらに本算法について使用した複合移動の効果を数値実験で示し、複合移動の考えがSimulated Annealing法にも有効であり、かつ、その実用的効果を明らかにする。また、Tabu Search法による解法との比較を行い、この提案算法ではTabu Search法より良質な解が導出され、頂点数に線形な計算時間で求められることを示す。

7．2．1 確率的複合移動によるSimulated Annealing法の実現

本問題はその特徴として、解の成分集合の個数、各成分集合の要素数がともに不定であり、これが解の近傍処理などの操作を複雑にし探索法の構成を難しくしている。そこで、前章の Tabu Search 法で用いた複合移動の考えが Simulated Annealing 法にも有効に働くことを示す。また、前章と同様な考えのもと、一列化グラフ上での頂点の移動と解の表現を管理することとする。すなわち、本章ではグラフ分割問題における従来の近傍操作である left-to-right 移動、right-to-left 移動[26]をブロック順を考慮して多重的に適用することにより、容量制約を考慮せずに、一列化された頂点の並びをより望ましい一列化の並びに積極的に変化させる多重頂点移動を構成する。次に、この新しい一列化に対して、容量制約を満たす最良のブロック分割を求める部分問題を解くことによって、実行可能性の回復と近似度のさらなる向上を計る。以上によって構成された解の複合移動の形成過程に Simulated Annealing 法[2],[4],[5]の考え方を取入れた近似解法を提案する。標準的な Simulated Annealing 法においては解の移動の前後におけるコスト差により採択基準を決定するのに対し、本近似解法では多重頂点移動における局部的な頂点移動ごとに採択基準を定め、この基準のもとで確率的に採択された頂点移動のみを合成し、それにもとづき解の確率的複合移動を形成する。この複合移動により、解に大きな変化をほどこし Simulated Annealing 法の性能を引き出す。ここで提案する方策の有効性とその特徴については第７．２．２、７．２．３章の数値実験の結果とあわせて述べる。

（１）　Simulated Annealing 法の複合移動の構成

近傍構造の構成は、基本的には無閉路有向グラフを変換した一列

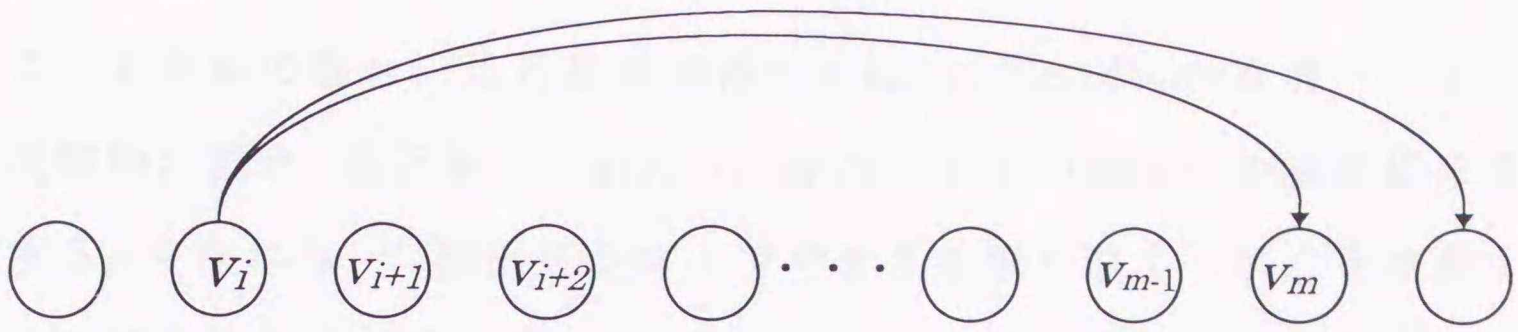


図７．９　一列化グラフの頂点の並び

化グラフの性質を壊すことなく、頂点の左あるいは右へ移動することによって構築する。Tabu Search法の場合、隣接するブロックへ頂点を移動する戦略が有効であった。しかし、Simulated Annealing法の場合、この戦略は効果的な結果が示されなかった。そこで、頂点の移動方法に関して、以下の考えのもとでSimulated Annealing法の効果を引き出す試みを行った。$d(v,u)$を$v \in V$を始点、$u \in V$を終点とする有向辺が存在するなら１、そうでなければ０の値をとる関数とする。このとき、$d(v_i,v_{i+1})=d(v_i,v_{i+2})=\ldots=d(v_i,v_{m-1})=0$かつ$d(v_i,v_m)>0$が成立するならば、$v_i$は$v_{i+1}$の直後から$v_m$の直前の位置までの範囲内で、一列化の性質を破壊することなく右移動が可能である。図７．９の一列化の並びではv_{i+1}の直後への移動とはv_{i+1}とv_{i+2}の間にv_iを移動することであり、v_mの直前に移動するとはv_{m-1}とv_mの間にv_iを移動することである。このv_mをv_iに対する一列化グラフ上での最近接後続点と呼ぶ。同様に、$d(v_{i-1},v_i)=d(v_{i-2},v_i)=\ldots=d(v_{q+1},v_i)=0$かつ$d(v_q,v_i)>0$が成立すれば、$v_i$は$v_q$の直後から$v_{i-1}$の直前の範囲内で左移動が可能である。この$v_q$を$v_i$の最近接先行点と呼ぶこととする。以上のように、一列化グラフ上の任意の頂点はその最近接後続（先行）点によって定まる範囲内で、任意の場所に移動可能である。しかし、本稿で提案する解法では頂点$v_i \in V_i$に対し

て、右移動の場合、最近接後続点v_mが$v_m \notin V_i$ならばv_mの直前へ、また左移動の場合、最近接先行点v_qが$v_q \notin V_i$ならばv_qの直後への移動に限定する。すなわち、一列化グラフ上でできうる限り遠くへ頂点を移動する方策をとる。以後、頂点vの最近接後続点の直後への頂点移動、およびvの最近接先行点の直前への頂点移動を、longmove-right(v)、longmove-left(v)で表す。なお、上記の移動の条件が成立しないときは頂点の移動は行わず、頂点の並びはそのままとする。

以上の頂点移動をベースとし、Tabu Search法の複合移動の考えと同様に、順次連続するブロックに適用し非実行可能解の範囲まで探索を広げる多重的な近傍移動を構築する。これにより、解に大きな変化をほどこしより強く最良な解の方向へ移動させ近似度の改善を行う。まず、任意の探索過程での解$x=(V_1,V_2, \ldots ,V_k)$はある一列化グラフDの頂点の並びと、ブレイク・ポイントの集合$\{b_1,b_2,\cdots,b_k\}$によって表現される。このとき、ブロック$V_i=[b_i,b_{i+1})$中からランダムに選択した頂点をv_iとする。分割集合$V_1,V_2, \ldots ,V_{k-1}$から、それぞれランダムに選びた した頂点$v_1,v_2, \ldots ,v_{k-1}$に対して定義されるlongmove-right移動の系列

longmove-right(v_1), longmove-right(v_2), ... , longmove-right(v_{k-1})

を、出発解$x=(V_1,V_2, \ldots ,V_k)$から始めて連続適用し、解$x'=(V_1',V_2',\cdots,V_k')$を得る。この際、ブレイク・ポイントの移動は考慮しないので、ブロックの容量制約は無視して頂点移動を行うことになる。この頂点の連続した移動を多重longmove-right移動と呼び、

$$x' = \overrightarrow{multiN}(x)$$

で表す。この結果に対して、再び$V'_k,V'_{k-1},\ldots,V'_2$からそれぞれ頂点$v_k,v_{k-1},\ldots,v_2$を改めてランダムに選び、それらに対応して定義されたlongmove-left移動の系列

longmove-left(v_k), longmove-left(v_{k-1}), … , longmove-left(v_2)

を連続適用することにより解の改善を行う。これを多重longmove-left移動と呼ぶこととし、同様に、

$$x'' = \overrightarrow{multiN}(x')$$

で表す。

また、多重longmove-right移動、多重longmove-left移動だけでは、解の実行可能性が失われるばかりでなく、成分集合の個数、大きさに大きな変化をもたらすことは望めない。そこで、実行可能性を回復し、さらに近似度を高めるために、多重頂点移動によって得られた一列化グラフに対して容量制約を満たす最良なブロック分割を求める部分的な最適化を組み入れる。すなわち、多重移動によって得られた解$x''=(V_1'',V_2'',\cdots,V_k'')$に対して$dp(x'')$を適用する。その結果、頂点のブロック間での移動によるコストの改善と、実行可能性の回復が行われる。これらの処理を一反復過程中において、

$$x''' = dp(\overrightarrow{multiN}(\overrightarrow{multiN}(x)))$$

の順序で行い、Simulated Annealing法にとって良好な近似効果を持つ複合移動を達成する。

（２）　確率的複合移動によるSimulated Annealing法の構成

通常のSimulated Annealing法では、反復過程中において現在の解xから上記の複合移動により導かれた解x'に対して、そのコスト差cost(x)-cost(x')を基準として確率的採択の計算を行い、移動の採否を決定する。これに対して本研究では、各頂点移動longmove-right(v)、longmove-left(v)ごとに局部的な採択基準量$\delta^{\pm}(v)$を計算し、改善されたときと、また非改善のときは確率exp(-$\delta^{\pm}(v)$/t)で頂点移動の処理を

行い、これらを合成して複合移動を構成する。したがって、実際の算法では前記の多重頂点移動過程中に確率的要素を導入することとなる。

頂点移動 longmove-right(v)、longmove-left(v)を実行すると、それにともなって目的関数値が変化する。その効果は通常、増分コストを用いて評価する。しかし、本論文では、以下で新たに定義する量$\delta^+(v),\delta^-(v)$を使用して頂点移動の効果を測定する。まず、これらを定義するために次の記号を導入する。一列化グラフ上の任意の頂点 v_pを左端点または右端点とし、大きさbの容量制約を満足し最大の重みを持つブロックをそれぞれ$V^+(v_p,b)$または$V^-(v_p,b)$で表す。すなわち、$w(v)$を頂点vの重みとして、制約$\sum_{i=0}^{q} w(v_{p+i}) \le b$を満たす$q$の最大値を$r$、$\sum_{i=0}^{q} w(v_{p-i}) \le b$を満たす q の最大値を s とすれば、それらはそれぞれ$V^+(v_p,b)=[v_p,v_{p+r}]$または$V^-(v_p,b)=[v_{p-s},v_p]$と定義される。頂点移動 v の最近接後続点をu、最近接先行点をu'として$\delta^+(v),\delta^-(v)$を

$$\delta^+(v)=\sum_{s\in V^-(v,B)} c(s,v)-\sum_{t\in V^+(u,B-w(v))} c(v,t) \tag{7.5}$$

$$\delta^-(v)=\sum_{t\in V^+(v,B)} c(v,t)-\sum_{s\in V^-(u',B-w(v))} c(s,v) \tag{7.6}$$

と定義する。これらは何れも頂点vを実際に移動する前に計算可能な量である。また、これは頂点移動によって一列化グラフ上で大きなコストを持つ辺の長さ（頂点の添字の差）をより短くし、できるだけ容量Ｂのブロック内にまとめると良い結果が得られるという経験則を反映させ、従来の増分コストの定義を修正して導びいたものである。さらに、この量は現在のブレーク・ポイントの位置を意識せずに計算でき、そのため計算処理を簡素化できる利点がある。

確率的採択基準に$\delta^+(v),\delta^-(v)$を用いSimulated Annealing法の考えに

```
Procedure 確率的多重longmove-right移動
for i=1 to k do begin
    v = Viの中からランダムに移動可能な頂点を一つ選択する。;
    if  δ+(v) <= 0   then
        longmove-right(v);
    else
        if exp(-δ+(v)/tmp) >= random[0,1) then
            longmove-right(v);
end;
```

図７．１０　確率的多重 longmove-right 移動の手続き

準拠して多重頂点移動を構成する算法について具体的に説明する。多重longmove-right移動の場合、頂点$v \in V_i$に対して$\delta^+(v)<0$は解の総コストが改善される傾向を示している。この条件$\delta^+(v)<0$が成立した場合には無条件で、また成立しない場合には確率$\exp(-\delta^+(v)/t)$で、最近接後続点の直前への頂点移動を行う。この処理をブロック$V_1, V_2, \ldots, V_k$の順に適用した結果、新たな解x'への移動が行われる。これを新たに確率的多重longmove-right移動と呼び、その手続きを図７．１０に示す。

同様に、確率的採択基準δ^-を用いて確率的多重longmove-left移動を定義する。

解xに対して、この確率的多重longmove-right移動を作用させた結果を$x' = \overrightarrow{RandMultiN}(x)$とし、この結果に確率的多重longmove-left移動を作用させた結果を$x'' = \overleftarrow{RandMultiN}(x')$で表す。これにもとづく解の複合移動を確率的複合移動という。

この確率的複合移動ベースとし、Simulated Annealing法の考えにし

```
Procedure simulated annealing
begin
    tmp := initial temperature;
    x := initial solution; x*:=x;
    while tmp > stoptmp do begin
        for i:= 1 to r do begin
            x' = RandMultiN(x);
            x'' = RandMultiN(x');
            x''':= dp(x'') ;
            if f(x''') < f(x*) then x* := x''';
            x := x''' ;
        end;
        tmp := tmp × phi;
        r := r × tau;
    end;
    best solution := x*;
end;
```

図７．１１　simulated annealingにもとづくアルゴリズム

たがって図７．１１の近似解法を構成する。この解法では計算時間と解の質が上記のパラメータtmp,stoptmp,r,phi,tauに強く依存することは明らかである。まず、tmp,stoptmpの値を受理比[2]（＝受理された移動数／要求された移動数）より決定する。理論的にはこの値を1.0から0.0へと変化させれば、良質な解が得られることはすでに知られている。しかし、このような変化に基づき実行すれば、計算時間を多大に必要とせざるをえない。本問題では傾向として0.85から0.15まで受理比を変化させることにより実用的計算時間内で良質な解が得られ、その範囲を越える変化により解を求めても、いちじるしい解の改善は見られなかった。したがって、受理率が0.85から0.15まで変化す

るようにtmpとstoptmpを設定する。これは単純な予備実験で容易に求まる。このとき、初期温度は無閉路有向グラフの辺のコストの大きさに依存し、その最大値の大きに対して1.4倍前後の値をとればよいことが判明した。

phiについては0.8,0.85,0.9,0.95とパラメータの値を変化させ効果を確かめてみた。値が高いほど良質な値が求まるが、それに反して計算時間が必要となる。ここでは、0.9を採用することにより解の質および計算時間の観点から良好な結果が得られている。

同様にして、rの値は5nからn/5までの範囲を調査し、その結果、rの初期値は頂点数nの1/5から1/8が適正であることが判明した。また、tauは1.0から1.2の範囲を調べた結果1.15以上になると比較的計算時間が高くなることから、1.1を採用する。

７．２．２　確率的複合移動の効果

確率的複合移動を用いて構築した提案算法と以下で述べる３つの近傍移動を用いた戦略の効果を数値実験的に検討し、確率的複合近傍移動が効果的な近傍移動を実現することを示す。代表的な実験例として、頂点数100から500の辺数がほぼ頂点数の３倍となるランダムグラフに対する結果を表７．６に示す。そこに示される値は１０回[2]の実験で求まる解のコストの平均値をそれぞれ示したものである。また、その他のグラフに対する実験でも同様な結果が得られている。

まず、今回提案する近傍移動による算法をProg1とする。これに対して、頂点をできうる限り一列化グラフ上で遠くへ移動する提案の効果を立証すために、頂点をその頂点が現在存在するブロックに隣接するブロックへ移動する短距離的な移動を採用した近傍移動による戦

表７．６ 各戦略法による計算結果

problem	Prog1			Prog2			Prog3			Prog4		
size	mean	best	worst	mean	best	worst	mean	best	worst	mean	best	worst
100	2460	2431	2480	2495	2450	2553	2500	2470	2526	2587	2545	2612
200	5632	5621	5694	5926	5817	6016	5784	5758	5816	6095	6065	6167
300	8755	8680	8796	9390	9312	9493	8905	8821	8985	9493	9445	9539
400	11508	11370	11637	12640	12499	12789	11769	11666	11833	12706	12655	12808
500	15139	15082	15212	16567	16535	16591	15486	15374	15628	16728	16634	16775
600	18425	18384	18462	20214	20015	20346	18880	18748	19035	20322	20199	20466
700	21444	21330	21570	23835	23771	23981	21986	21902	22178	23644	23552	23744
800	25014	24945	25065	27521	27386	27762	25779	25528	25929	27748	27679	27862
900	28065	28005	28106	31117	31036	31305	28896	28817	28953	31238	31043	31366
1000	31549	31492	31704	35019	34840	35141	32529	32448	32584	35243	35098	35468

略Prog2と比較する。次に、複数回連続的に頂点を移動する多重頂点移動の効果を測るために、ランダムに選択した頂点を左右交互に一度のみ移動する近傍を用いたProg3との比較を試みる。最後に局部的な段階で確率的要素を採用しSimulated Annealing法を構成した本算法の性能を示すために、解そのもののコストを基準として確率的採択を行う基本的なSimulated Annealing法を構成し比較する。これをProg4とする。以上の戦略により求まる複数のコストを平均し、解の質を判定する。このときの各戦略の計算時間はほぼ等しい結果である。

表７．６の実験結果より、Prog1の優越性は明らかであり、他の近傍移動、および基本的なSimulated Annealing法の採用に対して、今回提案する確率的複合移動の本問題に対する効果が示された。

さらに、頂点数400のランダムグラフを対象とし、提案算法の受理率とコストの温度（Tmp）に対する変化の様子を図７．１２、７．１３に表す。この曲線は基本的なSimulated Annealing法の変化[1],[2]と同様な傾向を示しており、提案算法がSimulated Annealing法の特性に従って挙動していることを示している。

７．２．３　Tabu Search法との比較

次に、この確率的近傍移動を用いたSimulated Annealing法にもとづく算法が他の戦略より優れた結果を導くことを実験的結果より示す。Simulated Annealing法は近傍構造をベースとする近似解法である。これと同様に近傍構造をベースとする近似解法としてTabu Search法[7],[8],[9],[13],[16]があり、昨今、種々の問題で優れた効果を示している[24],[25],[26]。本問題に対しても今回採用した複合移動を用いて効果的な近似解が得られている[21]。ただし、この場合、頂点の移動

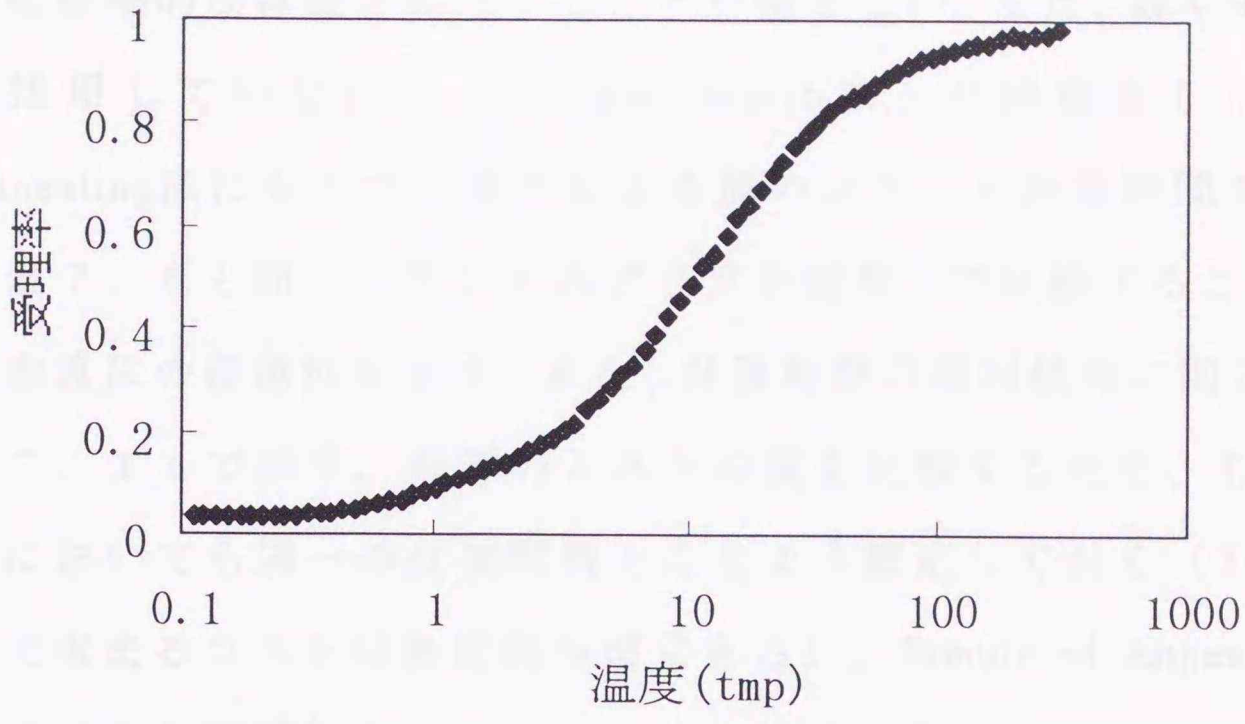


図７．１２　受理率と温度

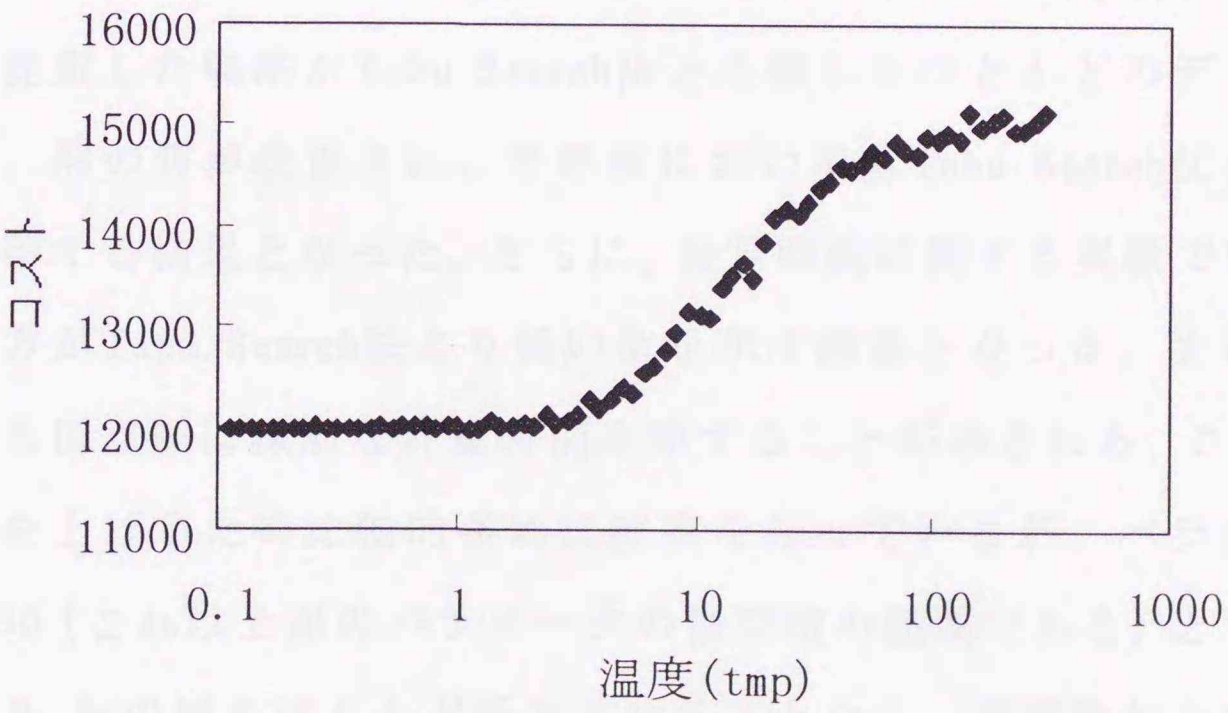


図７．１３　コストと温度

は短距離的な移動を採用することが望ましい。また、確率的要素は当然採用していない。このTabu Search法と今回提案したSimulated Annealing法にもとづく解法による解のコストと計算時間を表７．７（表７．６と同一なランダムグラフを使用）で比較することにより、提案算法の優越性を示す。また、計算時間の増加傾向に関しては特に図７．１４で示す。両者のコストの質を比較するため、Tabu Search法においても同一の反復回数となるよう設定しておく（Tabu Search法で求まるコストは決定的な値である）。Simulated Annealing法による値は１０回導出されたコストの平均である。また、その中での最良値と最悪値を付記しておく。ただし、ここで用いられるパラメーターは上記の推奨値にもとづき、tmp=40,stoptmp=2.0,r=100,phi=0.9,tau=1.1とし、両者の反復数を14510回と一致させる。計算環境は小樽商科大学情報処理センターSun SparcStation 2を採用している。

表７．７より、本問題に対してSimulated Annealing法にもとづく今回提案した戦略がTabu Search法と比較したほとんどのデータにおいて、解の質が改善され、最悪値においてもTabu Search法のコストを改善する結果となった。さらに、計算時間に関する実験では提案算法の方がTabu Search法より低い値を示す結果となった。また、両算法とも頂点数に線形な計算時間を要することが示される。この場合、精度を上げるため比較的多めに反復を行っているが、パラメーターをr=50（これは上記のパラメータの推奨値の範囲である）とすることにより、解の質をほとんど低下させることなく、反復数および計算時間をほぼ1/2程度に下げることが可能である。その他のランダムグラフにおいても以上の傾向が示され、提案算法の有効性が示された。ただし、並列構造を有するグラフに対してはTabu Search法の優位が示さ

れる傾向もある。

表７．７　Simulated Annealing 法と Tabu Search 法の比較実験

problem	Simulated Annealing				Tabu Search	
size	mean	best	worst	time(sec)	cost	time(sec)
100	2460	2431	2480	49	2451	60
200	5632	5621	5594	106	5688	129
300	8755	8680	8796	168	8909	195
400	11508	11370	11637	231	11889	268
500	15139	15082	15212	301	15583	346
600	18425	18384	18462	362	18820	403
700	21444	21330	21570	438	22198	475
800	25014	24945	25065	497	25867	547
900	28065	28005	28106	585	29168	619
1000	31549	31492	31704	658	33087	705

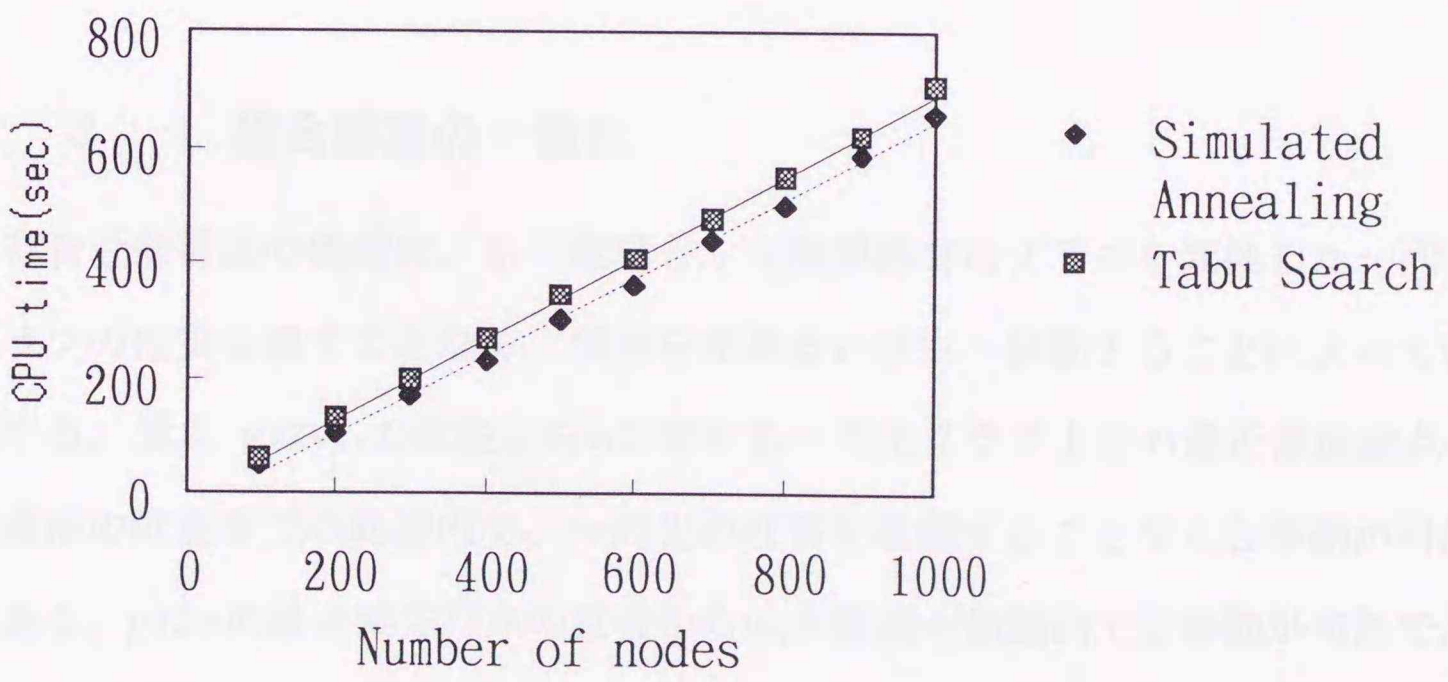


図７．１４　頂点数と計算時間

7．3 複合移動のメタ戦略に対する一般化

Tabu Search法にしても、Simulated Annealing法にしても近傍構造の構成を基本とする探索法である。グラフ分割問題では通常、単純な頂点移動によって効果的な近傍構造を構築できる。しかし、本問題はその特徴として、系列性を保存する多分割問題であり、また、解の成分集合の個数、各成分集合の要素数が不定であり、次の解への最良移動の決定は最良分割数の決定と合わせて複雑な組合せ探索を必要とするため、効果的な近傍構造が構成しにくい。これに対して多重的な頂点移動により解に大きな変化をほどこすことによって多様性を持たせ、部分的な最適化を導入することにより急速な収束を行う複合移動の考えが、Tabu Search法およびSimulated Annealing法の両者のメタ戦略に対して効果が得られたことは前章で示された。そこで、本章ではこの複合移動を両者のメタ戦略の構造にそれぞれ適合可能となるよう一般化の記述を試みる。これによって、広くメタ戦略に適合可能となり、複合移動の応用領域が広がるものと考えられる。

7．3．1 複合移動の一般化

複合近傍構造の構成は、第一段階として無閉路有向グラフを変換した一列化グラフの性質を壊すことなく、頂点を左あるいは右へ移動することによって構築する。頂点 v_iはv_{i+1}の直後からv_iに対する一列化グラフ上での最近接後続点v_mの直前の位置までの範囲内で、一列化の性質を破壊することなく右移動が可能である。v_iはv_iの最近接先行点の直後からv_{i-1}の直前の範囲内で左移動が可能であることはすでに述べた。以上のように、一列化グラフ上の任意の頂点はその最近接後続（先行）点によって定まる範囲内で、任意の場所に移動可能である。この移動場所の決定は各解法が採用するposition rule (*pr*)にしたがい、その問題と

解法の特徴によって決定される。頂点vの一列化グラフ上でのprにしたがう右側への頂点移動、およびvの左側への頂点移動を手続きright-move(v ; pr)、left-move(v ; pr)で表す。

以上の頂点移動をベースとし、さらに順次連続するブロックに適用し非実行可能解の範囲まで探索を広げる多重的な近傍移動を構築する。これにより、解に大きな変化をほどこしより強く最良な解の方向へ移動させ近似度の改善を行う。まず、探索過程の任意の段階での解$x=(V_1,V_2,\ldots,V_k)$はある一列化グラフDの頂点の並びと、ブレイク・ポイントの集合$\{b_1,b_2,\cdots,b_k\}$によって表現される。このとき、ブロック$V_i=[b_i,b_{i+1})$中から各解法が採用するnode-selection rule (nsr)にしたがって移動可能な頂点が一つ選ばれる。このnsrによってブロックV_iから選択する頂点を選びだす関数を構成する。以下にnsrを表す右側頂点移動の関数について記述する。

$$\overrightarrow{nsr}(V_i)=\begin{cases} v_i\ (\in V_i) & if\ \ there\ exit\ \exists\ node\ v_i\ \ selected\ by\ nsr \\ nil & otherwize \end{cases}$$

同様に、左側移動の場合も$\overleftarrow{nsr}(V_i)$として定義できる。ただし、V_i中に移動可能な頂点が存在しないとき、関数の値はnilを返すこととする。以上の２つの関数はnsrを与えることにより定まるものである。

以上のルールをもとに多重的な頂点移動の近傍を構成する。まず、任意の探索過程での解$x=(V_1,V_2,\ldots,V_k)$はある一列化グラフDの頂点の並びと、ブレイク・ポイントの集合$\{b_1,b_2,\cdots,b_k\}$によって表現される。このとき、ルールnsrとprによって定まるright-move移動の系列

right-move($\overrightarrow{nsr}(V_1)$;$pr$), right-move($\overrightarrow{nsr}(V_2)$;$pr$) ..., right-meve($\overrightarrow{nsr}(V_{k-1})$,$pr$)

を、出発解$x=(V_1,V_2,\ldots,V_k)$から始めて連続適用し、解$x'=(V_1',V_2',\cdots,V_k')$を得る。こ

こで、$\overrightarrow{nsr}(V_i)$はルール*nsr*によって誘導される関数である。また、right-move(*nil* ,*pr*)、left-move(*nil* ,*pr*)の場合は頂点の移動を行わない。この際、ブレイク・ポイントの移動は考慮しないので、ブロックの容量制約は無視して頂点移動を行うことになる。この頂点の連続した移動を多重right-move移動と呼び、

$$x' = \overrightarrow{multiN}(x; nsr, pr)$$

で表す。この結果に対して、再び、*nsr*と*pr*によって定義されたleft-move移動の系列

left-move($\overleftarrow{nsr}(V_k'); pr$), left-move($\overleftarrow{nsr}(V_{k-1}'); pr$), ..., left-move($\overleftarrow{nsr}(V_2'); pr$)

を連続適用することにより解の改善を行う。これを多重left-move移動と呼ぶこととし、同様に、

$$x'' = \overleftarrow{multiN}(x'; nsr, pr)$$

で表す。この頂点の多重的移動により解に大きな変化をほどこし解の移動に多様性をもたせる。ただし、この段階では解の実行可能性が失われるばかりでなく、成分集合の個数、大きさに大きな変化をもたらすことは望めない。

次に多様性による探索の拡大に対して、部分問題に対する最適化を適用することにより良質な解への収束性を加味する。すなわち、多重頂点移動によって得られた一列化グラフに対して容量制約を満たす最良なブロック分割を求める部分的な最適化を組み入れる。これによって、自動的に最良な分割数が決定されるとともに実行可能性への回復が計られる。この処理は前章で述べたように、一列化グラフの最適系列分割問題を解く算法[6],[17]をそのまま利用して実現でき、それを関数$dp(x'')$で表わす。これにより、与えられた一列化グラフのもとで最適なブレイク・ポイント列$\{b_1', b_2', \cdots, b_{k'}'\}$を見出す。その結果、頂点のブロック間での移動によるコストの改善と、実行可能性の回復が行われる。

これらの処理を一反復過程中において、

$$x''' = dp(\overleftarrow{multiN}(\overrightarrow{multiN}(x; nsr, pr); nsr, pr))$$

の順序で行い、良好な近似効果を持つ複合移動を達成する。上式が複合移動の一般化式であり、各手法に対して効果的なパラメータを導入することにより有効な複合移動が構成される。

７．３．２ Tabu Search 法に対する近傍構造の一般化

この一般化された複合移動に対して各解法に有効なルールを記述することによい、それぞれの近傍構造が表現できる。Tabu Search 法の場合、まず、*pr* は移動対象とする頂点を、その頂点を含むブロックに対して一列化グラフ上で隣接するブロックへ移動する *next_pr* を採用する。

次に、 *nsr* とそれを表現する関数 $\overrightarrow{nsr}(V_i)$ 、$\overleftarrow{nsr}(V_i)$ を以下のように定義する。*nsr* は使用するタブーリストをパラメーターとして含むルール *tabu_nsr*(*tabu-list*)と記述し、そのルールを表す関数を $\overrightarrow{tabu\_nsr}(V_i, \text{tabu-to-right})$、$\overleftarrow{tabu_nsr}(V_i, \text{tabu-to-left})$で記述する。図７．１５に関数 $\overrightarrow{tabu\_nsr}(V_i, tabu-to-right)$ のアルゴリズムを表す。

左側への移動の場合も $\bar{\delta}(v)$ にもとずき、$\overleftarrow{tabu_nsr}\ (V_i, tabu\text{-}to\text{-}left)$として同様に定義される。

この２つのルールを設定することにより Tabu Search 法の複合近傍構造が決定される。一般化された複合移動のもとでは以下のような過程の記述で表現されることとなる。

$$x' := \overrightarrow{multiN}(x; tabu\_nsr(tabu\text{-}to\text{-}right), next\_pr);$$

$$x'' := \overleftarrow{multiN}(x'; tabu\_nsr(tabu\text{-}to\text{-}left), next\_pr);$$

$$x''' := dp(x'');$$

以上を Tabu Search 法のアルゴリズムに適用することにより効果的な近似解法が構築される。

function $\overrightarrow{tabu\_nsr}(V_i, tabu\text{-}to\text{-}right)$

begin

if $\{V_i$ - *tabu-to-right*$\} \neq \varnothing$ begin

$\vec{\delta}(v_0) = \min_v \{\vec{\delta}(v) | v \in \{V_i - tabu\text{-}to\text{-}right\}\}$;

return v_0;

end else

return *nil*;

end

図７．１５ 関数 $\overrightarrow{tabu_nsr}(V_i, tabu\text{-}to\text{-}right)$のアルゴリズム

７．３．３ Simulated Annealing 法に対する近傍構造の一般化

まず、*pr* は右移動の場合、最近接後続点の直前へ、左側移動の場合、最近接先行点の直後へ移動する。すなわち、一列化グラフ上でできうる限り遠くへ頂点を移動する方策をとる。これを *long_pr* とする。

次に、確率的要素を組み込んだ近傍構造である確率的複合移動の*nsr*の記述を示す。*nsr*はパラメータ*tmp*を受け取る*rand_nsr*(*tmp*)とし、このルールを表す関数を $\overrightarrow{rand\_nsr}(V_i, tmp)$ 、 $\overleftarrow{rand\_nsr}(V_i, tmp)$ で記述する。図７．１６に関数 $\overrightarrow{rand\_nsr}(V_i, tmp)$ のアルゴリズムを示す。

左側への移動も同様に定義され、２つのルールにより Simulated Annealing 法の確率的複合近傍構造が構成される。一般化された確率的複合移動のもとでは以下のような過程の記述で表現されることとなる。

$$x' = \overrightarrow{MultiN}(x; rand\_nsr(tmp), long\_pr);$$

$$x'' = \overleftarrow{MultiN}(x'; rand\_nsr(tmp), long\_pr);$$

$x''' := dp(x'')$;

以上の確率的複合移動を Simulated Annealing 法 のアルゴリズムに適用することにより効果的な近似解法が構築される。

```
function rand_nsr(V_i, tmp)
begin
    v = V_iの中からランダムに移動可能な頂点を一つ選択する。;
    if  δ+(v) <= 0  then
        return v;
    else
        if exp(-δ+(v)/tmp) >= random[0,1) then
            return v;
        else
            return nil
end;
```

図７．１６ 関数$\overrightarrow{rand_nsr}(V_i, tmp)$のアルゴリズム

7．4 まとめ

本章ではまず、無閉路有向グラフの系列分割問題に対するTabu Search法の適用とその有効性について検討している。本問題の近傍構造は基本的には$e(v;V_i,V_j)$、$a(v;V_i,V_j)$で実現できる。しかし、本問題は系列性を保持した多分割問題であり、成分集合の個数、各成分集合の要素数は不定である複雑なグラフ分割問題であるため、$e(v;V_i,V_j)$、$a(v;V_i,V_j)$の単純な適用では計算時間、近似度ともにより良い結果は望めない。これに対して、本章では一列化グラフとブレイク・ポイントの集合を用いたデータ構造を利用し、近傍移動の設計では本問題の構成に合わせて変形したleft-to-right移動、およびright-to-left移動を連続的に適用し、さらに局部的な最適化を取り入れ、複合的な処理方法を用い効果的な複合近傍移動を実現した。従来の近傍移動に対して、この複合近傍移動の採用により解に大きな変化をほどこしTabu Search法の性能をより強く引き出し、その特徴も維持する結果となった。さらに、問題独自のヒューリスティックな知識を取り入れることによって、より近似度の高い算法が実現できた。また、今回の数値実験より多くの知見を得ることができた。タブーリストの長さの適正値に関しては本問題の場合、ブロックサイズの大きさに強く依存する結果を得た。次に、ヒューリスティックな戦略を用いた場合の効果を示し、同時に解の改善の性質を明らかにした。本実験から、タブーリストの適正値、および長期メモリーの効果など、２分割問題では示されていない性質、および異なる性質などの知見を得たことは以後の研究にとって興味深い。さらに、グラフが並列的な構造をもつとき、相対誤差5%以内の解（最適解が求まる場合もある）が求まりその有効性が示された。また、その計算時間もグラフの構造などに影響されることなく、ほぼ頂点数に線形に増加する傾向が示された。しかし、ランダムな構造をもつグラフに対してはTabu Search法は局所的な解に落ち込み、それ以上の改善が望めない減少も示された。

この問題を解決するために、次に確率的要素を複合移動に導入したSimulated Annealing法の適用を検討し、その有効性について論じた。同様に、成分集合の個数、各成分集合の要素数は不定である複雑なグラフ分割問題のため、通常の近傍移動のみでは近似度のより良い結果は望めないため、Tabu Search法と同様に複合移動を採用した。しかし、近傍移動の設計では確立的要素を組込み、Simulated Annealing法の構成に合わせて変形したleft-to-right移動、およびright-to-left移動を多重的に適用する。また、より計算しやすい採択基準を用いた局部的なコスト差による確率的採択を採用し効果を計っている。従来の近傍移動に対して、この確率的複合移動の採用により解に大きな変化をほどこしSimulated Annealing法の性能をより強く引き出し、その特徴も維持する結果となった。

今回考案した複合移動の本問題への効果と、Tabu Search法による解法との比較によって本算法の性能を数値実験により明らかにした。それによると、確率的複合移動の有効性が示されると同時に、ランダムグラフの場合、提案算法はTabu Search法により求まる解より良質な解が導かれ、Tabu Search法の問題点の改善がなされた。ただし、並列構造を有するグラフに対してはTabu Search法の優位が示される傾向もある。また、その計算時間もグラフの構造などに影響されることなく、ほぼ頂点数に線形に増加する傾向が示され、Tabu Search法に対して多少良好な値が得られる結果となった。

以上のように、メタ戦略にとって近傍構造は良質な解の導出と計算時間を左右する重要な枠組みである。これによって、その解法の性能が決定されると言っても過言ではない。本問題に対するTabu Search法、およびSimulated Annealing法にしても、多重的な頂点移動により解に大きな変化をほどこすことによって

多様性を持たせ、部分的な最適化を導入することにより急速な収束を行う複合移動により有効なメタ戦略が構成できた。それによって、この複合移動の考え方がメタ戦略に共通して有効であることが示され、複合移動の応用性が期待される。

そこで、今回有効であった複合移動の概念を各種のルールをパラメータ化することにより一般化し、複合移動の考えを普遍化するとともに、応用の領域を広げる可能性をもたせる。さらに、その一般化された複合移動の表現を用い、Tabu Search 法、および Simulated Annealing 法に代表されるメタ戦略への適合を示し、一般化された複合移動の表現で本問題に対するメタ戦略が表現可能であることを表し、十分に複合移動の普遍化された考えが適用可能であることを述べる。今後は、この複合移動の考えが多くのメタ戦略に応用可能であることを明らかにし、その可能性を追求していきたい。

最後に、この複合移動の中で用いられた部分的最適化の問題は一列化の最適系列分割問題の解法を用いており、それが多数回反復使用されることにより、第５章で述べた算法の改良が以上のメタ戦略の計算時間の改善に寄与することを述べておく。

参考文献

[1] Aarts,E., Korst,J. and Laarhoven,P. : A Quantitative Analysis of The Simulated Annealing Algorithm: A Case Study for The Traveling Salesman Problem, Journ. of Statistical Physics, Vol.50,pp189-206(1988).

[2] Aarts,E. and Korst,J. : Simulated Annealing and Boltzmann Machines, John Wiley & Sons(1989).

[3] Betts,j. and Mahmoud,k.I.: A Method for Assembly Line Balancing, Engineering Costs and Production Economics, Vol.18,pp.55-64(1989).

[4] Drexl,A. : A Simulated Annealing Approach to The Multiconstraint Zero-One Knapsack Problem, Computing, Vol.40,pp.1-8(1988).

[5] Johnson,D., Aragon,C., Mcgeoch,L. and Schevon, C. : Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning, Operations Research,Vol.37,No.6,pp.865-892(1989).

[6] Kernighan,B.W. : Optimal Sequential Partitions of Graphs, J.ACM, Vol.18, No.1, pp.34-40(1971).

[7] Glover,F. : Tabu Search Part I, ORSA J.C.,Vol.1,No.3, pp.190- 206(1989).

[8] Glover, F. : Tabu Search Part II , ORSA J.C.,Vol.2,No.1,pp.4-32(1990).

[9] Glover,F. : A User's Guide to Tabu Search, Annals of Operations Research,41,pp.3-28(1993).

[10] kaji, T. : Simulated Annealing Algorithm for the Sequential Partitioning Problem of Directed Acyclic Graphs, The Economic Review,Otaru University of Commerce, Vol.1, No.1, pp.1-2(1996).

[11] Kaji,T. and Ohuchi,A. : A Tabu Search Approach to the Optimal Sequential Partitions of Directed Acyclic Graph, The Electrical Engineering in Japan (in

printing).

[12]Kaji,T. and Ohuchi,A. :The Assignment Problem for System with Precedence Relationships among Elements using a Simulated Annealing Algorithm, Proc. of the 14th International Conference on Production Research, Ohsaka (to appear).

[13]Reeves,C.R: Modern Heuristic Techniques for Combinatorial Problems, BlackWell(1993).

[14]Salveson,M.E.: The Assembly Line Balancing Problem, The Journal of Industrial Engineering, May-June,pp.18-25(1955).

[15]茨木：組合せ最適化, 産業図書(1983)

[16]茨木：離散最適化法とアルゴリズム（岩波講座 応用数学）、岩波書店(1993).

[17]加地，大内：最適系列分割問題に対する効率的分枝限定法の構築と諸特性解析，情報処理学会論文誌，Vol.35, No.3, pp.364-372(1994).

[18]加地：Ｔａｂｕ　Ｓｅａｒｃｈによる無閉路有向グラフの最適系列分割問題と特性評価, 情報処理学会研究報告（数理モデルと問題解決研究会），Vol.95,No.66, pp.1-6(1995).

[19]加地 ： Tabu Searchを用いた無閉路有向グラフ系列分割問題の近似解法, 商学討究（小樽商科大学）, Vol.46, No.1, pp.125-141(1995).

[20]加地 ： 要素間に先行順位関係をもつシステムの配置問題とメタ戦略, 情報処理学会シンポジウム（問題発見とモデル化）, Vol.96, No.12, pp.1-7(1996).

[21]加地, 大内 ： Tabu Search法による無閉路有向グラフの最適系列分割問題の解法, 電学論ｃ, Vol.116-C, No.10, pp.1149-1157(1996).

[22]加地、大内： 要素間に先行順位をもつシステムの配置問題, 電気学会論文誌Ｃ分冊， Vol.117-C, No.2, pp.136-142(1997).

[23]久保 ： 巡回セールスマン問題への招待, 日本OR,Vol.39,No.3,pp.156-

162(1994).

[24]久保 ：モダンヒューリスティックスの新展開－Gnetic Algorithm,Simulated Annealing,Tabu Search,Neural Net 法は本当に有効か－,日本 OR 学会第 30 回シンポジウム(1994).

[25]久保 ：Local Search から Simulated Annealing,Tabu Search へ, モダンヒューリスティックス（平成 6 年度第 2 回 OR セミナー）,pp.1-32(1994).

[26]久保 ：メタヒューリスティックス, 離散構造とアルゴリズムIV, 近代科学社, pp.171-230(1995).

[27]西関、秋山 ：グラフとダイグラフの理論, 共立出版(1981).

[28]藤沢、久保、森戸 ：Tabu Search のグラフ分割問題への適用と実験的解析, 電学論 c ,Vol.114,No.4,pp.430-437(1994).

第８章　結論

本論文は、組合せ最適化問題の中で最も基本的かつ応用範囲の広いグラフ分割問題の一形態である無閉路有向グラフの系列分割問題の確立とその効果的解法の研究および評価等について行ったものである。本研究により、より汎用的な問題に対応できる無閉路有向グラフの系列グラフ分割問題を提案することにより、より広範な問題へと展開を試みた。この問題の基本となる一列化グラフとして限定されたKernighanの問題に対して詳細な解析と検討を行うことにより、新たに改善したアルゴリズムを提案し、無閉路有向グラフの系列分割問題の有効な構成要素を構築した。しかし、無閉路有向グラフの最適系列分割問題の解を求めるには計算時間が指数的オーダーとなる課題が残されていた。それに対して、限定された無閉路有向グラフに対して実用的な時間内で計算可能な厳密解法を提案するとともに、ランダムなグラフに対しても有効な近似解法を考案することによりその課題の解決に導いた。

８．１　各章の要約

８．１．１　第１章

第１章は序章であり、グラフ分割問題と最適化問題に対するこれまでの研究について総括的な説明を行った。そして、本研究の目的と意義を明らかにし、本論文の構成について述べるとともに、その概略を示した。

８．１．２　第２章

本研究の基本となる組合せ最適化問題に対する各種の解法について概要を説

明することにより、論文で必要とする基本的知識を総括した。特に、動的計画法と分枝限定法ではその計算手順について数理的に明確にした。また、多くのヒューリスティックな知識を統合化したメタヒューリスティックの代表的手法である Tabu Search 法と Simulated Annealing 法が近傍構造を土台とし構築され、アルゴリズムの効率が近傍構造に依存することを述べ、その基本的概念を与え、両者のアルゴリズムの考え、効率的な構成法について述べた。また、Simulated Annealing 法に関しては、漸近的に大域的最適解に収束することを数理的に説明している。

８．１．３ 第３章

本研究の出発点となり、さらに、重要な構成要素である一列化グラフの最適系列分割問題とは、連続的な頂点番号を保持した一列化グラフに対するきわめて特定化されたグラフに対する多分割問題である。このような特定化されたグラフ構造を利用してKernighanはエッジ数に線形な計算量をもつアルゴリズムを開発した。これらの Kernighan の研究について概括し、本研究への起点とする。

８．１．４ 第４章

第４章ではKernighanの一列化グラフの最適系列分割問題をより一般化した無閉路有向グラフの最適系列分割問題に拡張し、先行順位のある要素をその先行順位の制限を無視することなく、いくつかのグループに配置する問題を考えた。これらの問題は無閉路有向グラフの頂点を先行順位の関係を無視せずに各成分に配置する基本的分割問題の範疇に属するものと考えられる。この基本的分割問題を無閉路有向グラフの系列分割問題として提案し、その問題の特性を数理的に論証した。さらに、系列分割問題の一般的定式化と、その実例として無閉

路有向グラフを系列分割するときに生ずるカット・エッジのコストの総和を最小化する問題を取り扱い、この問題を“総カット値を最小化する系列分割問題”と定義した。この問題の提案により、より多くの汎用的なシステム構造への応用へと展開される可能性が開けた。

8．1．5 第5章

第5章は本問題の重要な構成要素であるKernighanの一列化グラフの最適系列分割問題の解法に対して理論的、実験的な特性評価を行い詳細な検討を示し、Kernighan の研究で不足している評価検討を補い、その特性を明らかにした。それによると、ブロックサイズが変動する場合や、エッジ数が疎および密である場合の検討が加えられ、さらに、アルゴリズムの構成上、探索法や限定操作の観点から改善が可能であるものと判断された。それに対して、その点を考慮し、動的計画法と多くの共通点をもち、柔軟に各戦略を組込みやすい分枝限定法の適用を考え、特に探索法では最良下界探索法と、限定操作では細分化禁止則と下界値を用いることにより効率的なアルゴリズムを考案した。そして、その数値実験を行って検証することにより、無閉路有向グラフの最適系列分割問題解法の有効な構成要素となりうるアルゴリズムであることを示した。

8．1．6 第6章

第6章では、本来実用時間内で計算困難な無閉路有向グラフの系列グラフ分割問題に対して、グラフに並列構造を有するとき実用時間で計算可能な厳密解法を提案した。まず、最少データ表現として既約グラフを用い、その上で、高速に展開可能な基本操作を考案し横型展開法により、効率よく探索を形成可能であることを明らかにし、その上で、動的計画法を用い効率的なアルゴリズム

を構築した。そして、数値実験により、要素間に先行順位をもつシステムにおいて並列構造が顕著に認められるとき、この算法が十分実用に耐えることを明らかにした。

8．1．7 第7章

第7章では、ランダムな無閉路有向グラフに対しても、実用時間内で計算可能な解を求める目的のために、従来の組合わせ最適化問題を解くための種々の戦略を有機的に結合させたり、あるいは反復させることにより、従来の近似解法を超えたパラダイムとして注目を浴びているメタヒューリスティックによって近似解を求めている。まず、メタヒューリスティックの中のTabu Search法の適用を試みている。この無閉路有向グラフの系列分割問題では、その特徴として、解の成分集合の個数、各成分集合の要素数がともに不定であり、これが、解の近傍移動などの処理を複雑にし、標準的なTabu Search法による解法の構成を難しくしている。そこで、近傍移動の設計では本問題の構成に合わせて変形した頂点のleft-to-right移動、およびright-to-left移動を多重的に適用し、さらに部分的な最適化を取り入れ、複合的な処理方法を用い効果的な近傍移動を実現し、効率的なTabu Search法によるアルゴリズムを提案している。さらに、Tabu Search法の局所解に落ち込みやすい欠点を補うため、確率的な要素を取り入れた複合移動を用いSimulated Annealing法によるアルゴリズムを提案した。そして、それらの効果を数値実験で検証し、その結果はSimulated Annealing法がTabu Search法により求まる解よりも良質な解を導き、それらの計算時間もグラフの構造などに影響されることなく、ほぼ頂点数に線形に増加する傾向が示した。また、今回両アルゴリズムに有効であった複合移動の概念を各種のルールをパラメータ化すること

により一般化し、各メタヒューリスティックへの適用を示した。

8．2 総括

本論文における検討課題をまとめると以下のようになる。

(1) 限定された一列化グラフに対して、多くのシステム構造はより一般的な無閉路有向グラフで構成されており、より、汎用的で、応用性の広い問題を扱うために、その上での系列分割問題について考察する必要があるが、それに関する研究は従来行われていない。

(2) (1)の重要な構成要素となる一列化グラフの系列分割問に対して、Kernighanによって効果的アルゴリズムが提案されている。しかしその中ではアルゴリズム全体の詳細な計算量、および、その特性については大きく検討されておらず、その解法の特性が明確でない。

(3) 動的計画法の最適性の原理と、分枝限定法における優越関係には大きな共通点があることが知られている。分枝限定法で下界値テストは行わず、探索法として順位を適当に選んだ横型探索法を用い、優越関係による限定操作のみを用いるとき、この分枝限定法は動的計画法の手順と等価になる。したがってこの立場では動的計画法と分枝限定法との間には本質的な差はないといってよく、動的計画法の計算手順は分枝限定法の一種とみなせる。しかし動的計画法をこのように分枝限定法とみなすと、必ずしもKernighanの算法は最良の探索法とはいえない横型探索法を用いていること、および下界値等による限定操作を全く用いていないという点からさらに改善の余地がある。この視点から標準的な動的計画法の上に組み立てられたKernighanの算法も改善可能と考えられるが、その観点からの研究は未だ行われていない。

(4) (1)に対する問題として無閉路有向グラフの系列分割問題が提案されるが、それに関して有効な厳密解法は提案されておらず、また、複雑な集合演算が

予想されるとともに、解の展開操作の高速化など困難な課題が予想される。

(5) 無閉路有向グラフの系列分割問題は特定なグラフに対して効果的な厳密解法の構築は可能であるが、多くのグラフに対しては実用的な計算時間内では計算困難な問題となり、それに対して、多くはヒューリスティックな知識を用い、近似解をとる戦略を用いるが、この問題に対しての効果的な近似解法は存在しない。

(6) Tabu Search法やSimulated Annealing法に代表されるメタヒューリスティックによる近似解法の研究が近年盛んに行われているが、これらは近傍の構造によってそのアルゴリズムの性能が左右される。しかし無閉路有向グラフの系列分割問題の場合、解の成分集合の個数、各成分集合の要素数がともに不定であり、これが、解の近傍移動などの処理を複雑にし、標準的な近傍移動により効果的な近似解を得られにくくしている。

以上の検討をもとに本論文では、まず、無閉路有向グラフの系列分割問題の確立に取り組み、重要な構成要素となるKernighanの一列化グラフの系列分割問題の解法に対する詳細な検討と、分枝限定法を用いた柔軟なアルゴリズムの改良を行った。次に、無閉路有向グラフの系列分割問題に対する解法を考案した。本研究において開発したアルゴリズムは

1. 一列化グラフの最適系列分割問題に対する効率的分枝限定法
2. 無閉路有向グラフの最適系列分割問題に対する動的計画法アルゴリズム
3. 無閉路有向グラフの最適系列分割問題に対する複合移動を用いたTabu Search法
4. 無閉路有向グラフの最適系列分割問題に対する確率的複合移動を用いたSimulated Annealing法

の４つであり、Kernighanによって提案されたアルゴリズムを含め、詳細に評価検討を行っている。また、本研究でより一般的な問題として無閉路有向グラフの系列分割問題を提案したことは応用問題への展開を含め有意義なことである。さらに、この問題に対して近代的な近似解法のパラダイムであるメタヒューリスティックを適用して選られた多くの知見は今後のメタヒューリスティックの研究にとって有用な資料を提供するものである。

本研究により開発された各種のアルゴリズムは有効なものと考えられるが、次に挙げる研究課題が残されていると考えられる。

- 並列構造をもつ大規模な無閉路有向グラフに対しての、より効果的な厳密解法の検討
- 無閉路有向グラフの最適系列問題に対して、さらに近似度の改良を考慮した近似アルゴリズムの検討
- 各種のメタヒューリスティック手法の統合化の検討

以上のような問題に対して、特に、現在近傍構造の改良を試みSimulated Annealing法の性能を改善する可能性が展開されつつある。

また、無閉路有向グラフの系列分割問題は、目的関数、制約条件を変化させることによって各種の問題へ適用可能であり、ライン・バランシングなどの応用研究への展開等も試みてみたい。

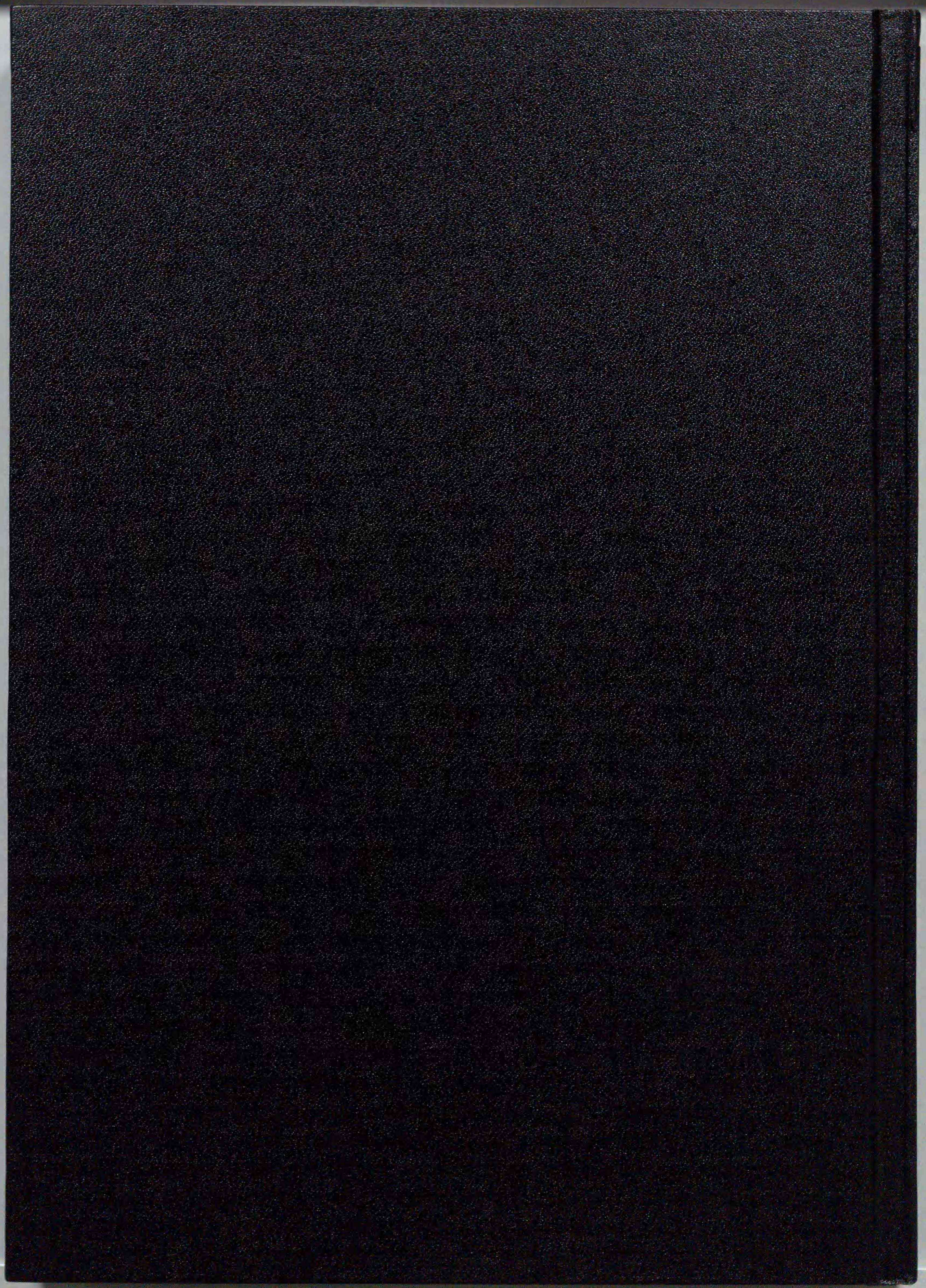
謝辞

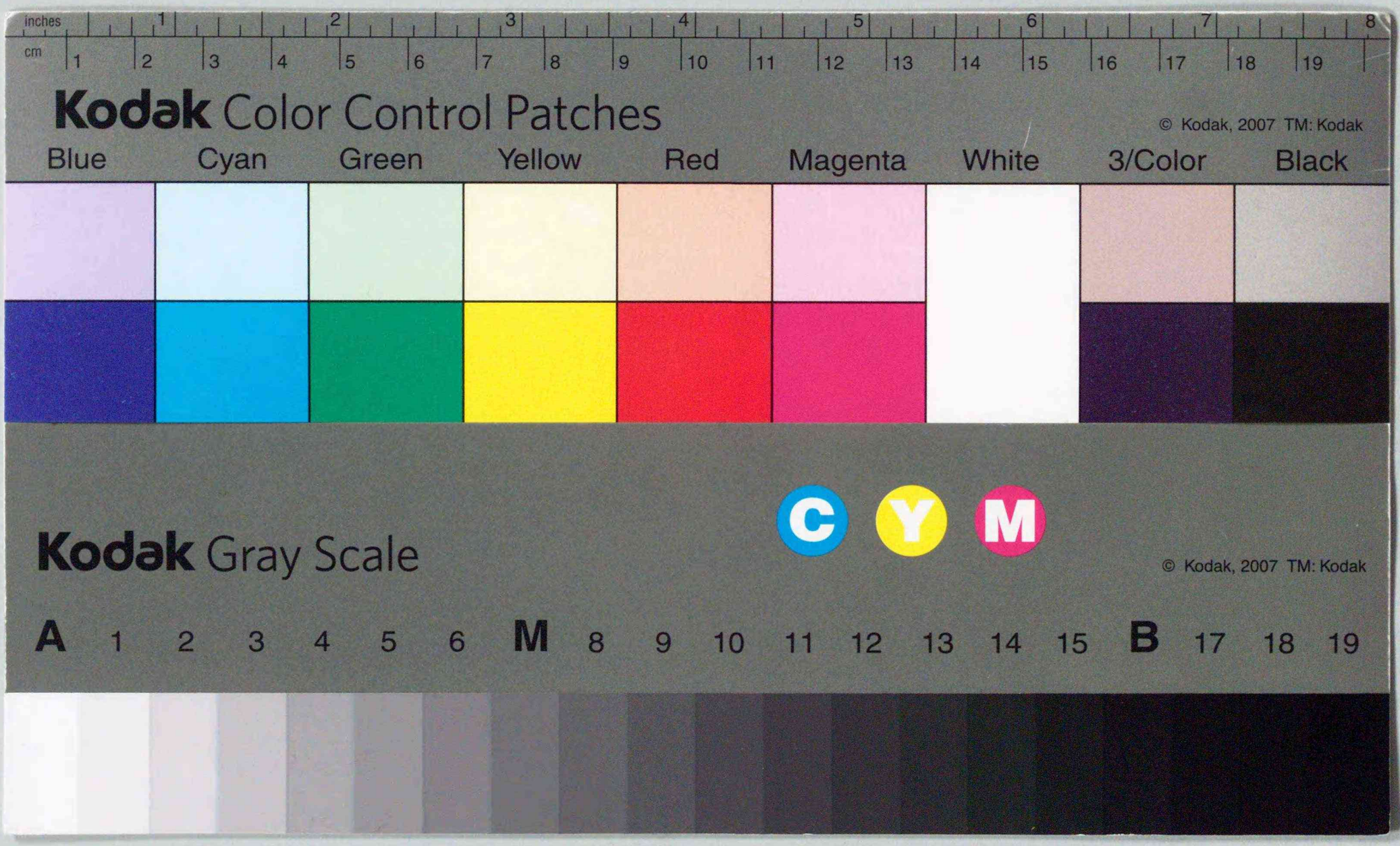
本研究は、著者が北海道情報大学経営情報学部情報学科助手、および小樽商科大学商学部社会情報学科助教授として在職中に、北海道大学大学院工学研究科システム情報工学専攻複雑系工学講座調和系工学分野大内　東教授のもとで行った研究をまとめたものであります。本研究の機会を与えてくださり、本論文の完成まで深い洞察力でご援助、ご指導をいただき、そして、広い御心で絶えず見守っていただいた大内　東教授に深謝いたします。

本研究に対して詳細な議論をいただきました北海道大学大学院工学研究科システム情報工学専攻複雑系工学講座表現系工学分野宮本　衛市教授、自律系工学分野嘉数　侑昇教授、ならびに混沌系工学分野和田　充雄教授に感謝いたします。

本研究を遂行するにあたり、貴重なご意見、御質問をいただいた複雑系工学講座調和系分野の諸先生、および大学院生の方々に感謝いたします。また、本研究に専心できるよう終始、数々のご援助をいただいた北海道情報大学、小樽商科大学の諸先生に感謝いたします。

最後に、学会、および研究会等で本研究に対する活発な御討議に参加していただいた数々の方々に感謝いたします。




inches
cm
Kodak Color Control Patches
© Kodak, 2007 TM: Kodak
Blue
Cyan
Green
Yellow
Red
Magenta
White
3/Color
Black
Kodak Gray Scale
C
Y
M
© Kodak, 2007 TM: Kodak
A 1 2 3 4 5 6 M 8 9 10 11 12 13 14 15 B 17 18 19