数理解析研究所講究録
333巻 1978年 189-198

最大カットの近似算法

東北大 工学部 小川 秀直
西関 隆夫
斎藤 伸自

1. まえがき

与えられた無向グラフにおいて，最大個数の枝をもつカットを求める，いわゆる最大カット問題は，NP-完全問題の一つであり，[1] 現在までのところ効率の良い，即ちグラフの節点数あるいは枝数の多項式オーダの手間をもつアルゴリズムはみつかっていない。これに対して，多項式オーダの手間を持つ様な近似算法が2,3提案されている。[2][3] このうち，現在までのところ最も良いと思われる久保他の方法[3]は，節点対の交換を繰り返して近似最大カットを求め，最後に極大化操作を施すという算法であり，その計算手間は$O(|E|^2)$である。

ここでは，最大型組合せ問題の近似解の安定度を表わす指標として，「ρ極大」なる概念を導入し，ρ極大カットの特徴付けを与え，次いで，ρ極大カットが得られる様な，最大カットの近似算法ρ-APMAXCUTを提案し，計算機実験によ

り，その有効性を示す。

2. k極大カットの特徴付け

まず，k極大を次の様に定義する。

［定義］ 集合Aの部分集合のうち，ある性質を満足する部分集合すべてからなる族を$\mathcal{A}$とする。$A_1 \in \mathcal{A}$なるA_1に対し，$|A_1 - A_2| \leq k$かつ$|A_2| > |A_1|$なる$A_2 \in \mathcal{A}$が存在しないとき，A_1はその性質に関してk極大であるという。

グラフのk極大カットを特徴付ける諸定理を以下に示す。

［定理1］[3] 与えられたグラフGのカットKが極大であるための必要十分条件は，KがGの木を少くとも1つ含む事，即ち，Gから$\overline{K}$の枝を開放除去して得られるグラフ$G[\overline{K}, \phi]$が連結である事である。

［定理2］ 与えられたグラフGのカットKが極大であるための必要十分条件は，$K = \bigoplus_{e \in T} S(e,T)$なる木Tが存在する事である。ここで，$S(e,T)$は，木Tと枝$e \in T$によって決まる基本カットセットであり，$\oplus$は$mod 2$の演算を表わす。

［定理3］ Kを与えられたグラフGの$(k-1)$極大カットとする。ただし，$k \geq 1$。

(a) Kがk極大であるための必要十分条件は，$k = |S \cap K| < |S - K|$なるGのカットセットSが存在しない事である。

(b) もし，$k = |S \cap K| < |S - K|$なるGのカットセットSが存在

すれば，$K \cap S$ は $G[\overline{K}, \phi]$ のカットセットである。

[定理4] Kを与えられたグラフGの $(k-1)$ 極大カットとする。このとき，Kが k 極大であるための必要十分条件は，$G[\overline{K}, \phi]$ に，次の増大条件を満足する様な k 本の枝からなるカットセット $S=\{(u,v) \mid u \in V_1, v \in \overline{V_1}\}$ が存在しない事である。ただし $k \geq 1$。

（増大条件）$e_i=(u_i, v_i)$, $u_i \in V_1$ かつ $v_i \in \overline{V_1}$ なる $k+1$ 本以上の枝 $e_i \in \overline{K}$ が存在する事。

次の様な，k 極大カットの簡単な十分条件が，定理3及び4の系として得られる。

[系] KをグラフGのカットとする。このときグラフ $G[\overline{K}, \phi]$ の枝連結度が $k+1$ 以上であるとき，Kは k 極大である。

k 極大の定義より，「カットKが $|K|$ 極大ならば，Kは最大カットである」事は明らかである。上述の定理を用いれば更に次の定理が得られる。

[定理5] Kをグラフ $G=(V,E)$ のカットとし，k を

$$k = \min\left(|K|-|V|+2, \left\lceil \frac{1}{2}(|E|-|V|) \right\rceil\right)$$

とする。Kが k 極大であるとき，Kは最大カットである。ただし $\lceil x \rceil$ は x より小さくない最小の整数を表わす。

3. 最大カットの近似算法 k-APMAXCUT

ここでは，得られる近似最大カットが k 極大である様な近

似算法 k-APMAXCUT を提案する。k-APMAXCUTは，次に定義するカットの集合 $\mathcal{K}^{(k)}(T)$ を用いた図1の算法である。ここで，$K(T)$は，木Tに対応して定まる $K(T)=\oplus_{e\in T} S(e,T)$ なる極大カットとし，木Tに対応して定まるカットの集合 $\mathcal{K}^{(k)}(T)$ は，

$$\mathcal{K}^{(1)}(T)=\{S(e,T)\mid e\in T\}+\{K(T)\oplus S(e,T)\mid e\in T\}$$

$$\mathcal{K}^{(k)}(T)=\mathcal{K}^{(k-1)}(T)+\{\oplus_{e\in D} S(e,T)\mid D\subset T \text{ かつ } |D|=k\}$$

$$+\{K(T)\oplus(\oplus_{e\in D} S(e,T))\mid D\subset T \text{ かつ } |D|=k\}\ (k\geq 2)$$

と定義する。

まず，k-APMAXCUT の計算手間及び使用メモリについては次の事がいえる。

［定理6］ k-APMAXCUTの計算手間は $O(|V|^k\cdot|E|^2)$ であり，使用メモリは $O(|E|)$ である。

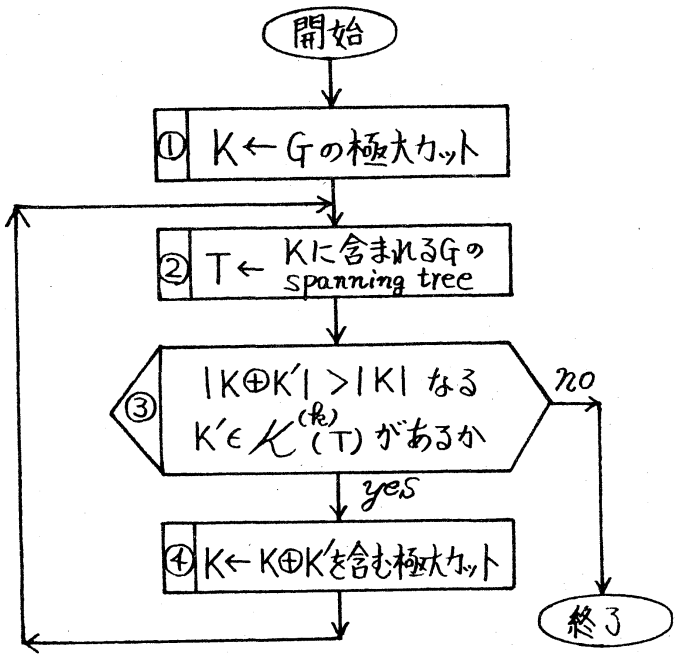

図1. k-APMAXCUTの流れ図

（証明）k-APMAXCUTの①，②及び④のいずれも1回実行するのに要する手間は $O(|E|)$ である。又，$|\mathcal{K}^{(k)}(T)|=O(|V|^k)$ であるから③の部分を1回実行するのに要する手間は $O(|V|^k\cdot|E|)$ で

ある。ループを回る毎に|K|は真に増大するのでループを回る回数は高々|E|回である。従ってk-APMAXCUT全体の手間は$O(|V|^k\cdot|E|^2)$である。使用メモリが$O(|E|)$である事は明らか。

次に，k-APMAXCUTでk極大が得られる事を示す。まず，$\mathcal{K}^{(k)}(T)$の部分集合$\tilde{\mathcal{K}}^{(k)}(T)$を次の様に定義する。

$$\tilde{\mathcal{K}}^{(1)}(T)=\{S(e,T)\mid e\in T\}$$

$$\tilde{\mathcal{K}}^{(k)}(T)=\tilde{\mathcal{K}}^{(k-1)}(T)+\{\oplus_{e\in D}S(e,T)\mid D\subset T\text{ かつ }|D|=k\}\quad(k\geq 2)$$

[定理7]　Tをグラフの木とする。$K'\in\tilde{\mathcal{K}}^{(k)}(T)$なるどのK'についても$|K(T)\oplus K'|\leq|K(T)|$であるとき，K(T)は$k$極大である。

(証明) もしK(T)がk極大でないとすると，定理3により，$|K(T)\oplus S|>|K(T)|$ かつ $k\geq|K(T)\cap S|<|S-K(T)|$ なるGのカットセットSが存在する。いま$S=\oplus_{e\in T\cap S}S(e,T)$であり，$|T\cap S|\leq|K(T)\cap S|\leq k$ であるから，$S\in\tilde{\mathcal{K}}^{(k)}(T)$であり矛盾する。

```
      procedure 1-MAXIMAL;
      comment graph G=(V,E) is represented by the adjacency lists L(v)
           for all v in V where each edge (v,w) on L(v) is marked whether
           it belongs to K or not.  K is the current cut and |V|=n;
      begin logical cond; integer COUNT; stack STACKA,STACKB;
 1.   START:
 2.        for all vertices v in V do mark v "new";
 3.        COUNT:=1;
 4.        v:= an arbitrary vertex in V;
 5.        SEARCH(v);
13.        for all vertices v in V do mark v "new";
14.        COUNT:=1;
15.        v:=1;
16.        make STACKA and STACKB empty;
17.        cond:=false;
18.        BRIDGE(v);
54.        if cond:=true then goto START
      end
```

```
      procedure SEARCH(v);
      begin integer array DFNUMBER(n);
 6.       mark v "old";
 7.       DFNUMBER(v):=COUNT;
 8.       COUNT:=COUNT+1;
 9.       for each vertex w on L(v) do
10.            if (v,w) is in K then
11.                 if w is marked "new" then
                         begin
12.                           SEARCH(w);
                         end
      end
      .


      procedure BRIDGE(v);
      begin integer array LOW(n),FATHER(n);
19.       mark v "old";
20.       LOW(v):=COUNT;
21.       COUNT:=COUNT+1;
22.       for each vertex w on L(v) do
23.            if (v,w) is in K then
24.                 if w is marked "new" then
                         begin
25.                           FATHER(w):=v;
26.                           BRIDGE(w);
27.                           if cond:=true then goto END
28.                           else if LOW(w)≥DFNUMBER(v) then
                                        comment the vertex v is an articulation
                                           point
29.                                     if LOW(w)>DFNUMBER(v) then
                                             comment in this case, the edge
                                                  (v,w) is a bridge;
                                             begin
30.                                               SAT(w);
43.                                               if cond:=true then goto L1
                                             end
44.                                     else LOW(v):=MIN(LOW(v),LOW(w));
                         end
45.                 else if w is not FATHER(v) then
46.                           LOW(v):=MIN(LOW(v),DFNUMBER(w));
47.            else put the edge (v,w) on STACKA;
48.       goto END
49.  L1:  while STACKB is not empty do
                begin
50.                pop up an edge (x,y) from top of STACKB;
51.                add the edge (x,y) to K;
                end
52.       delete the edge (v,w) from K;
53.  END:
      end
```

```
      procedure SAT(w);
      begin integer L,H,NUMBER;
31.       L:=DFNUMBER(w);
32.       H:=COUNT-1;
33.       NUMBER:=0;
34.       while L≤x≤H do
              begin
35.               pop up an edge (x,y) from STACKA;
36.               if y<L or H<y then
                      comment in this case, the edge (x,y) satisfies
                          the augmenting condition;
                      begin
37.                       NUMBER:=NUMBER+1;
38.                       put the edge (x,y) on STACKB;
                      end
              end
39.       if NUMBER≤1 then
              comment in K, there exists at most one edge which satisfies
                  the augmenting condition;
              begin
40.               pop up all edges from STACKB and put them on STACKA;
              end
41.       else comment in this case, the bridge (v,w) satisfies
                      the augmenting condition;
42.           cond:=true;
      end
```

図2. 1-MAXIMALのプログラム

定理7より，1極大カットは$O(|V|\cdot|E|^2)$の手間で求まる事が分かるが，次に，1極大カットを$O(|E|^2)$の手間で求める算法1-MAXIMALを，定理4にもとづいく示す。1本の枝からなるカットセットは，いわゆるブリッジ枝であるので，depth-first search[4]を用いれば手間を減らす事が出来る。同様にして，2極大カットを$O(|E|^2)$の手間で求める算法2-MAXIMALが作れる。

4. 計算機実験

著者らが開発したグラフ処理プログラムGRAMP[5]を用いく，1-MAXIMAL，2-MAXIMAL，1-APMAXCUT及び2-APMAXCUT

をプログラムし，種々のランダムグラフ[6]に適用し，それらと久保他の算法との比較を行った。まず種々の枝数をもつ節点数15のランダムグラフを各々200個発生し，上述の5種の算法により最大カットが得られる割合，即ち正解率を求めた。その結果を図3に示す。

k-APMAXCUTで得られるカットKは，$K' \in \mathcal{L}^{(k)}(\Gamma)$ なるどの様なカットK'についても，$|K| \geq |K \oplus K'|$ なる性質がある。これに対し，$\mathcal{L}^{(k)}(\Gamma)$ のかわりに，$T_i \subset K$ なる定数個の木 T_i $(i=1,2,\cdots t)$ によって定まる $\bigcup_{i=1}^{t} \mathcal{L}^{(k)}(T_i)$ を考え，$K' \in \bigcup_{i=1}^{t} \mathcal{L}^{(k)}(T_i)$ なるどの様なカットK'についても，$|K| \geq |K \oplus K'|$ なる性質があるカットKを得る様に，k-APMAXCUTを修正する事が出来る。この修正

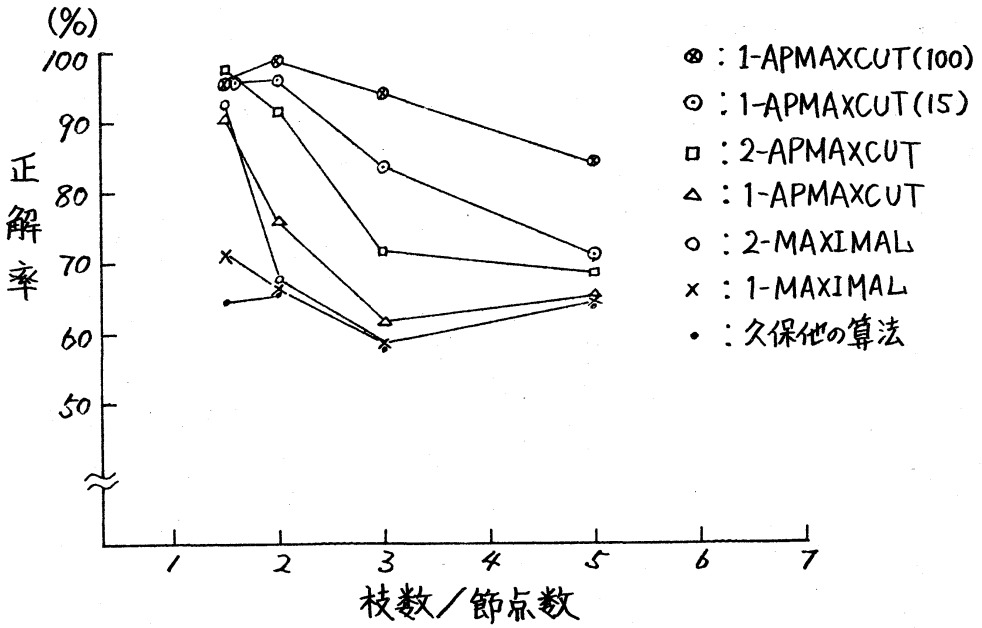

図3. 各算法の正解率（最大カットの得られる割合．$|V|=15$，各枝数につきランダムグラフ200個）

された算法を k-APMAXCUT(t)とかこう。kが定数である限り，k-APMAXCUT(t)の計算手間のオーダはk-APMAXCUTと同じである。図3にk-APMAXCUT(t)の正解率を示す。なお，木$T_1, T_2, \cdots, T_t$はランダムに選んでいる。同図で，2-APMAXCUTよりも，1-APMAXCUT(t)の方が良い正解率を得ている点は興味深い。

5. むすび

グラフの最大カットを求める新しい近似算法k-APMAXCUTを提案し，その算法で得られるカットはk極大であり，その計算手間は$O(|V|^k|E|^2)$以下である事を示した。特に1極大カットは1-MAXIMALにより$O(|E|^2)$の手間で求まる事を示した。

前節に示した様に，$K \in \bigcup_{i=1}^{t} L^{(k)}(T_i)$ なるどのK'についても $|K| \geq |K \oplus K'|$ なるカットKを求める様にk-APMAXCUTを修正したk-APMAXCUT(t)は，従来にはない新しい考え方にもとづいており，しかも大きいkのk-APMAXCUTより，小さいkのk-APMAXCUT(t)の方が良い特性をもつという興味深い傾向がある事を示した。しかし，良い特性を得るには，木$T_1, T_2, \cdots T_t$をどの様に選べば良いかは未解決であり，今後の研究に残されている。

謝辞　日頃御討論いただく山口大学高浪五男教授，東北大学浅野孝夫博士，並びに「k極大」を御示唆いただいた東京

工業大学冨澤信明博士に深謝いたします。なお，本研究の一部は文部省科学研究補助金：総合研究A 135017（昭和51年度）「ネットワーク構造を持つシステムに関する基礎研究」の援助のもとに行なわれたものである。

文　献

(1) M.R. Garey, D.S. Johnson and L. Stockmeyer: "Some simplified NP-complete problems," Proc. 6th Annual ACM Symp. on Theory of Computing, P.47 (1974).

(2) 高浪五男："枝の間放除去によるグラフのバイパータイト化，"信学論(A), 57-A. 2, P. 152 (昭49-02).

(3) 久保，築山，白川，有吉："極大カット算出の一手法，"信学論(A), J60-A, 4, P.383 (昭52-04).

(4) A.V. Aho, J.E. Hopcroft and J.D. Ullman; "The design and analysis of computer algorithms," Addison Wesley, Reading, Mass. (1974).

(5) 滝内，高見沢，西関，斎藤："グラフ処理言語-GRAMP-"，信学会回路とシステム研資，CST76-117 (昭51-12).

(6) 高見沢，滝内，西関，斎藤："ランダムグラフの統計解析，"信学会回路とシステム研資，CST76-122 (昭51-12).