4F-3

密なグラフに有効で単純な最大クリーク抽出アルゴリズム

卜部 昌平 †，富田 悦次 †，森田 昭広 ††

電気通信大学　† 電気通信学部 情報通信工学科，†† 情報システム学研究科 情報システム設計学専攻
〒 182-8585 東京都調布市調布ヶ丘 1-5-1

1　はじめに

筆者らが提案した最大クリーク抽出アルゴリズム MCQ[1] において，分枝限定のための前処理として行われている近似彩色処理がアルゴリズム全体の中でかなりの実行時間を占めていることが，実験により明らかになっている [2]．そこで本稿では，これまでよりも高速な近似彩色アルゴリズムを提唱する．このアルゴリズムは非常に単純で実装が容易である上に，枝密度が高いグラフに対して効率的に動作する．このアルゴリズムを組み込むことにより，最大クリーク抽出アルゴリズム全体の処理速度を高速化することに成功した．

2　アルゴリズム PAC

文献 [3] で提案されている近似彩色アルゴリズム ABSORB は，ある色 c に彩色できる節点を入力の節点集合から順に抜き出してゆき，これ以上彩色できないという状況になると次の色の彩色に移り，最終的に全ての節点が彩色されたときに停止する，という方法をとる．ここで，ある節点 p に色 c を彩色した時点で，p に隣接する節点はすべて，もう c で彩色することができないことが明らかであるので，そのような節点は，彩色の候補から外すことができる．この発想に基づくアルゴリズムが PAC である．以下に PAC の具体的な手順を示す．なお，以下，すべての節点集合は全順序集合であるとし，その和集合における順序は元の集合に対する演算記述の順序に従うものとする．

1. ある色 c で彩色中であるとする．今節点 p に彩色したとすると，隣接節点集合 $\Gamma(p)$ に含まれる全ての節点は色 c で彩色することができないので，彩色候補集合から外してしまう．すなわち，未彩色候補節点集合を R と置くと，今回彩色した p は彩色済み節点集合 C に，$\Gamma(p)$ 中の全ての節点は彩色対象外節点集合 B にそれぞれ移動し，p と $\Gamma(p)$ を除いたものを新たに R とおき直す．これを $R = \emptyset$ となるまで繰り返したときに，色 c の彩色が完了したとする．
2. 上の手順に従い，彩色対象の節点集合 R から独立節点集合 C および $B(= R - C)$ の 2 つの集合を生成する．C の要素は彩色済みであるから戻り値の集合に追加し，B を新たに R と置き直す．以上の手順を $B = \emptyset$ となるまで再帰的に繰り返すと，与えられた R 中のすべての節点に彩色が完了する．

```
procedure PAC(R, N)
    c := 1;
    make a list L from R;
    N := ∅;
    R := ∅;
    while L is not empty do
        make an orderd set W from L;
        while W ≠ ∅ do
            p := W[1];
            R := R∪{p};
            N := N∪{c};
            delete p from L;
            W := W − Γ(p) − {p}
        od;
        c := c + 1
    od
end.
```

図 1　アルゴリズム PAC

実際に手続き型プログラミング言語でこのアルゴリズムを記述する際には，節点集合からの節点の削除が頻繁に発生するため，単に順序付き集合を配列として実現するよりも，集合を線形リストとして表現した方が高速になる．この点を考慮して記述したアルゴリズムを図 1に示す．

3　計算機実験

今回提唱するアルゴリズム PAC の有効性を確認するため，C 言語により実働化し，計算機実験を行った．実験では文献 [2] のアルゴリズム MCQ′ を基本とし，(1) この彩色部分を変えないもの，(2) 彩色部分のみを ABSORB に変更したもの（これは文献 [3] の MCQ′′ と同じである），および (3) 彩色部分のみを PAC に変更したもの，の 3 つの速度を比較することで行った．これらのアルゴリズムは入出力関係が全く同じであるから，その違いは実行時間のみである．

表 1にランダムグラフ上での各アルゴリズムの実行速度を示す．グラフは節点数が全て 200 で，枝が確率 p の一様乱数に従って分布するグラフ群である．なお実験はそれぞれの枝密度で乱数の種を変えて生成した 10 のグラフに対して行い，その実行時間の平均を取った．使用した計算機とコンパイル環境は文献 [1, 2, 3] と同一のものである．

表より，枝密度が低い領域においては MCQ′ が最も高速であるが，ここで提唱するアルゴリズムとの差は非常にわずかで

A Simple Algorithm for Finding a Maximum Clique that is Efficient for Dense Graphs

Shouhei Urabe†, Etsuji Tomita†, Akihiro Morita††
†Dept. of ICE, UEC. ††Dept. of IS, UEC.
Chofugaoka 1-5-1, Chofu, Tokyo 182-8585, Japan.

表 1　ランダムグラフに対する実行時間 [sec]

n	p	MCQ′[2]	MCQ′′[3]	MCQ′+PAC
200	0.0	**0.000107**	0.000254	0.000136
200	0.1	**0.000808**	0.000846	0.000812
200	0.2	**0.00125**	0.00132	0.00127
200	0.3	0.00250	0.00268	**0.00248**
200	0.4	0.00619	0.00670	**0.00592**
200	0.5	0.0217	0.0230	**0.0200**
200	0.6	0.124	0.128	**0.110**
200	0.7	0.946	0.932	**0.812**
200	0.8	17.46	16.80	**13.81**
200	0.9	941.87	896.03	**615.31**
200	1.0	0.000257	0.000248	**0.000166**

あり，逆に枝密度の高い領域では提唱するアルゴリズムが高速であることがわかる．枝密度の高いグラフに対する高速性は他のアルゴリズムに対して顕著である．また，枝密度が低いグラフに対しても極端な速度低下を引き起こしていない．このことから，PAC を使ったアルゴリズムが一般に優位であると言える．

また，表 2には同じ計算機環境上で DIMACS[4] によるベンチマークインスタンスを解いた場合の各アルゴリズムの実行速度を示す．ほとんどのインスタンスにおいて提唱するアルゴリズムが一番高速であることが分かる．PAC を用いて遅くなったグラフも存在するが，遅くなったと言ってもその差は 1/10 秒以下であり，ほとんど差がないと言ってよい．従って，表 1，2より PAC を用いた最大クリーク抽出アルゴリズムが最も高速であると結論付けることができる．

4　まとめ

近似彩色アルゴリズム PAC を用いた最大クリクー抽出アルゴリズムを考案・実働化し，計算機実験によりこのアルゴリズムが従来知られている他のアルゴリズムに比べて一般的に高速であることを示した．提唱するアルゴリズムはその近似彩色部分のみが従来と異なるため，同様の変更を加えることで重み付き最大クリーク抽出などのアルゴリズムにも適用可能である．

今回提唱するアルゴリズムは枝密度が高いグラフのみに特化したアルゴリズムではない．枝密度が高い場合に特化したアルゴリズムとして COCR[5] があるが，COCR は非常に複雑であり，枝密度がさほど高くない場合には高速とはいえない．一方提唱するアルゴリズムは枝密度の低いグラフに対しても，従来のアルゴリズムよりもわずかに遅いかほぼ同等の速度で実行される．アルゴリズムが入力に対してロバストであることは現実問題への応用を考えた際には極めて重要であることから，今回提唱するアルゴリズムはその入力に対するロバスト性にも注目すべきである．ただし，僅かながら実行時間が遅くなるようなグラフも存在するため，提唱するアルゴリズムがどのような入力に対しても全面的に高速であるとまで主張するには至っていない．この点は今後の課題である．

謝辞　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．

表 2　DIMACS グラフに対する実行時間 [sec]

Graph name	MCQ′[2]	MCQ′′[3]	MCQ′+PAC
brock200_1	2.39	2.05	**1.98**
brock200_2	0.015	0.014	**0.0132**
brock200_3	0.071	**0.06**	**0.063**
brock200_4	0.29	0.27	**0.255**
c-fat500-2	0.0063	0.0058	**0.00543**
c-fat500-5	0.017	0.016	**0.0149**
c-fat500-10	0.0338	0.032	**0.0274**
hamming6-2	0.00011	0.00008	**0.000073**
hamming6-4	0.000098	**0.00002**	0.000097
hamming8-2	0.0036	0.0030	**0.00194**
hamming8-4	0.28	**0.26**	**0.260**
hamming10-2	0.33	0.16	**0.093**
johnson8-2-4	**0.000021**	0.000027	**0.000021**
johnson8-4-4	0.00055	0.00044	**0.000447**
johnson16-2-4	0.21	0.18	**0.197**
keller4	0.037	**0.03**	**0.0345**
MANN_a9	0.00018	0.00014	**0.000135**
MANN_a27	4.22	3.44	**1.87**
MANN_a45	6,212.62	2,134.63	**1,172.03**
p_hat500-1	0.038	0.04	**0.0372**
p_hat500-2	4.49	3.63	**3.20**
p_hat500-3	2,662.54	2,201.61	**1,790.34**
p_hat700-1	0.14	0.14	**0.132**
p_hat700-2	64.29	52.35	**44.61**
p_hat1000-1	0.75	0.74	**0.718**
p_hat1000-2	3,512.07	2,870.89	**2,477.71**
p_hat1500-1	6.32	6.35	**6.01**
san400_0.5_1	0.029	0.03	**0.0306**
san400_0.7_1	2.34	1.97	**1.764**
san400_0.7_2	0.45	0.39	**0.372**
san400_0.7_3	4.53	**4.14**	4.19
san400_0.9_1	5.32	4.51	**3.25**
san1000	7.14	9.09	**8.86**
sanr200_0.7	0.79	0.69	**0.674**
sanr400_0.5	1.12	1.06	**1.030**

参考文献

[1] E. Tomita, and T. Seki, “An efficient branch-and-bound algorithm for finding a maximum clique,” DMTCS2003, LNCS 2731, pp.278–289 (2003).

[2] 亀田宗克，富田悦次，“最大クリーク抽出アルゴリズムの高速化と解析・評価，” 情処研報，AL-93，pp.33–40 (2004).

[3] 森田昭広，富田悦次，亀田宗克，“最大クリーク抽出のより高速なアルゴリズム，” 情報科学技術レターズ (FIT2004)，pp.19–22 (2004).

[4] D.S. Johnson, and M.A. Trick, eds., Cliques, Coloring, and Satisfiability, vol.26 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science (1996).

[5] E.C. Sewell, “A branch and bound algorithm for the stability number of a sparse graph,” INFORMS J. Comput., vol.10, no.4, pp.438–447 (1998).