北海道大学工学部研究報告
第157号（平成3年）

Bulletin of the Faculty of Engineering
Hokkaido University No. 157 (1991)

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

加地　太一*　大内　東

（平成3年6月24日受理）

Complexity of Optimal Sequential Partitions of Graphs by Dynamic Programming

Taichi KAJI and Azuma OHUCHI

(Received June 24, 1991)

Abstract

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 of consecutive numbers.

For this problem, Kernighan discusses an algorithm using dynamic programming, in which computational amount of referencing to data is proportional to the number of edges in the graph.

But he does not indicate total complexity of the algorithm.

In this paper, we evaluate the total complexity from the both standpoints of theoretical and implementing level, then we compare it with the results of numerical computing.

The results reveal that the running times of the procedure depends on the number of nodes and block size, beside the number of edges.

1．まえがき

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グラフの参照量が辺数に線形比例するアルゴリズムを開発した。しかし，アルゴリズム全体の計算量についてはふれられていない。ここで我々はこのアルゴリズムの全体の計算量について検討を試みる。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

情報工学科　システム工学講座
* 北海道情報大学　情報学科

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。また，数値実験を行なって評価値を確かめる。

以下，第2章で最適系列分割問題について説明し，第3章でKernighanの動的計画法についての概要を述べる。第4章では，加算と比較の演算数に着目した理論的計算量の算出を示す。第5章では，具象化されたアルゴリズムについて示し，増分コストの計算法およびデータ構造について論議する。第6章で，具象化されたアルゴリズムに基づいた計算時間の算出法を示す。第7章では，先に求めた計算量での頂点数，ブロックサイズ，辺数の変化に対しての影響を示す。

2．最適系列分割問題

頂点集合 $V=\{1, 2, \cdots, N\}$，辺集合 $E=\{e_1, e_2, \cdots, e_r\}$からなる無向グラフを $G=\{V, E\}$とし，頂点数 $N=|V|$とする。各辺は非負のコスト $C=\{c_1, c_2, \cdots, c_r\}$をもつ。各頂点は$\{w_1, w_2, \cdots, w_N\}$の重みをもち，各々$0<w_i \leqq P$ である。ここでPはブロックサイズと呼ばれる数である。一般性を失うことなくすべての重み w_i とPは正の整数とすることができる。また頂点部分集合 S $(\subseteq V)$の重みは$|S|=\Sigma_{i \in s} w_i$ とする。

Gをk個の部分集合 G_1，G_2，$\cdots$，G_k に，以下の制約（1），（2）のもとで，カットされる辺のコストの総和が最小になるよう分割する：

(1) $|G_i|$ (G_i の頂点の重みの総和) $\leqq P$,

(2) 任意の G_i の各頂点番号は連続的な番号をもつ。

部分集合 G_i はブロックと呼ばれ $G_i=\{j, j+1, \cdots\cdots, m-1, m\}$となる。$G_i$ での一番小さな頂点番号を b_i として，ブレイクポイントと呼ぶ。$b_i=j$ は頂点 $j-1$ と頂点 j の間でカットすることを意味する。集合$\{b_1, b_2, \cdots, b_k\}$は一意的に分割$\{G_1, G_2, \cdots, G_k\}$を表現する。

ここで，グラフGが有向グラフの場合，(i, j)と(j, i)の辺を含んでいるのなら，$i<j$ である単一辺(i, j)におきかえる。その時，コスト $c_{ij}=c_{ij}+c_{ji}$ とする。

図1は連続的な頂点集合 $V=\{1, 2, \cdots, 10\}$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1のグラフに対してブロックサイズ $P=4$ で，この問題の最適解を求めた場合，図1上の破線でカットされる。このとき，各部分集合は $\{1, 2, 3, 4\}$，$\{5, 6\}$，$\{7, 8, 9, 10\}$ となり，ブレイ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。

3．KernighanのDPアルゴリズム

以上の問題に対してKernighanはDPを適用した。以下にそのアルゴリズムの概略を記す。

ブレイクポイントxにおける最良な部分解のコストを $T(x)$とすると，$T(x)$は以下の関数方程式により漸化的に求められる。

$$T(x)=\min_y\{T(y)+C(x, y)\} \tag{3.1}$$

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$\leqq$P”の範囲で選ぶ。$C(x, y)$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，次の式で示される。

$$C(x, y)=\Sigma_{j \geqq x}^{y \leqq i<x} c_{ij} \tag{3.2}$$

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

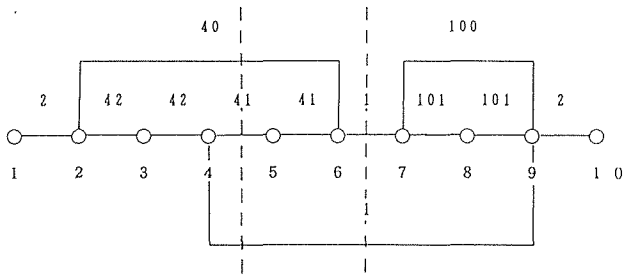


図1　系列分割グラフ

Kernighan が論文中で与えたアルゴリズムを示す。

1：Set　T(1)＝0；　L(1)＝0；
2：for　x＝2　to　n＋1　{
　　T(x)＝$\min_y$[T(y)＋C(x,y)]
　}
　但し，最小化は $d_{y,\,x-1} \leqq P$ をみたす全ての y の上でとる。(d_{ij} は頂点 i から j までの距離を表わし，重みの総和 $w_i+w_{i+1}+\cdots+w_j$ 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
　$z_1, z_2, \cdots, z_k$ 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(x, y) 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。

4．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。

算法のステップ1と3は単なる代入であり，ステップ5は出力である。ステップ4はたかだか N/P 程度の計算量であり，主な計算量が生ずるステップ2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，$d_{y,x-1} \leqq P$ は $x-y \leqq P$ 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 $C(x, y)$ の計算，および $C(x, y)$ を計算するために必要な中間量 W^x，Δ_i^x の計算に分解してステップ2の計算量を考察する。

4.1 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 $T(x)$；$x=2, 3, \cdots, N+1$ 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 $C(x, x-i)$；$i=1, 2, \cdots, x-1$ と，対応する最右端列の要素 $T(x-i)$；$i=1, 2, \cdots, x-1$ 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 $T(2)$, $T(3)$, $\cdots$, $T(N+1)$ と漸化的に求めていくことができる。

表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 $T(2)$, $\cdots$, $T(P)$ に対しては階段的に増大しており，一方 $T(P+1)$, $\cdots$, $T(N+1)$ に対しては常に一定幅 P のみに着目すればよい。そこで必要なすべての $C(x, y)$ がすでに計算されているという仮定のもとで，$T(2)$, $T(3)$, $\cdots$, $T(P)$ を求めるのに必要な計算量は加算と比較に着目して，

$$2\cdot(1+2+3+\cdots+(P-1))=P(P-1)$$

となり，一方，$T(P+1)$, $\cdots$, $T(N+1)$ を求めるのに必要な計算量は

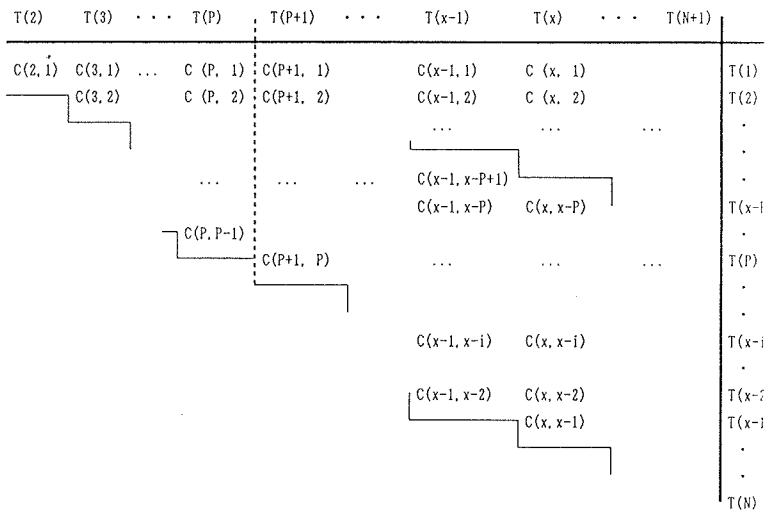
$$2\cdot P(N-P+1)$$

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，$T(2)$, $\cdots$, $T(N+1)$ を決定するのに要する計算量は

$$P(P-1)+2\cdot P(N-P+1)=2PN-P^2+P \qquad (4.1)$$

と求まる。

表1 増分コストの上三角タブロー



4.2 増分コスト C(x, y)の計算

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

$$\begin{aligned} C(x, y) = & C(x-1, y) \\ & + (c_{x-1, x} + c_{x-1, x+1} + \cdots + c_{x-1, N}) \\ & - (c_{y, x-1} + c_{y+1, x-1} + \cdots + c_{x-2, x-1}) \end{aligned} \tag{4.2}$$

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_i^x を用いて，

$$C(x, x-i) = C(x-1, x-i) + W^x - \Delta_i^x \quad (i=1, 2, \cdots, x-1) \tag{4.3}$$

と置く。ここで

$$w^x = c_{x-1, x} + c_{x-1, x+1} + \cdots + c_{x-1, N} \tag{4.4}$$

であり，Δ_i^x は次式である。

$$\Delta_1^x = 0$$

$$\Delta_j^x = \Delta_{j-1}^x + c_{x-j, x-1} \quad ; j=2, \cdots, x-1 \tag{4.5}$$

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(4.3)式からわかるように各々C(x, y)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(x, y)の総数は T(2)から T(P)までの階段的増大部分と T(P+1)から T(N+1)までの一定幅 P の中にある C(x, y)の個数の総和$(2PN-P^2+P)/2$ となる。したがって(4.3)式の全増分コストの計算量は，2回の演算を考慮すると，

$$2 \cdot (2PN-P^2+P)/2 = 2PN-P^2+P \tag{4.6}$$

である。

4.3 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(4.4)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

$$\Sigma_{x=2}^{N+1}(od(x-1)-1) = |E|-N \tag{4.7}$$

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(4.5)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，$id_{x-y<p}(x-1)$ 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 $x-y \leqq P$ を満たす入次数を $id_{x-y<p}(x)$ 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

$$\begin{aligned} & \Sigma_{x=2}^{N+1} id_{x-y<p}(x-1) = \Sigma_{x=2}^{N+1} id(x-1) - \Sigma_{x=2}^{N+1} id_{x-y>p}(x-1) \\ & = |E| - |E_{x-y>p}| \end{aligned} \tag{4.8}$$

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

4.4 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(4.1)，(4.6)，(4.7)および(4.8)式で与えられた量の総和であり，

$$\begin{aligned} & (2PN-P^2+P) + (2PN-P^2+P) + (|E|-N) + (|E|-|E_{x-y>p}|) \\ & = 4PN-2P^2+2P+2|E|-|E_{x-y>p}|-N \end{aligned} \tag{4.9}$$

となる。この(4.9)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，(4.9)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，PN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(4.9)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．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(4.9)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。

(1) 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。

(2) 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。

5.1 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2（図3）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

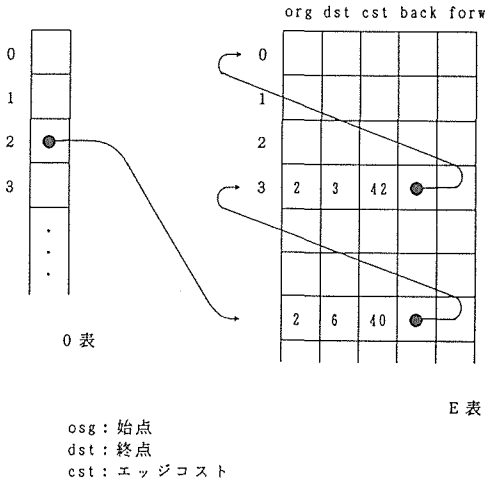


図2　流出辺リスト

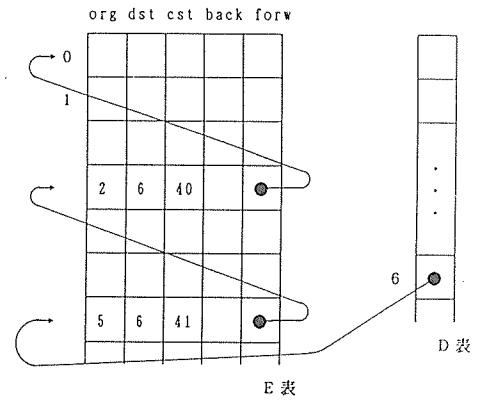


図3　流入辺リスト

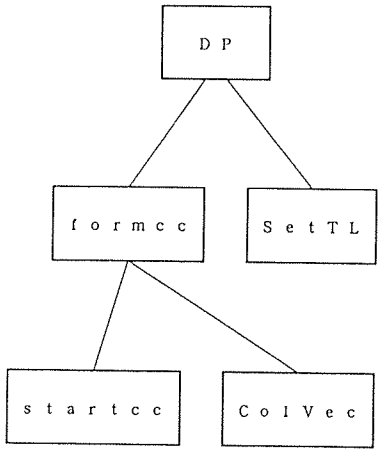


図4　手続き構成

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2（図3）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。

5.2　手続き構成

具象化されたプログラムは図4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(3.1)式から求める過程は，表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4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(3.1)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(4.3)式を使って必要な増分コスト計算を行なう過程を表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(x, 1), C(x, 2), ····, C(x, x−1)]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(x−1, 1), C(x−1, 2), ····, C(x−1, x−2), 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P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(4.3)式の中間量である w^x, Δ^x_i の過程を示す。(4.4)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(4.5)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 $id_{x-y<p}(x-1)$ となる。し

かし実現上，可変のPに対応する必要があり，また表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, P)；
}
```

{SetTL 手続き}

```
minval=MAXVAL；
for(y=unitf(x-P)；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-P；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 ;
```

6．具象化されたアルゴリズムの計算量

第5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4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2の演算時間記号で置き換えると，

startccの計算時間

$$
\begin{aligned}
&=\tau(=)\\
&\quad+\tau(=)+\tau([\mathrm{val}])\\
&\quad+\tau(\mathrm{loop})+\tau\ (\text{処理}_1)\\
&\quad+\tau(\mathrm{loop})+\tau\ (\text{処理}_2)\\
&\qquad\cdot\quad\cdot\quad\cdot\\
&\quad+\tau(\mathrm{loop})+\tau\ (\text{処理}_{\mathrm{od}(x-1)})\\
&\quad+\tau(\mathrm{return})
\end{aligned}
\tag{6.1}
$$

となる。

このループの回数はod(x−1)である。また，処理$_1$の計算量は$\tau(+=)+\tau([\mathrm{val}])$となるので，

$$
=\mathrm{od}(x-1)\cdot(\tau(\mathrm{loop})+\tau(+=)+\tau([\mathrm{val}]))+2\,\tau(=)+\tau([\mathrm{val}])+\tau(\mathrm{return}) \tag{6.2}
$$

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

表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	$\tau(\mathrm{loop})$	37
代入処理	$\tau(=)$	16
添字が変数である配列などにアクセスする処理時間	$\tau([\mathrm{val}])$	10
＋＝による加算代入処理	$\tau(+=)$	36
戻り値の処理時間	$\tau(\mathrm{return})$	20
・	・	・
・	・	・
・	・	・

DP の計算量$=\lambda NP-\beta P^2+\nu P+\eta N+\mu|E|+\Phi$ (6.3)

とまとめることができる。結果として(6.3)式は(4.9)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$(\lambda N+\nu)/2\beta$が最大値となる上に凸な曲線である。

さらに(6.3)式における係数の値を具体的に求める。表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(6.3)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$(10^{-7}s)$

$=292\cdot NP-164.5\cdot P^2+164.5\cdot P+497\cdot N+162\cdot|E|+16$ (6.4)

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ψN^2の計算量が現実には存在する。

7．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ψN^2とする。実測値では$\psi=80$であり，以下の実験ではψN^2を引いた値で考察する。

7.1 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。(6.4)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。

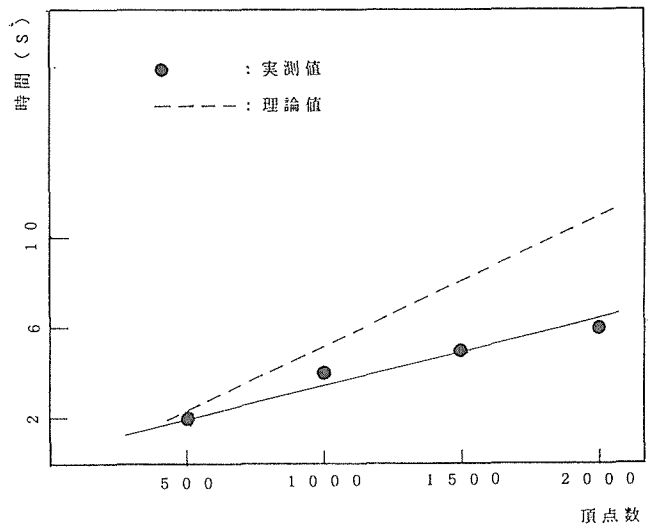


図5 頂点数と時間計算量

7.2　ブロックサイズによる変化

図6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(6.4)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。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。

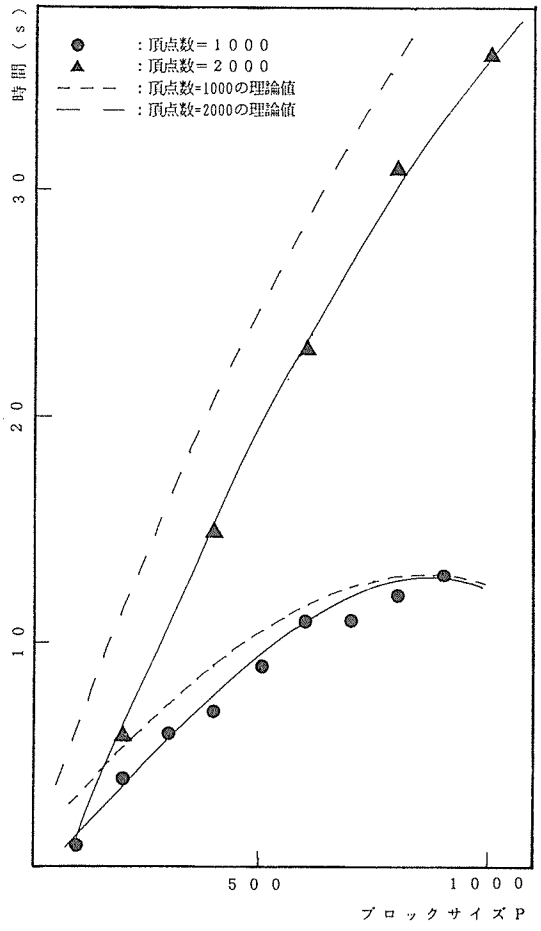


図6　ブロックサイズと時間計算量

7.3　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(6.4)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。

8．ま と め

本論分では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。

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 Branch-and-Bound Algorithms for Permutation Problems, J. ACM, Vol. 21, No. 1, pp. 140-156 (1974).
3）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).
4）加地，大内：最適系列分割問題に対するDPとBBアルゴリズム，情報処理学会第41回全国大会，pp. 89-80 (1990).