最大クリーク問題を解く高速なアルゴリズムの提案と従来の手法との実行時間比較実験

A fast algorithm for solving Max-Clique problem

谷　研究室　　原田　英幸
Hideyuki Harada

概要

最大クリーク問題は NP 困難に属する組合せ最適化問題であり，巨大なグラフに対しては計算時間が莫大で実質計算不能とされている．この問題を解くアルゴリズムを高速化することにより，今まで解けなかったようなグラフを解くことを目標とする．グラフアルゴリズムは応用分野が広く，多くの人の役に立つ研究である．

1　はじめに

最大クリーク問題とは，単純無向グラフ中で完全グラフを誘導する頂点集合のうち，頂点数が最大のものを見つける最適化問題の一種である．
最大クリーク問題は NP 困難に属するので多項式時間で解くことはできないが，指数関数時間内でもできるだけ速く解を得るアルゴリズムの開発が盛んである．
本研究では中央大学の江口出氏の研究結果を受け，電気通信大学の富田悦司教授の開発したアルゴリズムよりも高速なアルゴリズムを開発することを目指す．

2　定義

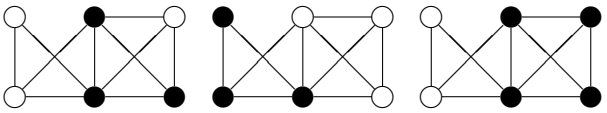
2.1　記号表記

任意のグラフ G に対して以下のように表記を行う
$V(G)$: グラフ G の頂点集合
$E(G)$: グラフ G の辺集合
$\Gamma(p)$: 頂点 $p \in V(G)$ に隣接する頂点の集合．近傍と呼ぶ．
$\mathrm{degree}(v)$: 頂点 v に隣接する頂点の数．次数と呼ぶ．

2.2　予備知識

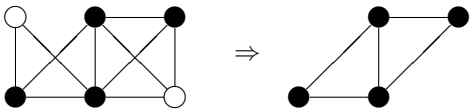
クリーク: 完全グラフを誘導する頂点集合
極大クリーク: 他のクリークの真部分集合でないクリーク
最大クリーク: グラフ中で頂点数が最大のクリーク
点誘導部分グラフ: $V' \in V(G)$ なる V' と，$E' = \{(u,v) \in E(G) | u \in V' \wedge v \in V'\}$ なる E' で構成される G の部分グラフ $G' = (V', E')$

例：



(a)　　(b)　　(c)

(a) 極大でないクリーク
(b) 極大クリーク
(c) 最大クリーク



塗りつぶされた頂点の集合 V' による点誘導部分グラフ

3　既存アルゴリズム

3.1　基本アルゴリズム dfmax

現在保持するクリークを Q としたとき，Q の頂点全てに隣接する頂点の集合を候補節点集合と呼び，R で表す．
R 中の頂点を適当に選び，Q に付け加えた上で新たな候補接点集合 R_p を構成し，Q と R_p に対して同様の操作を再帰的に適用する．Q_{max} には現在発見した極大クリーク中で最大サイズのものを保存する．

まずは候補接点集合 $R = V(G)$，$Q = \phi$ で初期化し、R と Q に対して以下の操作を行う．

1. $p \in R$ を一つ選び出す
2. $Q := Q \cup \{p\}$
3. $R_p := R \cap \Gamma(p)$
   (a) $R_p \neq \phi$ ならば R_p と Q に対して 1. からの操作を再帰的に適用する

(b) $(R_p = \phi) \wedge (|Q| > |Q_{max}|)$ ならば $Q_{max} := Q$

4. $R := R - \{p\}$, $Q := Q - \{p\}$

この操作を $R = \phi$ となるまで繰り返す.

```
dfmax(R, Q){
  while R ≠ φ do
    p := a vertex in R;
    Q := Q ∪ {p};
    R_p = R ∩ Γ(p);
    if R_p ≠ φ
      dfmax(R_p, Q);
    end if
    else if |Q| > |Q_max|
      Q_max = Q;
    end if
    Q := Q − {p};
    R := R − {p};
  end while
}
```

3.2 富田教授のアルゴリズム MCQ

頂点集合に対する彩色数は，頂点集合のクリークサイズの上界である．
候補接点集合を作成する際に近似彩色を行い，候補接点集合からそれぞれの頂点を選び出した場合にできるクリークサイズの上界を，各頂点に割り振る．
これにより得られた上界をもとに，探索を打ち切る．
実際に候補節点集合を探索する場合，頂点数に対して指数関数時間が必要だが，近似彩色は多項式時間で探索すべきかの判断が可能である．[1]

候補接点集合を作成したとき，各候補節点 p に以下の条件を満たす正整数 $N[p]$ を割り当てる．$(u, v) \in E(G)$ ならば $N[u] \neq N[v]$ $N[p] = k > 1$ ならば $N[v] = 1, 2, \ldots, k-1$ なる $v \in \Gamma(p)$ が，1 つずつ存在する．

候補節点集合を次数について昇順に整列し，候補節点集合の先頭から順に，頂点の近傍中の頂点にまだ割り振られていない数で最小の正整数を割り振る．

```
NUMBER-SORT(R, N){
  C_1 := φ; C_2 := φ;
  maxno := 1;
  while R ≠ φ do
    p := first vertex in R;
    k := 1;
    while C_k ∪ Γ(p) ≠ φ do
      k + +;
    end while
    if k > maxno then
      maxno := k;
      C_{maxno+1} := φ;
    end if
    N[p] := k;
    C_k := C_k ∪ {p};
    R := R − {p};
  end while

    i := 1;
  for k := 1 to maxno do
  for j := 1to |C_k| do
  R[i] := C_k[j];
  i + +;
  end for
  end for
}

EXPAND(R, N){
  while R ≠ φ do
    p := first vertex in R;
    if |Q| + N[p] > |Q_max|
      Q := Q ∪ {p};
      R_p := R ∩ Γ(p);
      if R_p ≠ φ
        NUMBER-SORT(R_p, N′);
        EXPAND(R_p, N′);
      end if
      else if |Q| > |Q_max|
        Q_max := Q;
      end if
      Q := Q − {p};
      R := R − {p};
    end if
    else return;
  end while
}
```

3.3 江口氏の研究 CLIQUE

候補節点集合を降順で整列し，先頭から選んでクリークに加えていく．追加する頂点の次数を $degree(p)$ とす

るとき，$|Q| + degree(p) < |Q_{max}|$ となるならば探索を打ち切る．
現在発見した最大クリークサイズ以下の次数の頂点を削除し，グラフサイズを縮小する．
これを富田教授のアルゴリズム MCQ に追加して，アルゴリズムの高速化を図る．
本項では，詳しい内容は割愛する．

4 アルゴリズム改良案

4.1 隣接ノード上界

各ノードが持つ情報を増やし，探索の際の手がかりとする．ここでは「そのノードを含むクリークのサイズの上界」を付加する．

$$addeg(v) = \min\{\max\{degree(p)|p \in \Gamma(v)\}, degree(v)\}+1$$

すなわち、各頂点自身とその近傍からなる頂点集合を次数についてに降順で整列し，$i > degree(node[i])$ となる最小の i を見つける．$degree(node[i]) + 1$ は $R \cup \{v\}$ で誘導されるサブグラフにおける，頂点 v を含んだクリークサイズの上界となる．
これを候補節点集合を作成するたびにその全ての頂点について求め，降順で候補節点集合を整列し，先頭から頂点を選び出してクリークに追加していく．
また，探索打ち切りの基準にも使用する．

```
SET_ADDEG(R, addeg){
  for all v in R
    L := Γ(v) ∪ {v};
    sort L with decending order by degree;
    while i < degree(i) ∧ i < |L| do
      i++;
    end while
    addeg(v) := degree(i) + 1;    end for
}
```

4.2 候補節点集合に対する上界

サイズ n のクリークはそのノードを含むクリークサイズの上界が n 以上のノードが n 個以上必要である．この性質を利用して候補節点集合内でできるクリークサイズの上界を求める．
次数 +1 はそのノードを含むクリークサイズの上界である．候補節点集合を格納する配列 R を $addeg$ についてに降順で整列し，$i > addeg(R[i])$ となる最小の i を見つける．$addeg(R[i])$ は R における最大クリークサイズの上界となる．
この操作は非常に短時間で実行可能であるので，候補節点集合から頂点を取り除くたびに適用しても平気である．

```
UPPER(up, R, addeg){
  while up < addeg(up) ∧ up < size of R do
    up++;
  end while
}

ADDEG(R, N){
  SET_ADDEG(R, addeg);
  sort R with decending order by addeg;
  while R ≠ φ do
    up :=UPPER(up, R);
    p := first vertex in R;
    if |Q| + addeg(up) > |Q_max|
      Q := Q ∪ {p};
      R_p := R ∩ Γ(p);
      if R_p ≠ φ
        ADDEG(R_p, Q);
      end if
      else if |Q| > |Q_max|
        Q_max := Q;
      end if
      Q := Q − {p};
      R := R − {p};
      up − −;
    end if
    else return;
  end while
}
```

5 実行結果

C++クラスライブラリ LEDA4.5 を利用し，上記の 3 つのアルゴリズムを実装し，フリーのベンチマークグラフ集「DIMACS ベンチマークグラフ」のいくつかのグラフを解いて比較実験を行った．
それぞれのアルゴリズムによる実行時間については後のページに表で記す

実行環境
OS Linux version 2.6.9-1.6_ FC2
CPU Intel Pentium4 2.66GHz
RAM 1024MB

コンパイラ gcc 3.3.3

参考文献

[1] E. Tomita and T. Seki, ”An efficient branch-and-bound algorithm for finding a maximum clique. ” Discrete Mathematics and Theoretical Computer Science, Dijon, France, Lecture Notes in Computer Science 2731, pp.278-289 (2003).

graph				used time by algorithms(sec)			
file	$\|V\|$	$\|E\|$	$\|Q_{max}\|$	density	dfmax	MCQ	ADDEG
MANN_a9	45	918	16	0.93	14 min	0.92	20min
keller4	171	9435	11	0.64	7 min	39.26	4min
c-fat200-2	200	3235	24	0.16	7 min	0.07	0.11
c-fat200-5	200	8473	58	0.43		1.06	0.19
p_hat300-1	300	10933	8	0.16	5.3	2.4	0.94
c-fat500-2	500	9139	26	0.07	Over 1h	0.4	0.25
c-fat500-5	500	23191	64	0.19		3.72	0.48
c-fat500-10	500	46627	1126	0.37		26.29	1.6
p_hat1000-1	1000	122253	10	0.24		26 min	4 min
p_hat1500-1	1500	284923	11	0.25		Over 2h	49 min