アルゴリズム 21－6
（1991. 5. 29）

# 分枝限定法による系列グラフ分割問題の高速化と拡張

加地太一　　大内　東
北海道情報大学　北海道大学

頂点が連続的な番号を保持し、始点が初期番号、終点が最終番号となり、両端点が異なるグラフを系列グラフG（V, E）と呼ぶ。本論文における最適系列グラフ分割は系列グラフに対して、各頂点に与えられた重みの総和がブロックサイズP（＞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和が最小となるよう分割する問題である。

この問題に対して著者等はこれまでに開発したBranch－and－Bound法（BB）を用いた手法に対して、新たな改良改善を試み、また、その効率の評価、および計算量を算出する。さらに新しい問題の拡張を試みる。

## An Improved Branch and Bound Algorithm for Sequential Partitions of Graphs and Related Problems

Taichi Kaji*　　Azuma Ohuchi**

* Department of Information Science,
Hokkaido Information University,
59-2, Nishinopporo, Ebetu, Hokkaido, Japan
** Department of Information Engineering,
Hokkaido University,
N-13, w-8, Sapporo, Hokkaido, Japan

Optimal sequential partitions of graphs is to find a minimum cost partition of the nodes of a graph into subsets of a given size, subjecting to the constraint that the sequence of the nodes may not be changed, that is, that the nodes in a subset must be consecutive numbers.

An improved Branch and Bound algorithm for the problem is proposed and its time complexity is discussed. Some varieties of the problems are also considered.

## 1．はじめに

頂点が連続的な番号を保持し、始点が初期番号、終点が最終番号となり、両端点が異なるグラフを系列グラフG（V，E）と呼ぶ。本論文における最適系列グラフ分割は系列グラフに対して、各頂点に与えられた重みの総和がブロックサイズP（＞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和が最小となるよう分割する問題である。

この問題に対してKernighan[1]は動的計画法（ＤＰ）を用いることによって効率的なアルゴリズムを開発している。また浅野[3]は区分木を用いたデータ構造を採用し、積極的な効率化を行っている。この問題に対して、著者等はこれまでに開発してきたＢｒａｎｃｈ－ａｎｄ－Ｂｏｕｎｄ法（ＢＢ）を用いた手法に対して、新たな改良改善を試み、また、その効率の評価、および計算量を算出する。さらに新しい問題の拡張を試みる。

## 2．問題の設定

ノード集合V＝｛1、2、・・・、n｝、エッジ集合E＝｛$e_1$、$e_2$、・・・、$e_r$｝からなる 無向グラフをG＝｛V、E｝とする。各エッジは非負のコストC＝｛$c_{i,j}$｜始点をi、終点をj（i＜j）としたエッジのコスト｝をもつ。この時Gの各ノードは｛$w_1$、$w_2$、・・・、$w_N$｝の重みをもち、各々$0 < w_i \leqq P$となる。ここで、Pはブロックサイズと呼ばれる正の数である。また、ノード集合Sの重さは$|S| = \Sigma_{i \in s} w_i$となる。

Gをk個のサブ集合$G_1$、$G_2$、・・・、$G_k$に、以下の制約のもとで、カットされるエッジのコストの総和が最小になるよう分割する。

（1）$|G_i|$＝部分集合のノードの重みの総和 ≦P

（2）任意の$G_i$の各ノード番号は連続的な数をもつ

部分集合$G_i$はブロックと呼ばれ$G_i$＝｛j，j＋1，・・・，m－1，m｝となる。$G_i$での一番小さなノードの名前を$b_i$として、ブレイクポイントと呼ぶこととする。$b_i$＝jはノードj－1とノードjの間でカットすることを意味する。 集合｛$b_1$，$b_2$，・・・、$b_k$｝は一意的に分割｛$G_1$，$G_2$，・・・，$G_k$｝を表現する。

ここで、グラフGが有向グラフであり、かつ（i，j）と（j，i）のエッジを含んでいるのなら、i＜jである単一エッジ（i，j）におきかえる。その時、コスト$c'_{i,j} = c_{i,j} + c_{j,i}$となる。

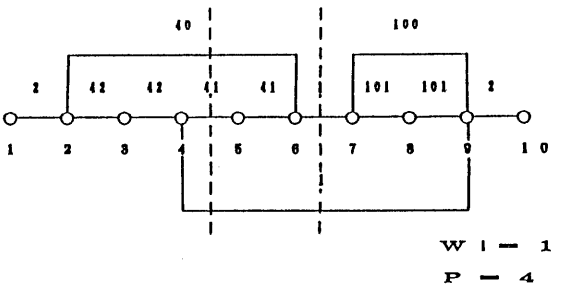


図1　系列分割グラフ

## 3．分枝限定法（ＢＢ）

分枝限定法は６つのパラメーター、分枝規則、探索規則、削除規則、優劣関係、下限値関数、上限値による分類的構成によってアルゴリズムを構築する。以上のパラメーターを基本とし分枝限定法によるアルゴリズムを作成、考案する。以下に系列分割問題の算法の概略を述べる。

```
procedure Branch and Bound
begin
  CurNode = 初期ノード;
  while not (CurNodeの下限値 < 現在の上限値
        and CurNodeの最終ブレイクポイント = N+1)
    ExpandSelect( CurNode );
    展開された子ノード集合を整列オープンリスト
    に格納する;
    CurNode = オープンリストの先頭要素;
    オープンリストの先頭要素を削除する;
  end
end;
最適解 = CurNode;

ExpandSelect( CurNode )
begin
  CurNodeに対して最終ブレイクポイントからブロッ
  クサイズ分の要素の増分コスト計算をする;
  while (CurNodeに対して分枝規則により子ノードS
        が分枝できる)
    if (レベル削除規則によって削除されない)
      if (上限値によって削除されない)
        子ノード集合にSを付加する;
end;
```

以上のような算法においての６つのパラメーター構成の詳細を記す。

（１）分枝規則（ブランチングルールＢ）

ブランチングルールは探索空間を広げていくために選ばれたノードを展開するための規則の集合である。派生過程において、いかに探索空間を小さくするかが問題である。ブランチングルールＢとしてはノード系列の初期値を開始点として次の２つの性質により順次展開していく。

性質１．各ブロックの大きさがＰを越えてはならない。

性質２．あるブロックを細分化すればコストへの寄与が大きくなる。ゆえに探索木の中で、あるブロックとそのブロックを細分化した状態が生じるとき、細分化した状態をコスト最小化の立場で無視してよい。

１）と２）の規則より次の不等式が導ける。

$BP_{i-1}+P \geqq BP_i > BP_{i-2}+P$

（ノードウェイト＝１のときの不等式）

$BP_i$ ： i番目のブレイポイント

Ｐ ： ブロックサイズ

以上の不等式にしたがってブランチングノードを生成する。

（２）選択規則（セレクションルールＳ）

セレクションルールＳは現在のアクティブノードから次のブランチングノードを選択するルールの集合である。このルールの設定によって探索の経路が決まり、いかに早く最適解に到達できるかを決定する。ここではこの問題に効率のよい最良下限探索を採用する。

最良下限探索は探索過程の現時点でコストの値が最も低いノードを選んで進んでいくバックトラック法である。その時点までに得られているノードの中から、最も目標に近いノードを選んで展開する。

（３）優越関係（ドミナンス・リレイションＤ）

部分解$\pi_1$と部分解$\pi_2$に対して、$\pi_1$、$\pi_2$を先祖とする完全解の中でコストが最小なものを比較し、２つのノードの優位性をみる。

もし ｍｉｎ｛$\pi_1$を先祖とする完全解のコスト集合｝＜ｍｉｎ｛$\pi_2$を先祖とする完全解のコスト集合｝ なら $\pi_1 D \pi_2$となる。この問題において以下の２つのドミナンスリレイションが成立する。

１）（部分解$\pi_k$）Ｄ（部分解$\pi_k$のブロック$G_i$を細分化した部分解）

２）最終ブレイクポイントの値が等しい部分解$\pi_a$、

$\pi_b$において、$\pi_a$のコストが$\pi_b$より低ければ

$\pi_a D \pi_b$

となる。劣性な部分解には最適解が存在しないので、削除が可能である。このドミナンスリレイションの設定がＢＢの効率の決定的要因となる。

（4）下限値関数Ｌ

下限値関数Ｌ（$\pi$）は部分解$\pi$に対して派生するすべての完全解に下限であるコストを表す関数である。

（5）上限値Ｕ

Ｕは今現在知られている完全解のなかで最良なコストの値である。

（6）削除規則（エルミネーションルールＥ）

Ｅは上限値やドミナンスリレイションを用いてアクティブノードや新しく生成されたインアクティブノードを除去するルールの集合である。探索過程において、Ｌ（$\pi_1$）＞Ｕまたは$\pi_2 D \pi_1$の場合、$\pi_1$を削除し、探索空間を縮小させる。

## ４．高速化のための問題点

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。これによって、スパース部への非参照およびコスト計算の効率化が行われる。

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

$C(x, y) = \Sigma_{y \le i < x} c_{ij}$を直接用いる方法と以下の漸化式

$C(x, y) = C(x-1, y)$

$\quad + (C_{x-1,x} + C_{x-1,x+1} + \cdots + C_{x-1,n})$

$\quad - (C_{y,x-1} + C_{y+1,x-1} + \cdots + C_{x-2,x-1})$

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。

$Out[1] = c_{1,2} + c_{1,3} + \cdots + c_{1,n}$

$Out[2] = c_{2,3} + c_{2,4} + \cdots + c_{2,n}$

・・・

$Out[i] = c_{i,i+1} + c_{i,i+2} \cdots + c_{i,n}$

・・・

$Out[n-1] = c_{n-1,n}$

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

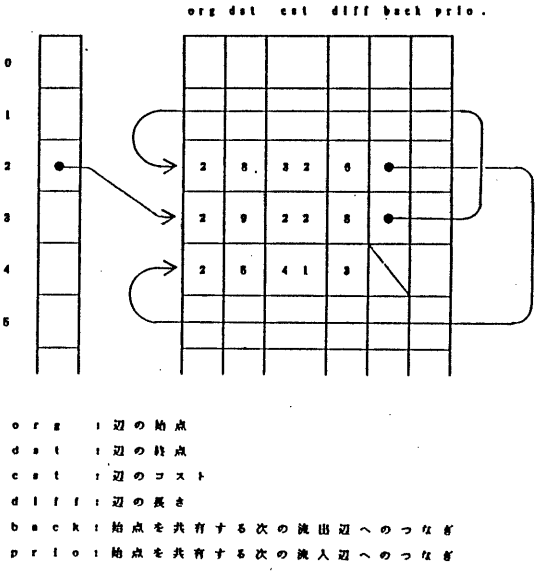


図2　グラフデータの表現

をＩｎ［ｉ，ｊ］に格納した表をつくる。図3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（４．１）式は

＝Ｃ（ｘ－１，ｙ）
　＋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（ｘ－１，ｙ）
　＋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。

$$In[1,2] \quad In[1,3] \quad In[1,4] \quad \cdots$$
$$In[2,3] \quad In[2,4]$$
$$In[3,4]$$
$$\cdots$$

$$In[1,2] = c_{1,2}$$
$$In[1,3] = c_{1,3}$$
$$In[2,3] = c_{1,3} + c_{2,3}$$
$$\cdots$$
$$In[1,j] = c_{1,j} + c_{2,j} + \cdots + c_{i,j}$$
$$\cdots$$

図3　Ｉｎ［ｉ，ｊ］タブロー

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4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。

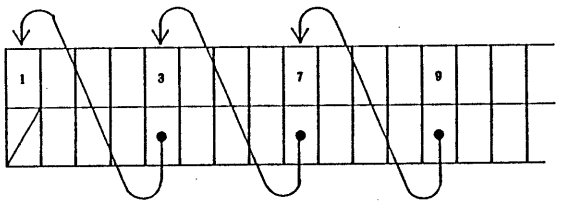


図4　探索空間のノード状態の表現

4）ＡＳリスト処理の高速化

展開されたノードをオープンリストに格納するために、下限値の低い順に双方向整列リストを構成する。図5に示すように各セルにおいて、ｌｉｎｋには探索ノードの状態へのポインター、ｌｂｄには下限値が格納される。このリスト構造によって項目の出し入れの高速化が計れる。また双方向にすることによって、リストの処理が容易に行われる。また同レベル要素を高速に参照するために、付表を作成する。さらに全ての解を求めるようにするために、同レベルで同コストなセルをつなぐ。

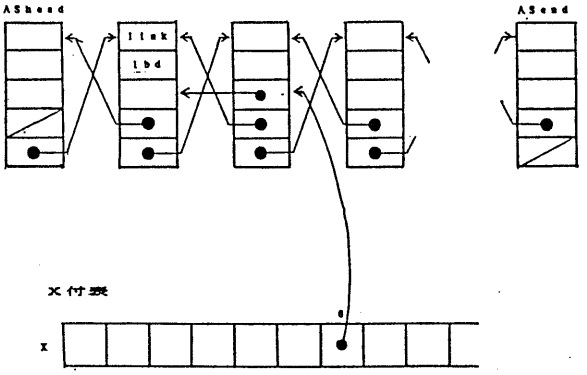


図5　オープンリストとレベルの付表

5）削除ルールの順序

削除ルールの順序においては細分化や同レベルのドミナンスが強いので、これを優先させることにより効果がいちじるしいものとなる。以下に順序を示す。

1：細分化によるドミナンス

2：同レベルによるドミナンス

3：上限値による選択

## 5．ＢＢの効率の評価

ここではＢＢの側面から、このアルゴリズムの効率を検討する。効率の尺度として分解操作の総数と、一回当たりの分解操作における活性化ノードの発生率を用いる。分解操作の総数によって、探索領域の大きさを示し、活性化ノードの発生率によって削除ルールの効率を示す。また、必要な記憶容量の評価値としてオープンリストの平均長を用いる。

図6にブロックサイズを4、エッジ数をノード数の2倍とした場合のノード数の増加に対する変化を示し、図7にノード数を１０００、エッジ数を２０００とした場合のブロックサイズの増加に対する変化を示した。

ノード数増加の場合、分解操作の数は図6に示すようにノード数に比例する。また、一回当たりの活性化ノードの発生率はほぼ1.3と非常に低いものとなっており、またオープンリストの平均長はほぼブロックサイズに同等な低い値におさまっている。

ブロックサイズの増加の場合、分解操作の数は大きな減少傾向を示す。1回当たりの活性化ノードの発生率は4を前後とした値になる。逆に、オープンリストの平均長はブロックサイズに比例し増大するという結果が得られている。

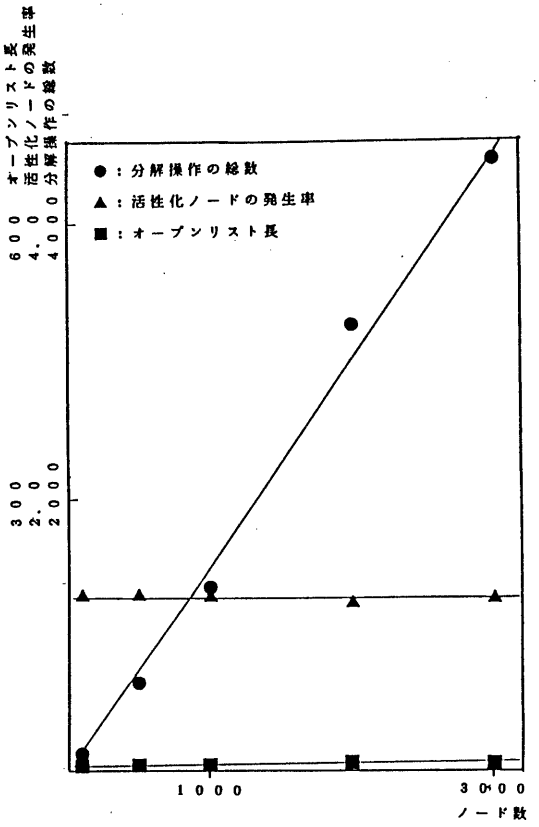


図6　ノード数に対する効率評価

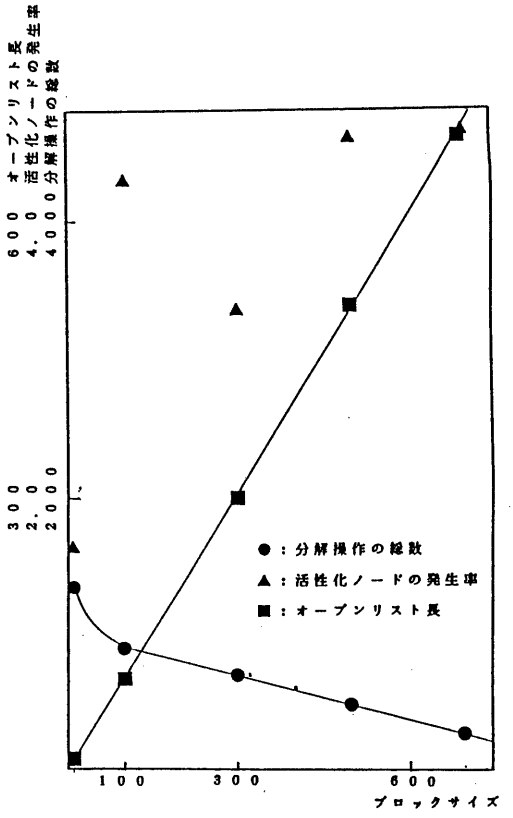


図7　ブロックサイズに対する効率評価

## 6．計算量

ＢＢはヒューリスティックな解法であるため、理論的な計算量は算出しにくい。ここでは、この問題の計算量を実験的な値を利用することによって求める。

１つの分解、限定操作過程において必要な計算はコスト計算、論理判断、オープンリストの変更の３つからなり、以上の操作を分解操作の個数分だけ繰り返す。ただし、Ｐがブロックサイズ、Ｅがエッジ数、およびＬがオープンリスト長とする。

１）コスト計算

コスト計算は１つの分解操作において、漸化式をブロックサイズ分、使用することになる。ここで、エッジへの参照部を補助段階においているので、Ｏ（Ｐ）とみることができる。また、補助段階の計算量は分解操作数と独立して、１つは出次数に依存し、１つは効率的なデータ圧縮を行うことによって、入次数に依存すると考えられるのでＯ（Ｅ）である。

２）論理判断

これは削除ルールを適用して、分岐規則によって生成された子ノードから活性化ノードを選択する論理判断である。一つの分解操作における、この判断の回数は最悪の場合、ブロックサイズの数であるので、論理判断の計算量はＯ（Ｐ）である。

３）オープンリストの変更

・先頭要素の取り出し、削除

リスト構造で構成されたオープンリストの先頭要素を取り出し、および削除するのに用する計算量はＯ（１）である。

・活性化されたノードの挿入

順序ずけられたリスト構造に子ノード集合である順序ずけられた要素をマージするので、この計算量は最悪の場合、Ｏ（Ｌ）である。

・変更、更新

同レベルな部分解の変更操作において、レベルの付表を用いることによって、　この計算量はＯ（１）となる。

・削除

上限値テストによるオープンリストの削除は最悪の場合、Ｏ（Ｌ）である。

しかし、この操作はこの問題においては使用する率が低い。

したがってオープンリストの変更における計算量はＯ（Ｌ）とみられる。

以上より、ＢＢの計算量は、これらの計算量の和に分解操作数Ｎを乗じたＯ（ＮＰ＋ＮＬ）と考えられる。また、補助データの構成を含めると、Ｏ（ＮＰ＋ＮＬ＋Ｅ）とみなされる。

## 7．拡張性

さらに問題の機能の拡張を計ってみる。図　に示すような同レベルな部分解をオープンリスト上に連結することによって、すべての解を求める算法を開発した。

また、ブレイクポイントとブロックの相対的位置にもとずく制限を加える問題を考察してみた。たとえば、ブレイクポイントおよびブロックサイズが１つの関係の場合ではブレイクポイントを任意のブロックの先頭あるいは後方に配置する問題、またブレイクポイントが２つに対して、任意の同一なブロック内にその２つのブレイクポイントを配置するか、配置しないかなどの問題が考えられる。１つの例として、図８に示すような２点のブレイクポイントを同一ブロック内に配置する場合、次のような削除ルールを付加することによ

って、容易に実現できる。

```
if  (A<ブレイクポイント<=B)
    削除
else
    not 削除
```

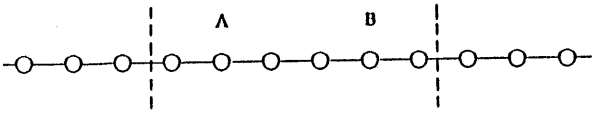


図8　ある性質をもつノードの
同一ブロック内への配置

## 8．おわりに

本論分ではＢＢの高速化について述べ、それらの評価を試みた。理論的な結果ではノード数とブロックサイズに線形に依存する値が得られた。さらに、問題の拡張、多様化を計ってみ、その容易性を示した。今後、ＢＢの基本ツールの構築および系列分割問題の多様化などの方向について検討していきたい。

## 参考文献

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

2) Walter.H.Kohler: Characterization and Theoretical Comparison of Brnch-and-Bound Algorithms for Permutation Problems, J.ACM, Vol.21, No.1, pp.140-156(1974).

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

4) 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).

5) 加地、大内: Ｂ＆Ｂによる最適系列分割問題における改善と拡張、情報処理学会第４２回全国大会、pp.105-106(1991).