2006－MPS－62 (19)
2006－BIO－7 (19)
2006／12／22

# 極大クリーク全列挙アルゴリズムCLIQUESを基にした極大2部クリーク全列挙アルゴリズム

仲川 崇史, 富田 悦次
電気通信大学 電気通信学研究科 情報通信工学専攻
〒182-8585 東京都調布市調布ケ丘 1-5-1
{tak, tomita}@ice.uec.ac.jp

**概要**. 2部グラフ中の極大2部クリークの全列挙は極大クリーク全列挙アルゴリズムを用いて行うことができる．しかし，この手法は膨大なメモリが必要となり得る．本稿では極大クリーク全列挙アルゴリズム CLIQUES を基に極大2部クリークの列挙に特化したアルゴリズムを提唱する．更に，計算機実験によりそのアルゴリズムが CLIQUES よりも使用するメモリが非常に小さく，かつ高速であることを示す．

# An Algorithm for Generating All Maximal Bipartite Cliques Based on CLIQUES that Generates All Maximal Cliques

Takashi Nakagawa and Etsuji Tomita
Graduate school of Electro-Communications
The University of Electro-Communications
Chofugaoka 1-5-1, Chofu, Tokyo 182-8585, Japan
{tak, tomita}@ice.uec.ac.jp

**Abstract.** An algorithm for generating all maximal cliques can generate all maximal bipartite cliques in a bipartite graph, while it may take very large space. In this note, we present a specialized algorithm for generating all maximal bipartite cliques based on CLIQUES that generates all maximal cliques. We show that the algorithm takes much smaller space and is faster than CLIQUES by computational experiments.

## 1 はじめに

極大2部クリークを列挙する問題にはデータマイニング等様々な応用がある [1],[2]．極大2部クリークの全列挙は与えられた2部グラフに簡単な変換を施し極大クリーク全列挙アルゴリズムを適用することで行える．しかし, この手法は再帰が非常に深くなり膨大なメモリが必要となり得るため，大規模問題に適用するのは難しい．Makino and Uno は極大クリーク全列挙アルゴリズム ALL_MAX_CLIQUES* を基にした極大2部クリーク全列挙アルゴリズムを優れた実験結果とともに発表している [3]．また，筆者らは最大計算時間量が節点数の関数として最適である極大クリーク全列挙アルゴリズム CLIQUES を提唱し，計算機実験によりこのアルゴリズムが実際にも高速であることを示している [4], [5]．本稿では CLIQUES を基にした極大2部クリーク全列挙アルゴリズムを提唱し，計算機実験によりこのアルゴリズムは CLIQUES よりも使用するメモリが非常に小さく, かつ高速であることを示す．

## 2 諸定義と記法

グラフ $G=(V,E)$ について，$v \in V$ の隣接節点集合を $\Gamma(v)$ と表す．節点部分集合 $C \subseteq V$ によって誘導されるGの部分グラフ中の任意の節点が互いに隣接している時，$C$ を $G$ のクリークと呼ぶ．$C$ が他のクリークの真部分集合でない時 $C$ を極大クリークと呼ぶ．

グラフ $G=(V,E)$ について，$V$ が2つの節点集合 $V_1, V_2$ ($V_1 \cap V_2 = \emptyset$) より構成され，$V_1$ 及び $V_2$ 中の節点同士の隣接関係は存在しない時，$G$ を2部グラフと言う．節点部分集合 $C = C_1 \cup C_2 (C_1 \subseteq V_1, C_2 \subseteq V_2)$ により誘導される $G$ の部分グラフにおいて，$C_1$ 中の任意の節点と $C_2$ 中の任意の節点が隣接している時，$C$ を $G$ の2部クリークと呼ぶ．$C$ が他の2部クリークの真部分集合でない時 $C$ を極大2部クリークと呼ぶ．

## 3 極大2部クリーク全列挙

### 3.1 グラフの変換

与えられた2部グラフ $G=(V=V_1 \cup V_2, E)$ に対し，$V_1$ 及び $V_2$ がクリークとなるように枝を追加したグラフを $G'=(V,E')$ とする．また，$G'$ において節点 $v \in V$ に隣接している節点の集合を $\Gamma'(v)$ と表す．この時，$G$ における極大2部クリークは $G'$ における極大クリークとなる．よって，$G'$ に極大クリーク全列挙アルゴリズムを適用すれば $G$ 中の極大2部クリークを全列挙できる．しかし，この手法では極大2部クリークとともに $V_1$, $V_2$ も出力する可能性がある．これは不必要な計算時間及びメモリを使うことになる．

### 3.2 アルゴリズムBCU

本稿で提唱するアルゴリズム BCU を図 1, 2 に擬似 PASCAL で示す．BCU は CLIQUES と同様の深さ優先探索で極大2部クリークを全列挙し木の形で出力する．

手続き EX′_D1($SUBG, CAND$) は $V_1$ 中の節点と $V_2$ 中の節点を1つずつ含むクリーク $Q$ を生成し，再帰手続き EXPAND′ により極大になるまで拡張する．この手続きにより，BCU は $V_1$ 中の節点と $V_2$ 中の節点を両方含む $G'$ 中の極大クリーク，すなわち $G$ 中の極大2部クリークのみを抽出する．また，EX′_D1 は $G'$ 中で次

```
procedure BCU(G = (V = V_1 ∪ V_2, E))
begin
    E' := E ∪ (V_1 × V_1) ∪ (V_2 × V_2)
      −{(v_i, v_i) | 1 ≤ i ≤ |V|};
    EX'_D1(V, V)
end { of BCU }

procedure EX'_D1(SUBG, CAND)
begin
    u := a vertex that has the largest degree in SUBG;
    while CAND − Γ'(u) ≠ ∅ do
        p := a vertex in CAND − Γ'(u);
        print (p, “,”);
        SUBG_p := SUBG ∩ Γ'(p);
        CAND_p := CAND ∩ Γ'(p);
        if p ∈ V_1 then
            while CAND_p ∩ V_2 ≠ ∅ do
                q := a vertex in CAND_p ∩ V_2;
                print(q, “,”);
                SUBG_pq := SUBG_p ∩ Γ'(q);
                CAND_pq := CAND_p ∩ Γ'(q);
                EXPAND'(SUBG_pq, CAND_pq);
                print (“back,”);
                CAND_p := CAND_p − {q}
            od
        else
            while CAND_p ∩ V_1 ≠ ∅ do
                q := a vertex in CAND_p ∩ V_1;
                print(q, “,”);
                SUBG_pq := SUBG_p ∩ Γ'(q);
                CAND_pq := CAND_p ∩ Γ'(q);
                EXPAND'(SUBG_pq, CAND_pq);
                print (“back,”);
                CAND_p := CAND_p − {q}
            od
        fi
        print(“back,”);
        CAND := CAND − {p}
    od
end { of EX'_D1}
```

図 1: アルゴリズム BCU 及び手続き EX'_D1

```
procedure EXPAND'(SUBG, CAND)
begin
  if SUBG = ∅ then print (“clique,”)
  else if CAND ≠ ∅ then
    if (CAND ⊆ V_1 or CAND ⊆ V_2) and there is no vertex
     in SUBG adjacent to all vertices in CAND then
      for all vertices v ∈ CAND do print(v, “,”) od
      print(“clique,”);
      for i := 1 to |CAND| do print(“back,”) od
    else
      u := a vertex u in SUBG
       that maximizes |CAND ∩ Γ'(u)|;
      while CAND − Γ'(u) − {u} ≠ ∅ do
        p := a vertex in CAND − Γ'(u) − {u};
        print (p, “,”);
        SUBG_p := SUBG ∩ Γ'(p);
        CAND_p := CAND ∩ Γ'(p);
        EXPAND'(SUBG_p, CAND_p);
        print(“back,”);
        CAND := CAND − {p}
      od
      if u ∈ CAND ∩ V_1 then
        print(u, “,”);
        SUBG := SUBG ∩ Γ'(u);
        CAND := CAND ∩ Γ'(u);
        if there is no vertex in SUBG adjacent to
         all vertices in CAND ∩ V_1 then
          for all vertices v ∈ CAND ∩ V_1 do print(v, “,”) od
          print(“clique,”);
           for i = 1 to |CAND ∩ V_1| do print(“back,”) od
        fi
        while CAND ∩ V_2 ≠ ∅ do
          p := a vertex in CAND ∩ V_2;
          print(p, “,”);
          SUBG_p := SUBG ∩ Γ'(p);
          CAND_p := CAND ∩ Γ'(p);
          EXPAND'(SUBG_p, CAND_p);
          print(“back,”);
          CAND := CAND − {p}
        od
        print(“back,”)
      else if u ∈ CAND ∩ V_2 then
        print(u, “,”):
        SUBG := SUBG ∩ Γ'(u);
        CAND := CAND ∩ Γ'(u);
        if there is no vertex in SUBG adjacent to
         all vertices in CAND ∩ V_2 then
          for all vertices v ∈ CAND ∩ V_2 do print(v, “,”) od
          print(“clique,”);
          for i = 1 to |CAND ∩ V_2| do print(“back,”) od
        fi
        while CAND ∩ V_1 ≠ ∅ do
          p := a vertex in CAND ∩ V_1;
          print(p, “,”);
          SUBG_p := SUBG ∩ Γ'(p);
          CAND_p := CAND ∩ Γ'(p);
          EXPAND'(SUBG_p, CAND_p);
          print(“back,”);
          CAND := CAND − {p}
        od
        print(“back,”)
      fi
    fi
  fi
end { of EXPAND' }
```

図 2: 手続き EXPAND'

数が最大の節点 $u$ について，$Q$ に加える 1 つ目の節点の候補から $\Gamma'(u)$ 中の節点を除き探索領域を削減する．これは CLIQUES[4] でも用いられている手法である．

再帰手続き EXPAND$'(SUBG, CAND)$ において $SUBG$ は $G'$ において保持しているクリーク $Q$ 中の全節点に隣接している節点全ての集合，$CAND(\subseteq SUBG)$ は $Q$ に加えられる候補節点の集合である．EXPAND$'$ は $Q$ に $CAND$ 中の節点を 1 つまたは 2 つ加えた後，更に深い再帰レベルの EXPAND$'$ を実行することで $Q$ を極大にまで拡張する．ただし，$CAND$ の定義と $G'$ において $V_1$ 及び $V_2$ はクリークであることより，$CAND \subseteq V_1$ または $CAND \subseteq V_2$ であるとき $Q \cup CAND$ がクリークであることは自明である．よって，このときは再帰を行わず，$Q \cup CAND$ が極大であるか調べ，極大であればこれを出力する．

再帰を行う場合，初めに $CAND \cap \Gamma'(u)$ を最大にする節点 $u \in SUBG$ について，EX$'$_D1 と同様に $Q$ に加える 1 つ目の節点の候補から $CAND \cap \Gamma'(u)$ 中の節点を除く．その後，$CAND - \Gamma'(u)$ 中の節点を $Q$ に加え，更に $Q$ を $G'$ における極大クリークとなるまで拡張した後，その節点を $CAND$ から取り除くという操作を $CAND - \Gamma'(u) = \emptyset$ となるまで繰り返す．$Q$ に加えた節点 $p$ について $u \in V_1$ かつ $p \in V_2$，または $u \in V_2$ かつ $p \in V_1$ となるときは，$Q$ に対し 1 つ深い再帰レベルの EXPAND$'$ を直接適用することで $Q$ を拡張する．しかし，$u \in V_1$ かつ $p \in V_1$ となるときは $Q$ に $p$ を加えた後，更に $Q$ に $CAND \cap \Gamma'(p) \cap V_2$ 中の節点を 1 つ加え，その後 1 つ深い再帰レベルの EXPAND$'$ を実行するという特別な操作を行う．同様に，$u \in V_2$ かつ $p \in V_2$ となるときは $Q$ に $p$ を加えた後，更に $Q$ に $CAND \cap \Gamma'(p) \cap V_1$ 中の節点を 1 つ加え，その後 1 つ深い再帰レベルの EXPAND$'$ を実行する．ここで，$G'$ において $u \in V_1$ ならば $u$ は $CAND \cap (V_1 - \{u\})$ 中の全節点と隣接しているため $CAND - \Gamma'(u) - \{u\} \subseteq V_2$ となる．同様に，$u \in V_2$ ならば $CAND - \Gamma'(u) - \{u\} \subseteq V_1$ となる．すなわち，実際に特別な操作を行うのは $p = u$ のときのみである．

上記の操作では $u \in V_1$ の場合は $Q \cup (CAND \cap V_1)$ が，$u \in V_2$ の場合は $Q \cup (CAND \cap V_2)$ が $G'$ 中の極大クリークであったとき，これを抽出することができない．よって，EXPAND$'$ はこの極大クリークを別途に生成する．この操作は再帰を使わずに行うことができる．

以上の操作により，EXPAND$'$ は $u \in V_1$ ならば $CAND \cap V_2$ 中の節点を，$u \in V_2$ ならば $CAND \cap V_1$ 中の節点を必ず 1 つ $Q$ に加えた後に 1 つ深い再帰レベルの EXPAND$'$ を実行することになる．この操作の目的は $CAND \cap V_1$ または $CAND \cap V_2$ のどちらか片方から優先的に節点を取り出し $Q$ の拡張に用いることにより，再帰が浅い段階で $CAND \subseteq V_1$ または $CAND \subseteq V_2$ となるようにすることである．ここで，$V_1$ と $V_2$ の内，$u$ が含まれていない方に含まれる節点を優先的に $CAND$ から取り除くようにするのは以下の理由からである．$u \in V_1$ ならば $u$ は $CAND \cap (V_1 - \{u\})$ 中の全節点と，$u \in V_2$ ならば $u$ は $CAND \cap (V_2 - \{u\})$ 中の全節点と隣接している．よって，$|CAND \cap V_1| > |CAND \cap V_2|$ ならば $u$ は $V_1$ 中の節点から選ばれ，さもなければ $u$ は $V_2$ 中の節点から選ばれる可能性が高い．よって，$V_1$ と $V_2$ の内 $u$ が含まれていない方に含まれる節点を優先的に $CAND$ から取り除くようにすれば，$CAND \cap V_1$ と $CAND \cap V_2$ の内どちらか小さい方から優先的に節点を取り除くことになる可能性が高い．$CAND \cap V_1$ と $CAND \cap V_2$ の内小さい方から優先的に節点を取り除くようにすれば，より再帰が浅い段階で $CAND \subseteq V_1$ または $CAND \subseteq V_2$ となることが期待できる．

## 4 計算機実験

アルゴリズム CLIQUES 及び BCU を C 言語で実装し計算機実験で性能を評価した．使用した計算機は Pentium4 3.60GHz の CPU と 2GB の主メモリ及び Linux OS を搭載したものである．コンパイラ及びフラグは gcc -O2 である．本来 CLIQUES は保持しているクリーク $Q$ に節点を 1 つ加えるごとに 1 つ深いレベルの再帰を行うアルゴリズムであるが，今回の実験で実装した CLIQUES のプログラムは BCU と同様に $CAND \subseteq V_1$ または $CAND \subseteq V_2$ のときはそれ以上深い再帰を行わないようにしている．更に，抽出した極大クリークが $V_1$ または $V_2$ であった時はこれを出力しないようにしている．実験に用いたグラフはランダムグラフ及び局所的ランダムグラフ [3] である．ランダムグラフは節点数 $n = |V_1|$, $m = |V_2|$ 及び枝存在確率 $p$ の組み合わせごとに，局所的ランダムグラフは $n, m$ 及びパラメータ $r$(文献 [3] 参照) の組み合わせごとに 10 個ずつ作成し，実行時間の平均を求めた．ただし，25,000 秒以上かかる場合は 1 つのグラフでのみ測定した．実験結果を表 1 に示す．表中で dens は $|E|/(|V_1| \times |V_2|)$ で求めた枝密度，$\omega$ は最大 2 部クリークサイズ，$\sharp$cliques は極大 2 部クリークの数である．CL. [sec] は CLIQUES の実行時間，BCU [sec] は BCU の実行時間を表す．単位は秒である．CL. dep. は CLIQUES の再帰の深さ，BCU dep. は BCU の再帰の深さを表す．表中で最小の実行時間，再帰の深さは太字で表している．

実験結果より，全てのグラフで BCU の再帰は CLIQUES よりも浅くなり，ほとんどのグラフで 1/10 以下となった．特に $V_2$ が $V_1$ より小さいグラフでは再帰の深さの差はより顕著となり，$n = 256,000$，$m = 2,560$，$r = 10$ 及び $n = 128,000$，$m = 1,280$，$r = 20$ の局所的ランダムグラフでは CLIQUES は使用メモリが大きくなり過ぎ実行できなかった．以上より，BCU は CLIQUES よりも必要となるメモリが非常に小さいと言

表 1: 極大 2 部クリーク全列挙アルゴリズムの実行結果

(a). ランダムグラフに対する実行結果

| $n$ | $m$ | $p$ | $\omega$ | ♯cliques | CLI. [sec] | BCU [sec] | CLI. dep. | BCU dep. |
|---|---|---|---|---|---|---|---|---|
| 1,000 | 1,000 | 0.1 | 131-143 | $1.1 \times 10^7$ | 111.2 | **92.1** | 106-122 | **8** |
| 1,000 | 1,000 | 0.2 | 241-251 | $2.4 \times 10^9$ | 21,773.4 | **18,280.2** | 208-227 | **10-11** |
| 3,000 | 3,000 | 0.01 | 51-57 | $1.5 \times 10^5$ | 50.9 | **14.7** | 43-51 | **5** |
| 3,000 | 3,000 | 0.1 | 362 | $3.6 \times 10^9$ | 52,987.3 | **44,311.7** | 330 | **9** |
| 5,000 | 5,000 | 0.001 | 16-18 | $9.7 \times 10^3$ | 78.7 | **3.2** | 14-16 | **4** |
| 5,000 | 5,000 | 0.01 | 77-83 | $9.5 \times 10^5$ | 333.5 | **125.3** | 64-76 | **5** |
| 3,000 | 30 | 0.3 | 936-978 | $4.1 \times 10^5$ | 9.6 | **9.5** | 872-923 | **10** |
| 3,000 | 30 | 0.4 | 1.238-1,268 | $3.1 \times 10^6$ | 39.0 | **37.3** | 1.141-1,269 | **12** |
| 10,000 | 100 | 0.1 | 1,054-1,080 | $9.1 \times 10^5$ | 74.3 | **50.4** | 961-1,061 | **8** |

(b). 局所的ランダムグラフに対する実行結果

$r = 10$

| $n$ | $m$ | dens | $\omega$ | ♯cliques | CLI. [sec] | BCU [sec] | CLI. dep. | BCU dep. |
|---|---|---|---|---|---|---|---|---|
| 16,000 | 16,000 | 0.00066 | 20-22 | $3.4 \times 10^5$ | 4,419 | **81** | 19-21 | **7-8** |
| 32,000 | 32,000 | 0.00033 | 20-22 | $6.9 \times 10^5$ | 29,350 | **348** | 20 | **7-8** |
| 64,000 | 64,000 | 0.00016 | 20-22 | $1.4 \times 10^6$ | >100,000 | **1,519** | | **7-8** |
| 16,000 | 160 | 0.066 | 1,102-1,116 | $4.2 \times 10^6$ | 268 | **143** | 1,025-1,085 | **12-13** |
| 64,000 | 640 | 0.016 | 1,115-1,131 | $1.7 \times 10^7$ | 3,675 | **679** | 1,022-1,075 | **12-13** |
| 256,000 | 2,560 | 0.0041 | 1,129-1,136 | $6.7 \times 10^7$ | – | **3,647** | | **12-13** |

$r = 20$

| $n$ | $m$ | dens | $\omega$ | ♯cliques | CLI. [sec] | BCU [sec] | CLI. dep, | BCU dep, |
|---|---|---|---|---|---|---|---|---|
| 16,000 | 16,000 | 0.0013 | 33-35 | $3.3 \times 10^6$ | 5,061 | **234** | 30-33 | **9** |
| 32,000 | 32,000 | 0.00064 | 34-35 | $6.7 \times 10^6$ | 38,625 | **860** | 33 | **9** |
| 64,000 | 64,000 | 0.00032 | 34-35 | $1.3 \times 10^7$ | >100,000 | **4,136** | | **9** |
| 32,000 | 320 | 0.064 | 2,123-2,144 | $1.7 \times 10^9$ | 14,465 | **12,017** | 2,018-2,126 | **15-16** |
| 64,000 | 640 | 0.032 | 2,148 | $3.4 \times 10^9$ | 35,655 | **29,597** | 2,070 | **15** |
| 128,000 | 1,280 | 0.016 | 2,153 | $6.7 \times 10^9$ | – | **64,693** | | **15** |

える．BCU の再帰が浅いのは手続き EX′_D1 により $V_1$ 及び $V_2$ という $G'$ 中のサイズの大きなクリークを抽出しないようにし，手続き EXPAND′ において再帰の浅いうちに $CAND \subseteq V_1$ または $CAND \subseteq V_2$ となるようにしているためと考えられる．$V_2$ が小さいグラフで再帰の深さの差が顕著なのは，EXPAND′ により $CAND$ から $V_2$ 中の節点が無くなりやすいためと考えられる．

BCU は実行時間もほとんどのグラフで CLIQUES より小さくなり，特に疎なグラフ程その差は顕著となったが，疎でないグラフでは逆に実行時間の差が小さくなった．これは，CLIQUES では $Q$ に節点を 1 つ加える度に $Q$ に加える候補節点から $\Gamma'(u)$ 中の節点を除く操作を行っているが，BCU では $u$ を $Q$ に加えた場合，更に $Q$ に $u$ に隣接している節点を加える際にはこのような操作を行わないため，グラフが密である程 CLIQUES の方が効果の大きい分枝限定を行うからと考えられる．

## 5 まとめ

極大クリーク全列挙アルゴリズム CLIQUES を基にした極大 2 部クリーク全列挙アルゴリズム BCU を提唱し，計算機実験により性能を評価した．実験結果より，BCU は CLIQUES よりも使用するメモリが非常に小さく，かつ全体的に高速であることを確認した．これより，BCU は CLIQUES では実行できない大規模問題にも適用可能であると期待できる．今後は他の極大 2 部クリーク全列挙アルゴリズムとの性能比較や実問題に対する応用が課題となる．

**謝辞**. 本研究に非常に有用なコメントをいただいた院生の須谷洋一氏，貴重なご意見をいただいた京大バイオインフォマティクスセンター阿久津達也教授及び北大・コンピュータサイエンス専攻原口誠教授に感謝します．なお，本研究は科学研究費補助金基盤研究 (B) の支援を受けている．

## 参考文献

[1] T. Uno, T. Asai, Y. Uchida and H. Arimura, "LCM: An efficient algorithm for enumerating frequent closed item sets," Proceeding of IEEE ICDM'03 Workshop FIMI'03 (Available as CEUR Workshop Proceedings Series, Vol.90, http://ceur-ws.org/vol-90) (2003).

[2] 岡田吉史, 藤渕航, ホートン・ポール, "極大 2 部クリーク列挙法による遺伝子発現モジュールの抽出," 情処研報, 2006-BIO-6, pp.17-23 (2006).

[3] K. Makino and T. Uno, "New algorithms for enumerating all maximal cliques," SWAT 2004, LNCS 3111, pp.260-272 (2004).

[4] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments," Theoret. Comput. Sci. 363, pp.28-42 (2006).

[5] 仲川崇史, 富田悦次, "極大クリーク全列挙アルゴリズムの実験的比較評価," 情処研報, 2004-MPS-52, pp.41-44 (2004).