グラフ彩色問題の厳密解アルゴリズム

辻旭人, 富田悦次
電気通信大学大学院 電気通信学研究科 電子情報学専攻

あらまし　グラフ彩色問題の効率の良いアルゴリズムの１つとして, DSATUR+ [2] がよく知られている. 本稿では, DSATUR+のパラメータ設定に関する問題点等を主として改善したアルゴリズムを提唱し, いくつかのグラフを対象とした計算機実験により, その有効性を示す.

An Exact Algorithm for Graph Coloring

Akito TSUJI and Etsuji TOMITA
The Course in Communications and Systems Engineering,
The Graduate School of Electro-Communications, The University of Electro-Communications

Abstract　DSATUR+ [2] is known as one of the efficient algorithms for the exact graph coloring problem. In this note, we present an algorithm which improves DSATUR+ by mainly expelling the setting of some parameter. The proposed algorithm is shown to be effective by computer experiments for some graphs.

1　はじめに

グラフ彩色問題は最も重要な組合せ最適化問題の一つであり, 多くの実問題に適用される. しかし, グラフ彩色問題は NP 困難のクラスに属しており, 多項式時間内で厳密解を得られるアルゴリズムはほぼ期待し難く, 非常に長実行時間を要する. そこで, 多項式時間的でないにしても, より高速な厳密解アルゴリズムが望まれる.

ところで, 現在よく知られていて, 効率の良い厳密解アルゴリズムの１つとして, DSATUR [1] を基本とし, それを効率化した DSATUR+ [2] がある. しかし, DSATUR+には, その中で適用する近似彩色アルゴリズムのパラメータの 1 つの設定が困難であるという問題点がある. そこで, 本稿ではその問題点を改善したアルゴリズム, 及びそれを更に効率化したアルゴリズムを提唱し, その有効性を計算機実験により明らかにする.

2　アルゴリズム DSATUR [1]

厳密彩色アルゴリズム DSATUR * の動作を以下に示す.

1. 最初の色数を節点の数− 1 とする.
2. 彩色されていない節点集合の中で, 隣接節点に割り当てられた異なる色の数 (saturation degree) が最大である節点を選ぶ. そのような節点が複数の場合には, その中で色が割り当てられていない節点集合により構成された誘導部分グラフ中, 次数最大のものを選ぶ.
3. 選んだ節点に { (グラフのいずれかの節点に割り当てられていて, かつ隣接節点に割り当てられていない色)+(可能ならば新たな１色)} の中で色番号が最小であるものを割り当てる.
4. ２. と３. を繰り返す. ３. で割り当てる色が無くなった場合には, 1 つ前の節点に戻り, 別の色で色番号が最小なものを割り当てて, 再度２. と３. を繰り返す.
5. 全部の節点に対して色が割り当てられたら, その色数よりも１つ少ない色数で２. ～４. を試みる. これをその色数では彩色できないことがわかるまで繰り返す.

この探索の過程は, 前に選ばれた節点に関する順序対 (節点番号, 割り当てた色) とその次に選ばれた節点に関する順序対 (節点番号, 割り当てた色) を親子関係の枝で結んで得られる木として表現される. この探索木における枝を分枝, その枝数を分枝数と呼ぶ.

3　アルゴリズム DSATUR+ [2]

DSATUR は, 以下のような手法で効率化す

*最初に, DSATUR は近似解法版 [1] の名前として用いられたが, その後厳密解法版の名前としても用いられている [2] . 本稿では, DSATUR は厳密解法版 [2] の名前を意味する.

ることができる.

- DSATURの適用の前に, 下界として最大クリークを抽出し, その節点に固定して色を割り当てておく. このようにすると, DSATURの手続きの際にそれらの節点の再彩色を試みる必要はなくなり, またそれらの節点に隣接している節点に割り当て可能な色数も減少するので, 分枝数の減少を期待できる.

- 高速な近似彩色アルゴリズムを使って, ある程度の精度まで上界を求めてからDSATURを適用した方が効率が良くなる.

以上の効率化手法を使った具体的な改良アルゴリズムの1つが文献[2]のDSATUR+である. 動作を以下に示す.

1. 節点次数の総和が最大である最大クリークを抽出する.
2. 近似彩色アルゴリズムRLF [3]を適用し, ある程度の精度の近似彩色数を求める.
3. 2. で求めた近似彩色数－1を指定彩色数kとして近似彩色アルゴリズムTABUCOL [4]を適用する. その答がfalse (k色で彩色失敗) ならばk+1を上界とし, true (k色で彩色成功) ならば1つずつ少ない色数をkとして答がfalseになるまでTABUCOLの適用を繰り返す.
4. 1. で抽出した最大クリークの節点を固定して彩色する.
5. 3. で求めた上界－1からDSATURを適用する.

DSATUR+の問題点は, 与えられたグラフに対するTABUCOLの解の探索の回数$nbmax$の設定である. $nbmax$の値が大きすぎるとTABUCOLに時間がかかり, 小さすぎると精度の良い上界が見つからず, 結果的にDSATURの手続きで時間がかかる. すなわち, DSATURを単独で適用した場合に分枝数が少なくて済むような“簡単な”グラフには$nbmax$を小さくしてTABUCOLにあまり時間をかけず, 他方, 分枝数がある程度多くなる“難しい”グラフには$nbmax$を適度に大きくし, 多少TABUCOLに時間をかけても精度の良い上界を求めたい. しかし, 同じ節点数, 枝の存在確率のグラフであっても, そのグラフのseed (乱数の種) や枝の偏り方の違いによって分枝数は大きく変ってくる. 従って, グラフが与えられたときに, その分枝数を予測するのは困難であり, 結果として$nbmax$の効率の良い設定は困難である.

4 アルゴリズムS-DSATUR+

本稿で提案するS-DSATUR+はDSATUR+を改良し, 全てのグラフに対し同じパラメータで良好な結果が得られるようにしたアルゴリズムである. 動作を以下に示す.

1. MCLIQ [5]を部分変更したアルゴリズムにより, 節点次数の総和が最大である最大クリークを抽出する.
2. RLFを適用し, 上界を求める.
3. 1. で抽出した最大クリークの節点を固定して彩色する.
4. 2. で求めた上界－1からDSATURを適用する.
5. (アルゴリズム開始からの) 分枝数がある定められた値 (*TABUCOL number*と呼ぶ) の1つに達したら, その時点の色数をkとし, 分枝数に応じて設定した$nbmax$の値でTABUCOLを適用する. その答がfalseならばそのままDSATURを続ける. trueならば1つずつ少ない色数をkとして答がfalseになるまでTABUCOLを適用し, 最終的に, 求まった上界－1より再びDSATURを適用する.
6. 5. をDSATURが終了するまで繰り返す.

S-DSATUR+の特徴は, DSATURの手続きである程度分枝数が増加したら, その分枝数からグラフの“難しさ”を判断し, その“難しさ”に適した$nbmax$の値でTABUCOLを適用して新たな上界を求めることである. そのために, いくつかのグラフに対しDSATURとTABUCOLを適用した予備実験を行い, *TABUCOL number*とその際の$nbmax$の値を設定する.

S-DSATUR+は分枝数の増加に従い, 小さい$nbmax$からTABUCOLを適用していくの

で, 適切な $nbmax$ で TABUCOL を適用するまでに多少の時間がかかるが, そのグラフに不適切な大きな $nbmax$ で適用することは少ない. よって, どんな規模, 種類のグラフに対してもパラメータを変えずに妥当な時間で結果を得ることができる.

5 S-DSATUR+の効率化

5.1 S-DSATUR+2

S-DSATUR+中の DSATUR の手続きにおいて, 次に色を割り当てる節点の選び方は,

1. saturation degree が最大である節点を選ぶ.
2. そのような節点が複数の場合は, その中で色が割り当てられていない節点集合により構成された誘導部分グラフ中, 次数最大のものを選ぶ.

となっている. これに 3 つ目の選択基準

3. さらにそのような節点が複数の場合は, その中でまだ色が割り当てられていない隣接節点に割り当てられる色数の合計を一番減少させる節点を選ぶ.

を追加したものを S-DSATUR+2 とする. そのために, 各々の候補節点 v について, v に割り当て可能な各々の色 c に対し, v の未彩色である隣接節点で c を割り当て可能なものの個数を計算し, その総和をとる. そして, その総和が最大となる節点を選ぶ.

このとき, 3. は 1. や 2. と比べ多少計算時間がかかるので, 毎回 3. を行うと, 全体としての実行時間が長くなる. そこで, 探索木のある深さ (探索木の最大深さは節点数と等しいので, 具体的には節点数の何分の 1 と設定する) まで 3. を行う.

これは [2] で提案されている DSATUR+に対する同様の効率化手法を, S-DSATUR+の効率化となるように変更したものである.

5.2 S-DSATUR+3

DSATUR の手続きでの深さ優先探索において, 「次に彩色する節点」を選択する第 1 基準を「saturation degree が最大である節点」として効率化が行われている. つまり, DSATUR の手続きの前に固定して彩色される最大クリークを, DSATUR の手続きで次々に選ばれていく「次に彩色する節点」の saturation degree が大きくなるようなものにすれば, アルゴリズムの効率が良くなると考えられる.

S-DSATUR+2 では次数の総和が最大である最大クリークを抽出していたが, それをその最大クリークのみを固定して彩色したとき (即ち, DSATUR の手続きを適用する直前の状態),

$$\sum_{v\in \text{全節点}-\text{最大クリーク}} (v\text{のsaturation degree})\times(v\text{の次数})$$

の値が最大となる最大クリークを抽出するように変更したものを S-DSATUR+3 とする. 上の式は, saturation degree が大きい節点 (即ち, 探索木において根に近い所で選ばれる可能性の高い節点) が他の節点と多く隣接している場合に大きくなる. この値が大きい最大クリークを抽出することによって, 探索木において根に近い所で選ばれる節点がそれ以後に選ばれる節点の saturation degree を大きくすることが期待できる.

6 計算機実験

各アルゴリズムを計算機実験により比較する.

使用計算機の CPU は Athlon 1.2GHz, OS は Linux である. プログラミング言語は C を使い, コンパイルは gcc -O2 で行った.

ランダムグラフの seed の異なるもの 5 つと, 作為的に枝に偏りをもたせたグラフ $G10$ [6] (枝の密在部分が節点数の 10 分の 1 箇所ある) の seed の異なるもの 5 つを対象とした各アルゴリズムの実行結果を表 1 に示す. 表中の値はその計 10 個のグラフに対する実行時間の平均である. ここで, n は節点数, p は枝の存在確率, ω は最大クリークの節点数, χ は彩色数を表す.

S-DSATUR+と S-DSATUR+2, S-DSATUR+3 において, $TABUCOL\ number$ を $10^4 \times 10^{2i}(i = 0, 1, 2, ...)$ とし, その際の $nbmax$ を各々$10^4\times10^i(i = 0, 1, 2, ...)$ とした. S-DSATUR+2 と S-DSATUR+3 中の DSATUR の手続きで, 次に色を割り当てる節点を選ぶときの 3 つ目の選択基準による判定を行う深さは, (節点数)/5 までとした.

表 1: 各アルゴリズムの平均実行時間

Graph				DSATUR+ [2]				S-DSAT UR+	S-DSAT UR+2	S-DSAT UR+3
				$nbmax =$ 1,000	$nbmax =$ 10,000	$nbmax =$ 100,000	$nbmax =$ 1,000,000			
p	n	ω	χ	time(s)	time(s)	time(s)	time(s)	time(s)	time(s)	time(s)
0.1	60	4.2	4.5	**0.0**	0.1	2.7	29.2	0.0	0.0	0.0
	70	4.2	4.5	**0.0**	0.4	4.0	40.5	0.0	0.0	0.0
	80	4.5	4.9	**0.0**	0.7	7.3	73.0	0.3	0.3	0.3
	90	4.7	5.2	**0.0**	0.7	7.7	77.9	0.1	0.1	0.1
0.3	60	6.3	7.5	**0.0**	0.3	7.4	75.8	0.0	0.0	0.0
	70	6.7	8.3	**0.4**	1.4	11.3	113.2	0.9	0.8	0.8
	80	6.8	8.7	116.1	117.0	**11.1**	99.3	8.1	8.1	8.1
	90	6.9	9.5	319.3	**319.0**	330.2	443.1	323.6	312.3	228.8
0.5	60	8.6	11.0	**0.1**	0.4	7.7	75.6	0.4	0.4	0.4
	70	9.2	11.9	16.1	**8.5**	14.8	87.2	9.3	9.1	9.1
	80	9.7	13.1	2,517.1	2,506.7	1,397.6	**368.5**	303.0	274.4	274.4
0.7	60	12.1	15.6	**2.6**	3.1	9.1	75.0	3.0	2.8	2.8
	70	13.3	17.4	125.5	**120.3**	127.4	166.0	129.5	128.4	128.4
	80	13.8	18.8	4,008.8	3,894.7	**1,716.2**	1,728.1	1,759.9	1,642.8	1,600.2
0.9	60	23.5	25.6	**0.0**	0.9	10.3	106.4	0.1	0.1	0.1
	70	26.1	29.0	**1.8**	2.6	12.1	108.7	2.3	2.2	2.3
	80	27.5	31.4	45.8	44.3	**19.7**	110.5	18.8	18.0	18.0
	90	29.2	34.3	1,158.0	1,107.6	**625.9**	710.7	638.2	569.2	569.2

7 考察

表 1 より, DSATUR+は結果が TABUCOL の $nbmax$ の設定に左右され, その設定が悪いと, 設定の良いときに比べて大幅に効率が悪くなってしまうことが分かる. これにより, $nbmax$ の最適な値が予測できないことは非常に問題であることが明らかである. それに対し, S-DSATUR+では DSATUR+で TABUCOL の $nbmax$ を最適にした場合とほぼ同等の実行結果が得られている. 全てのグラフに対し同じパラメータでこの様な実行結果が得られており, S-DSATUR+は非常に有効であると言える.

S-DSATUR+2 は S-DSATUR+と比べ, 比較的実行時間がかかるところでその削減に成功している. また, S-DSATUR+3 は他のアルゴリズムで抽出する次数の総和が最大である最大クリークと同じ最大クリークを抽出することが多かったが, そうでない場合, たとえば, 90 節点で枝の存在確率 0.3 の場合などで実行時間の削減に成功していることが認められる.

8 まとめ

本稿で提案した S-DSATUR+が DSATUR+の問題点を解決したアルゴリズムであることを示した.

さらに, S-DSATUR+2 と S-DSATUR+3 が S-DSATUR+と比べて実行時間を削減できることを示した.

謝辞 御検討及び参考資料を提供いただいた本学若月光夫助手, Suwat 氏に感謝します.

参考文献

[1] D. Brelaz: “New methods to color the vertices of a graph,” Commun. ACM, 22, pp.251-256 (1979).

[2] E. C. Sewell: “An improved algorithm for exact graph coloring,” DIMACS Series in Discrete Mathmatics and Theoretical Computer Science, 26, pp.359-373 (1996).

[3] F. T. Leighton: “A graph coloring algorithm for large scheduling problems,” Journal of Research of the National Bureau of Standards 84, no.6, pp.489-506 (1979).

[4] A. Hertz, D. de Werra: “Using tabu search techniques for graph coloring,” Computing, 39, pp.345-351 (1987).

[5] 富田悦次, 今松憲一, 木幡康弘, 若月光夫: “最大クリークを抽出する単純で効率的な分枝限定アルゴリズムと実験的評価,” 信学論, J79-D-I, 3, pp.1-8 (1996).

[6] 富田悦次, 藤井利昭: “最大クリーク抽出の効率化手法とその実験的評価,” 信学論, J68-D, 3, pp.221-228 (1985).