グラフ彩色問題に対する局所探索法の実験的評価

大井智裕 * 山口一章* 増田澄男*

1 グラフ彩色問題

頂点集合 V，辺集合 E とする無向グラフ $G = (V, E)$ の各頂点 v に色を塗る．その時，各辺 $e_ij = (v_i, v_j)$ に対し，v_i，v_j には異なる色 c_p，c_q を塗る．全ての頂点が塗り終わったとき，色の彩色に使われる色が最も小さくなる数，彩色数 $\chi(G)$ とその塗り方を求める問題をグラフ彩色問題という．

図 1 の左のグラフが入力として与えられた場合，その最適解は彩色数 $\chi(G) = 3$ の時で，塗り方の一つは右のグラフのようになる．

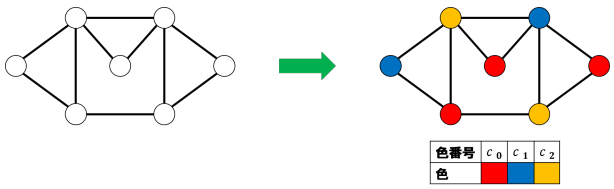

図 1: グラフ彩色問題の例

この問題は NP 困難であること [1] が知られており，電波塔の配置 [2] やスケジューリング問題 [3][4] など，さまざまな数理問題への応用が知られている．

しかし，グラフの頂点数が多くなると，正確な χ(G) を求めるためには，膨大な時間がかかる．よって，本稿ではより実用的な，さほど長くない時間で出来るだけよい解を求める方法，発見的手法を取り上げる．辺で繋がれた頂点同士が同じ色である状態を「衝突」と呼び，グラフの全ての頂点に色が塗られて，衝突がない塗り方は特に「合法な彩色」と呼ぶ．またある頂点と辺で繋がっている頂点を隣接頂点と呼び、隣接頂点と一つ以上の衝突を持つ頂点を，以下，衝突頂点と呼ぶ．この問題の発見的手法として，何らかの方法で，さほど多くない色数 k の合法な彩色を求めた後，$k-1$ 色の合法でない彩色を行い，頂点の色を変えていって頂点衝突を減らす方法がある．今回は，その頂点と色の選び方に 2 つの方法を提案する．

*神戸大学大学院

2 SEQ 法

前述の方法では，まず，与えられたグラフの合法な彩色が必要である．そのため，無彩色のグラフに対する塗り方の 1 つ SEQ 法 [5] を紹介する．まず，一つの頂点に接続されている辺の数をその頂点の次数といい，色は色番号 c_0，c_1... で表す．何も塗られていない無向グラフに対し，頂点の次数が多い順に頂点に色を塗っていく．その時，色は衝突が起こらない中で最も色番号が小さいものを選ぶこととする．図 2 を例にすると，v_1, v_4 が次数 3，v_2, v_3, v_5 が次数 2 であるので，v_1, v_4 から彩色していく．この方法の時間計算量は，頂点数と辺数をそれぞれ n, m としたとき $O(m+n)$ であり，高速に彩色可能である．

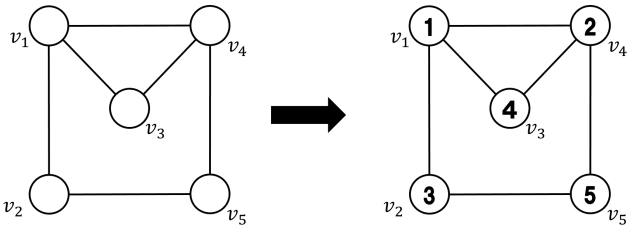

図 2: SEQ 法による彩色の順番

3 従来法

ここでは，文献 [6] の手法を紹介する．SEQ 法を使い，事前に塗られたグラフを用意し，そのグラフの色番号で最大の k で塗られている頂点を，色番号 $k-1$ に塗り直す．それによっていくつかの衝突が発生する．頂点の色を変更していくことを衝突がなくなるまで続け，衝突がなくなれば，色数を 1 減らして，できた衝突をなくすという動作を繰り返す．

衝突をなくす際，変更する頂点を v，色を i とし，変更したときの衝突数から現在の衝突数を引いた値を評価関数 $\delta(v, i)$ とする．全ての衝突頂点と色に対して，$\delta(v, i)$ を調べ，$\delta(v, i)$ が最小になる (v, i) を採用し，頂点 v を i

色に塗り替える．しかし変更を繰り返すと，評価関数最大の頂点と色は同じ解が何度も現れるようになるので，タブーリストを用いることにする．

タブーリストとは一度色を変更した際に，一定の変更回数の間，変更した頂点と色を保持するリストのことで，タブーリストに入っている頂点と色の組み合わせには変更しない．この保持する期間は，衝突している頂点数に応じて増減する．

この方法を使っても，グラフとタブーリストの状態がループする可能性があり，ループした場合に脱出する方法はないのが問題点である．

Algorithm 1 従来法

入力：k 色で塗られたグラフ G
出力：時間内で最適に塗られたグラフ
v：頂点, i：色
t ：経過時間, t_d: 制限時間
while $t < t_d$ **do**
  色番号 k で塗られている頂点を $k-1$ に塗り替える
  while 衝突頂点数 > 0 **do**
    タブーリストにない v, i の組合せの中で$\delta(v, i)$ が最小となるものを探す
    頂点 v を i で塗る
    (v, i) をタブーリストに入れる
    タブーリスト内の保持期間を過ぎたエントリを削除する
  end while
end while

4　提案法

4.1　初期解の生成

提案法 1，2 では，最初に用意される SEQ 法を使ったのち作られる色数 k を 1 減らしたグラフから，以下のようにして $k-1$ 色の彩色を求める．まず，各色に対し頂点の次数の総和を求める．その値が最も小さい色を，ランダムにそれ以外の色に変更する．

4.2　提案法 1

提案法 1 では，ランダム性を加えることで，従来法のように状態のループに陥らないように配慮している．まず，ランダムな色変更による，無意味な色の変更を減らすために予め G の極大クリークを一つ求める．

クリークとは図 3 のようにいずれの頂点 2 つをとってもその間に辺が存在する部分グラフのことである．衝突をなくす際，衝突の頂点と色を考える際に目的のグラフ中に存在するクリークの頂点と色は変更する候補に加えない．理由としては，クリークの頂点は必ず異なる色が塗られるので，色を変えると衝突を起こす可能性が高いためである．また彩色数とクリークの頂点数が同じになったとき，そこで探索を終了出来る利点がある．

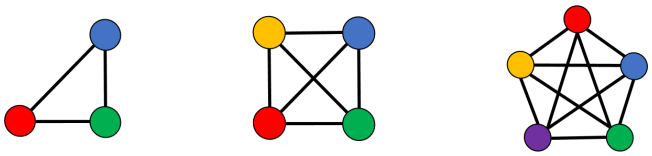

図 3: クリークの例

クリークの探索以降従来法と同様に色数を減らし，できた衝突頂点をなくすことを繰り返す．

まず 4．1 の方法で初期解を作成する．そこから色を変える頂点は衝突している点からランダムに選ばれ，変更する色は，ランダムで変更候補の色を決定し，その色に変更した場合，変更する頂点と衝突する隣接頂点が 1 つ，または 0 になる場合にはその色に変更する．そうでなければランダムでまた変更候補の色を決定する．この行為を最大で一定の回数 n_c まで繰り返し，決まらなかった場合は最後に決定した色に変更する．変更する色を探す際の回数は衝突している頂点数に応じて増減する．

4.3　提案法 2

提案法 2 では，色の変更時間を記録することで，同じ塗り方が起こらないように配慮している．

提案法 1 と同様に色数を減らし，できた衝突頂点をなくすことを繰り返す．まず 4．1 の方法で初期解を作成する．提案法 2 では色を変更し，衝突をなくす時，色の選び方は常に選んだ頂点の色を変更した時に最も衝突が減る色にする．頂点の選び方は，1 番初めは最も衝突の数が多い頂点を選ぶ．その色に変更したのち，衝突ができなければ，その時点で最も衝突の多い頂点，衝突ができた場合は，新たに衝突ができた頂点のうち色の変更時間が最も古い頂点の色を変更する．以降は同様に衝突ができるできない場合に応じて，変更する頂点を決めていく．

新たに衝突する頂点がずっとなくならずループする可能性があるので，一定の回数 n_d の間新たに衝突頂点ができ続けた場合，次に色を変更する頂点はその時点で最も衝突の多いものにする．

Algorithm 2 提案法 1

入力：k 色で塗られたグラフ G
出力：時間内で最適に塗られたグラフ
v：頂点, i：色, n：色変更回数
t ：経過時間, t_d: 制限時間
$n \leftarrow 0$
while $t < t_d$ **do**
  頂点次数最小の色番号 k を調べる
  k で塗られた頂点をランダムに選ばれた r に塗り替える
  while 衝突頂点 > 0 **do**
    ランダムに衝突頂点 v を選ぶ
    while v を i に変更したのちの隣接する衝突頂点の数 > 1 *and* $n < n_c$ **do**
      i をランダムに変更
      $n \leftarrow n + 1$
    end while
    頂点 v を i で塗る
    $n \leftarrow 0$
  end while
end while

Algorithm 3 提案法 2

入力：k 色で塗られたグラフ G
出力：時間内で最適に塗られたグラフ
v：頂点, i：色, n：色変更回数
t ：経過時間, t_d: 制限時間
$n \leftarrow 0$
while $t < t_d$ **do**
  次数の総和が最小の色番号 k を探す
  色番号 k で塗られている頂点をランダムに選ばれた r に塗り替える
  while 衝突頂点 > 0 **do**
    if (v, i) に変更して衝突ができる *and* $n < n_d$ **then**
      できた衝突点のうち最も変更されていない点 v を選ぶ
      $n \leftarrow n + 1$
    else
      衝突が最も多い頂点 v を選ぶ
      $n \leftarrow 0$
    end if
    衝突頂点が最も少なくなる i を選ぶ
    頂点 v を i でする
  end while
end while

5 計算機実験

5.1 実験環境

ベンチマーク問題のグラフを用いて実験と従来法との比較を行った，実験環境は以下のとおりである．

CPU : Intel(R) Core(TM)i7-8700 CPU3.2GH z，メモリ：8.0GB，OS : Windows10，使用言語：Java，計算時間：100 秒．

提案法 1 で，クリークは事前に 5 秒間探索して見つかった一番大きいものを使用する．また提案法 1 はランダム性を用いるため，10 回の平均の結果を使用する．

表 4: 従来法と提案法の計算結果の比較 (単位：色)

グラフ名	従来法	提案法 1	提案法 2
DSJC125.1	5	5	5
DSJC125.5	18	18.2	21
DSJC125.9	44	44.6	52
DSJC250.1	9	7.4	9
DSJC250.5	30	32.2	37
DSJC250.9	73	84.2	88
DSJC500.1	13	12.4	14
DSJC500.5	50	61.8	63
DSJC500.9	131	155.4	167
DSJC1000.1	22	16	26
DSJC1000.5	98	103.8	117
DSJC1000.9	237	281.8	308

表 4 より，提案法 1 では頂点に対し，辺が少ないグラフでは提案法の方がよりよい解になっているが，他の多くのグラフで従来法の方がよい解が出た．提案法 2 では全体的に従来法の方がよい解が出た．

6 まとめと今後の課題

グラフ彩色問題について，最良の変更を随時行う従来法と，2 つの提案法を比較した．

提案法 1 は，一部のグラフではよい結果が得られたが，特に頂点に対し辺が多いグラフにおいて，解が悪化した．これは，そういったグラフにはクリークが多く存在し，事前に探索したクリークが意味をなさないからだと考えられる．提案法 2 は，全体的に悪い結果になった．これは，塗り方がループすることへの対策が十分でなかったと考えられる．

これからの改善点として，提案法 1 では変更する頂点をランダム 2 つから衝突が多い方を選ぶことで，ランダム性を減らすことや事前に探索するクリークを頂点に対

する辺の量に応じて，増やすといったことが考えられる．提案法 2 では，ループを解消するために，一定時間衝突がなくならない時に変更する頂点を衝突が最も大きい頂点を変更するといった方法が考えられる．

参考文献

[1] Garey， M. and Johnson， D. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness， W. H. Freeman & Company.

[2] Gamst A.（1986） Some lower bounds for a class of frequency assignment problems. IEEE Transactions of Vehicular Echnology 35，8–14.

[3] Leighton FT(1974)． A graph coloring algorithm for large scheduling problems. Journal of Research of the National Bureau of Standards 84，489–503.

[4] deWerra D．(1985) An introduction to timetabling. European Journal of Operational Research 19，62-151.

[5] D. de Werra(1990)， Heuristics for graph coloring，Computing (Suppl. 7) 191-208.

[6] Galinier P，Hao J-K．(1999)．Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization.3，97-397.