頂点彩色問題に対する近似解法のための初期解生成アルゴリズム

学生番号　2015044
氏名　内田恭貴
提出年度　平成 30 年度

目次

1. はじめに

本研究では,**グラフ**上の**頂点彩色問題**を取り上げる.グラフとは,**頂点**の集合と,頂点と頂点を結ぶ**辺**の集合から成る.2 つの頂点が辺で結ばれているとき,両者は**隣接**しているという.与えられたグラフに対する頂点彩色問題とは,できるだけ少ない種類の色を用いて,隣接する頂点に同じ色が割り当てられないよう,すべての頂点に色を割り当てることを問う問題である.現実での応用例としては,スケジューリング問題[1]が挙げられる.様々な制約の下,与えられた仕事を効率良く遂行するためのスケジューリングを求める問題において,頂点彩色問題の解法が役立つ[2].

頂点彩色問題は**NP 困難**問題[3]の 1 つで,効率良く（＝**多項式時間**で）**最適解**を求めることは絶望的と考えられる.そのため[4]のような**メタヒューリスティクス**[5]に基づいた**近似解法**が提案されてきた.多くの近似解法では TabuCol[6]や PartialCol[7]のような**局所探索法**[8]がサブルーチンとして用いられる.しかしながら,このような局所探索法を含む近似解法に用いられる**初期解生成アルゴリズム**は,非効率な側面を幾つか併せ持っており改善の余地が十二分に存在した.

そこで本研究では,こういった局所探索法を含む近似解法全般のための,新しい初期解生成アルゴリズム「**リサイクル法**」を提案する.また DIMACS のベンチマークグラフ[9]を用いて,従来法より優れていることを実験的に確認した.加えて,より複雑で性能の良い上位近似解法にもリサイクル法を組み込み,リサイクル法の有効性を実験的に示すことに成功した.

以下,2 章では頂点彩色問題の定義と応用について説明し,3 章では近似解法について解説し,本論文で取り上げる 3 つの近似解法を紹介する.4 章においては既存の初期解生成アルゴリズムとリサイクル法について取り上げる.次の 5 章では計算実験の結果を示し考察を行い,6 章でまとめを行う.

2. 頂点彩色問題

2.1 定義

与えられたグラフに対する頂点彩色問題とは, できるだけ少ない種類の色を用いて, 隣接する頂点に同じ色が割り当てられないよう, すべての頂点に色を割り当てることを問う問題である. 本論文におけるグラフとは頂点と辺で構成されたものを指す. グラフの頂点を彩色した例を図 1, 図 2, 図 3 に示す.

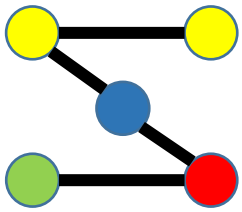

図 1：実行不能解

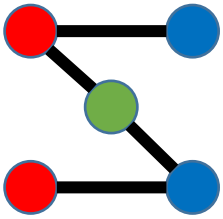

図 2：実行可能解(色数：3)

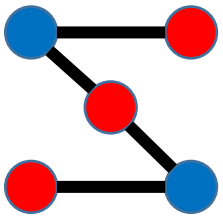

図 3：実行可能解(色数：2)

図 1 では, 隣接する頂点に同じ色が割り当てられないようにしなければならないという制約条件を満たさないため(上 2 つの頂点に「黄」が割り当てられている), **実行不能解**である. 図 2 では, 制約条件に違反していないため, **実行可能解**である. 「赤, 青, 緑」の 3 色で彩色を行うことが出来た. 図 3 も同じく, 制約条件に違反していないため, やはり実行可能解である. 「赤, 青」の 2 色で彩色を行うことが出来た. ここで, 図 2 と図 3 の実行可能解を比較すると, 図 3 の解がより良いと判断することができる. 何故ならば, 頂点彩色問題の目的はできるだけ少ない種類の色を用いて彩色することだからである. 従って, 3 色での塗り分けより 2 色での塗り分けのほうが良い解であると判断できる.

2.2 応用例

頂点彩色問題には様々な応用が存在する. 有名な応用例としてはスケジューリング問題[1]が挙げられる. 様々な制約の下, 与えられた仕事を効率良く遂行するためのスケジューリングを求める問題において, 頂点彩色問題の解法が役立つ[2].

本論文では具体的な例としてタクシー会社の予約客に対するスケジューリング問題を紹介する. 図 4 はそれぞれの予約客がタクシーを利用する時間を表に示したものである. 横軸が時間を表している. 例えば, 予約客 1 はある時刻から傍線部の長さの時間分タクシーを利用するということを示す. 従って, 傍線部が重なり合う予約客は同一のタクシードライバーが担当できないことを表す. この図 4 を一定の条件下でグラフに落とし込んだものを図 5

に示した. グラフに変換する際に定めた条件は, 予約客を頂点とし, タクシーを利用する時間が重なる予約客同士を辺で結ぶというものである. また, 変換したグラフを彩色したものが図 6 である. この場合においては, 彩色した色がタクシードライバーに対応する. 自明だが, 同一色は同一のドライバーであることを指す.

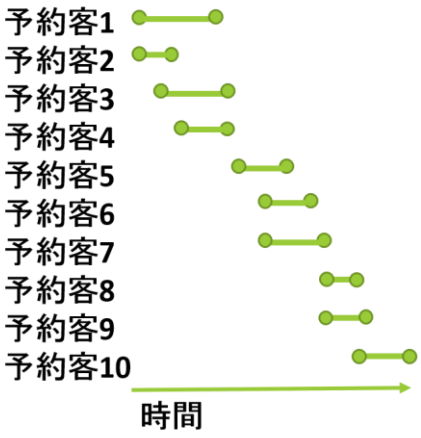

図 4：タクシー予約のスケジュール

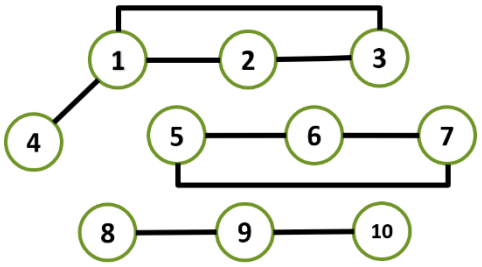

図 5：図 4 のスケジュールから得られたグラフ

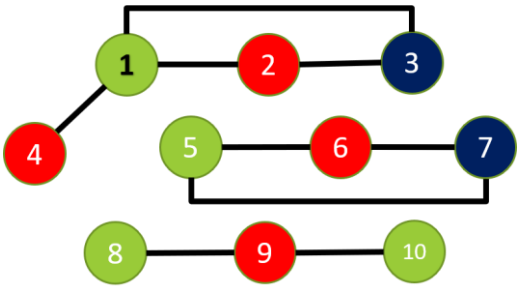

図 6：図 5 に対する実行可能解のグラフ

頂点彩色問題においては彩色に使用する色の数を最小化することが目的であることから, この例ではタクシードライバーの人数を最小化することが目的となる. この最小化によって, より少ないタクシードライバーでより多くの予約客を輸送することができるようになる. このことはタクシー会社の運営の効率化に直結する.

3. 近似解法

頂点彩色問題は古くから知られる NP 困難問題[3]の 1 つである.NP 困難問題は,多項式時間で**最適解**を求めることが絶望的とされている.本研究で取り上げるグラフは頂点数が 250 個と比較的少ないものから,4000 個もの頂点を有するものもある.従って,大規模なグラフを実用的な時間内で解く必要があるため,本研究では近似解法という手法を使う必要がある.

近似解法とは,多項式時間で解くことが絶望的な問題を実用的な時間内で**近似解**を求めるための解法である.しかしながら,必ずしも最適解を導き出すことができるわけではないことに注意する必要がある.近似解法には,模擬焼きなまし法・禁断探索法・大近傍探索法・局所探索法[8]など様々な種類が知られている.

3.1 近似解法の枠組み

頂点彩色問題に対する多くの近似解法は,**k-彩色可能性問題**を繰り返し解くことによって近似解を求める.与えられたグラフに対する k-彩色可能性問題とは,グラフが k 色で彩色可能かどうかを問い,もし可能ならばその証拠(k 色で塗られた実行可能解)をも問う問題である.

頂点彩色問題に対する近似解法の手順をまとめる.

1. 適当な手法(ランダム法,欲張り法など)を用いて実行可能解を求め,その色数を k とする.
2. k←k-1 とする.
3. 終了条件(計算時間や反復回数など)が満たされるまで,k-彩色可能性問題を解く.
   - 終了条件が満たされたら 4 へ.
   - 終了条件が満たされる前に実行可能解を得られたら 2 へ.
4. k+1 色塗られた実行可能解を出力し,終了.

次に本論文で取り上げる近似解法と解空間モデルの関係を表 1 に示した.

表 1：近似解法の戦略と解空間モデルの関係

		解空間モデル	
		実行不能解許容モデル	不完全解許容モデル
近似解法の戦略	一点探索	TabuCol	PartialCol
	多点探索	HEA	本論文では取り上げない

近似解法には,**一点探索**と**多点探索**の 2 種類の戦略が存在する.一点探索は局所探索法とも言われており,1 つの解を移動させながら探索を行うものである.本論文では一点探索のアルゴリズムの例として TabuCol[6]と PartialCol[7]を取り上げる.一方で多点探索とは複数の解を使用して探索を行うものである.本論文では HEA[10]を取り上げる.これらの詳細は 3.2 節以降で説明する.

また,解空間モデルにも**実行不能解許容モデル**と**不完全解許容モデル**の 2 種類が存在する.実行不能解許容モデルとは, 隣接する頂点に同じ色が割り当てられることを許容するもので,そのような頂点対の個数が解のペナルティ値(評価値)である.他方で不完全解許容モデルとは, k 色のいずれも割り当てられない頂点を許容するもので,そのような頂点の個数が解のペナルティ値である.いずれのモデルにおいても,ペナルティ値が 0 の解が見つかれば,k 色彩色可能と判定することができる.

いずれのモデルにおいてもペナルティ値という概念を導入したが,これは実行可能性への「近さ」を示すと考えられるので,探索開始時点のペナルティ値が 0 に近ければ近いほど理想的である.

3.2 近似解法の例

本節では近似解法の例として表 1 に挙げた TabuCol,PartialCol,HEA の 3 つを紹介する.

3.2.1 TabuCol

TabuCol は実行不能解許容モデルの下でタブーサーチ[5]に基づいた解空間の探索を行う.実行不能解許容モデルでは隣接する頂点に同じ色が割り当てられることを許容するものであるから,図 7 のような解を許容して探索を行う.図 7 の場合,隣に同じ色が割り当てられている頂点対(青色の頂点対)が 2 つ存在するため,ペナルティ値は 2 と判定することが出来る.

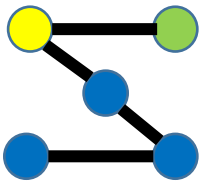

図 7：実行不能解許容モデルの許容解

3.2.2 PartialCol

PartialCol は不完全解許容モデルの下でタブーサーチに基づいた解空間の探索を行う.不完全解許容モデルは k 色のいずれも割り当てられない頂点を許容するものであるから,図 8 のような解を許容して探索を行う.図 8 の場合,いずれの色も割り当てられていない頂点(点線で表示している頂点)が 1 つ存在するため,ペナルティ値は 1 と判定することが出来る.

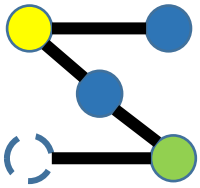

図 8：不完全解許容モデルの許容解

3.2.3 HEA(Hybrid Evolutionary Algorithm)

HEA(Hybrid Evolutionary Algorithm)は,TabuCol をベースとした多点探索型の近似解法であり,実行不能解許容モデルの下,隣接する頂点に同じ色が割り当てられることを許容して探索を行う点は TabuCol と同様である.加えて HEA では,TabuCol による解空間探索によって生じる複数の解候補を保持,活用して探索を行うものである.この複数の解候補によって構成される解候補群を母集団と呼ぶこととする.HEA は,遺伝アルゴリズム[5]のフレームワークに基づき,解の**交叉**,**選択**,**突然変異**によって母集団を更新する.

解候補の活用例について,具体例を交えながら解の交叉について解説する[2].無論,TabuCol の探索過程において複数の解候補が母集団として保持されている状態を想定する.以下の 2 つの群が母集団にあるとする.

・解候補 1[{v1,v2,v3},{v4,v5,v6,v7},{v8,v9,v10}]

・解候補 2[{v3,v4,v5,v7},{v1,v6,v9},{v2,v8,v10}]

解候補が 2 つ存在しており,{}でくくった中に含まれている頂点は同じ色で彩色されていることを示している.この 2 つの候補を利用し新たな解を生成することを解の交叉という.交叉の手順は以下の通りである.

1. いずれかの候補において使用されている回数が最も多い色の頂点群を選出.
2. 選出された頂点群に含まれている頂点を双方の候補から取り除く.
3. いずれかの候補において使用されている回数が 2 番目に多い色の頂点群を選出.
4. 選出された頂点群に含まれている頂点を双方の候補から取り除く.

この流れをすべての頂点が選出されるまで繰り返す.実際に上記の解候補をこの手順に従い活用し新たな解を生成してみる.

1 回目の選出 {v4, v5, v6, v7}※候補 1 より選出し双方から対応する頂点を除去

・解候補 1[{v1, v2, v3}, {v8, v9, v10}]

・解候補 2[{v3}, {v1, v9}, {v2, v8, v10}]

2 回目の選出 {v2, v8, v10}※候補 2 より選出し双方から対応する頂点を除去

・解候補 1[{v1, v3}, {v9}]

・解候補 2[{v3}, {v1, v9}]

3 回目の選出 {v1, v3}※候補 1 より選出し双方から対応する頂点を除去

・解候補 1[{v9}]

・解候補 2[{v9}]

4 回目の選出 {v9} →終了

ここで k=3 であることから 4 回目の選出頂点を新たな色に彩色すると k=4 になってしまう. このことから, 4 回目の選出頂点を 1, 2, 3 回目の選出頂点群のいずれかに割り当てる必要がある. 当然, TabuCol が探索のベースであることから必ずしも隣り合う頂点に異なる色が割り当てられるようにする必要性はない. 今回の例では 2 回目の選出頂点群に v9 を加えた. 新たな解は[{v4, v5, v6, v7}, {v2, v8, v10, v9}, {v1, v3}]となった.

HEA ではこの他, 選択や突然変異などの操作によって母集団を更新し, ペナルティ値 0 の解(すなわち実行可能解)が得られた時点でそれを出力し, 探索を終了する.

4. 初期解生成アルゴリズム

3 章で述べた TabuCol[6]や PartialCol[7]などの近似解法には,適当な**初期解**を与えて探索をスタートさせる必要がある.このため,従来 Greedy[2]や DSATUR[11]などの欲張り法に基づいたアルゴリズムが用いられてきた.ところが,

・直前の計算（すなわち k+1 色彩色可能性の判定）の結果が何ら反映されないこと,

・k が小さくなるほど,初期解の質が相対的に悪くなること,

などから本研究では,直前の計算で得られた k+1 色の割当を用いて,k 色彩色可能性判定計算のための初期解を生成することを考える.従って,前述した実行不能解許容モデルと不完全解許容モデルに適応したリサイクル法を本章で提案し説明する.

提案するリサイクル法の最大の目的は,既存の初期解生成アルゴリズムによって生成される初期解のペナルティ値より,さらに小さいペナルティ値の初期解を生成することである. 3.1 節で触れた通り初期解のペナルティ値が小さいほど,少ない計算時間で最適解(すなわちペナルティ値 0 の解)へたどりつくことが予想されるからである.

加えてその単純さから,リサイクル法は実装が極めて容易である.また計算時間も短い.グラフの頂点の個数を n とすると,リサイクル法を用いた初期解生成の時間量は O(n)である.これに対し,Greedy や DSATUR の時間量は $O(n^2)$ である[2].

以下,4.1 節では従来法の Greedy,4.2 節では従来法の DSATUR を紹介し,4.3 節においてリサイクル法を説明する.

4.1 Greedy

Greedy アルゴリズムは,TabuCol や PartialCol に用いられている初期解生成アルゴリズムの 1 つである.Greedy 法では予め頂点に色を塗る順番を決める.その順番に従って頂点を訪れ,その時点で使用している色のいずれかで彩色できる場合は,隣り合う頂点に同じ色が塗られないように彩色を行う.訪れた頂点にその時点で使用している色のいずれかで彩色できない場合は,新たに色を生成して彩色する.

図 9 は Greedy 法の初期解生成手順を図に示したものである.上段左は頂点に色を塗る順番を決める段階であり,頂点の数字はこの段階で決定された色を塗る順番を表している.上段中央は順番 1 に設定された頂点に青色を彩色した段階である.上段右も同様にして順番 2 に設定された頂点に黄色の彩色を行った段階である.訪れた時点では青色しか使用されていないので,青色を彩色するしかないが隣に青色の頂点が存在するので,青色彩色は不可能である.よって,新たに黄色を生成し塗り分けを行った.このように予め決められた順番に

従って,使用される色の数が少なくなるように塗り分けを行うことを繰り返す.塗り分けの結果生成された解が下段右のグラフとなる.

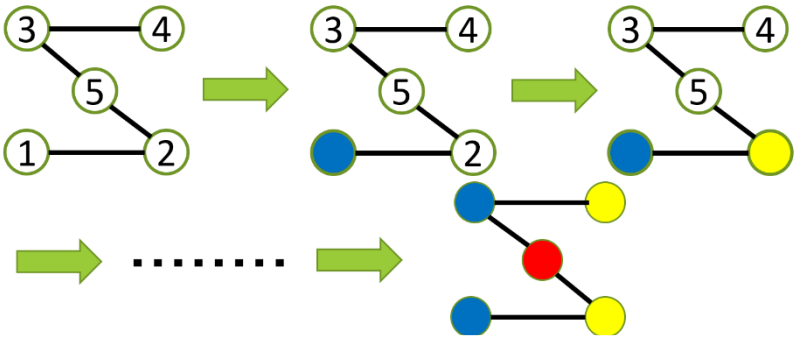

図 9：Greedy の初期解生成手順

4.2 DSATUR

DSATUR アルゴリズムは,Greedy 法と同様に TabuCol や PartialCol に用いられている初期解生成アルゴリズムである.DSATUR 法では彩色する頂点の順番を動的に決定する.すなわち,既に彩色された頂点の影響を受けて次の選出される頂点が決まることになる.頂点の選出方法を除けば彩色の方法は Greedy 法と同様である.頂点選出の手順は以下の通りである.

1. 空白(色の塗られていない)の頂点に隣接する頂点の色の数をカウント.
2. 隣接する頂点の色の数が小さいものから彩色.

頂点選出の手順を図 10 に示した.頂点の中の数字は選出された順番を指している.図 10 の左のグラフにおいて,頂点 1 では隣接する頂点の数は 1 つで,青色の頂点のみに隣接している.他方で,頂点 2 では隣接する頂点は 2 つで,青色と赤色の 2 色に隣接している.ここで,上記の頂点選出手順より隣接している色の数が少ない 1 番から選出されて彩色することになる.図 10 の右のグラフは 1 番の彩色後を示している.

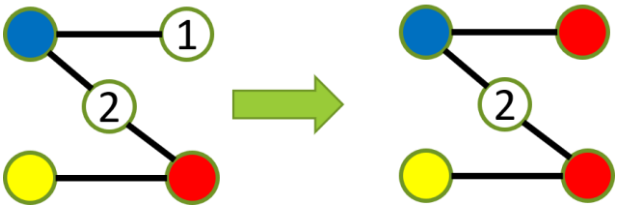

図 10：DSATUR の初期解生成手順

4.3 リサイクル法

4.3.1 実行不能解許容モデル

実行不能解許容モデルにおけるリサイクル法のアルゴリズムは以下の通りである. 3.1 節で示した近似解法の枠組みの中で用いることを想定しているため, この手順においては, グラフに対する k+1 色の実行可能解が存在しているものとする.

1. k+1 色の割当のうち, 使用されている数が最も少ない色を求める.
2. 使用数が最も少ない色を空白（色の塗られていない状態）へ変更する.
3. 空白へ変更した頂点に k 色のいずれかを割り当てる.

この手順で初期解を生成したのちに, 実行不能解許容モデルの下で, タブーサーチ[5]に基づいた解空間の探索を行う.

提案した初期解生成アルゴリズムを図 11 に示す. 左は k+1 色で彩色されたグラフである. 左のグラフで使用されている回数が最も少ない色は赤色である. 青色と緑色は 2 回ずつ使用されていることに対して赤色は 1 回しか使用されていない. 従って, この場合は赤色が空白に変更されることになる. 中央のグラフは赤色を空白に変更したグラフである. 頂点を空白に変更した後に, その頂点を k 色のうちのいずれかの色をランダムに割り当てる. 右のグラフが k 色のうちのいずれかをランダムで彩色したものである. この例では青色が彩色された. この状態から実行不能解許容モデルの下, 解空間を探索して k 色の彩色可能性を判定する.

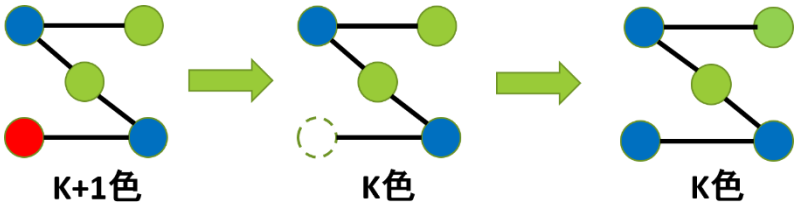

図 11：実行不能解許容モデルに対応したリサイクル法による初期解生成手順

同じく, 実行不能解許容モデルの下で解空間を探索する多点探索型の近似解法(HEA[10]など)も同様の手順でリサイクル法を適用し初期母集団を生成する. 流れは以下の通りである. こちらも同様に前提として k+1 の実行可能解が 1 つ以上生成されているものとする.

1. 母集団に属する個々の解についてリサイクル法を適用する.
2. 母集団解と生成した解を入れ替える.

4.3.2 不完全解許容モデル

不完全解許容モデルにおけるリサイクル法のアルゴリズムは以下の通りである. この手順においても, k+1 色の彩色がなされた実行可能解が存在しているものとする.

1. k+1 色の割当のうち, 使用されている数が最も少ない色を求める.
2. 使用数が最も少ない色を空白（色の塗られていない状態）へ変更する.

この手順で初期解を生成したのちに, 不完全解許容モデルの下で, タブーサーチに基づいた解空間の探索を行う.

提案した初期解生成アルゴリズムの概要を図 12 に示した. グラフの形や色は実行不能解許容モデル(4.3.1 節)の解説に使用したものと相違点はない. 従って, 左のグラフは同様に赤色が空白に変更されることになる. 右のグラフは赤色を空白に変更したグラフである. 1 色を空白に変更することで全体では空白を除くと k 色が使用されているグラフとなり, この状態から不完全解許容モデルの下, 解空間を探索して k 色の彩色可能性を判定する.

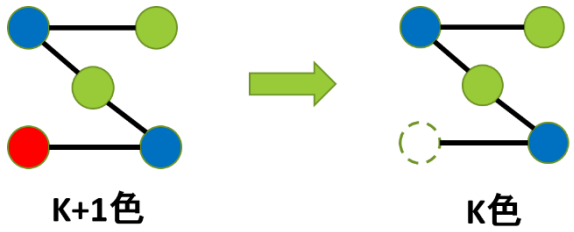

図 12：不完全解許容モデルに対応したリサイクル法による初期解生成手順

5 計算実験

リサイクル法が既存の初期解生成アルゴリズムである Greedy[2],DSATUR[11]と比べ有効であることを示すために 3 つの近似解法(TabuCol[6],PartialCol[7],HEA[10])の初期解生成法として用いる実験を行った.既存のアルゴリズムよりリサイクル法で生成した初期解のペナルティ値が小さくなること,出力解に使用される色数が少なくなることを実験的に証明することが目的である.

5.1 設定

4 章で提案したリサイクル法をＣ言語で実装し,既存の初期解生成アルゴリズムである Greedy と DSATUR と比較した.実験には DIMACS[9]の問題例 21 個を使用し,乱数の種を 10 種類ずつ設け,10 種の最適解探索の実行結果を平均したものを比べた.実験に使用したコンピュータの CPU は Intel®Celeron®2975U でクロック周波数は 1.40GHz,メインメモリは 4GB である.

また,解の探索終了条件は問題例データのチェック回数が一定回数(TabuCol,PartialCol は 10 億回,HEA は 100 億回)を超えた時点としている.ここでのチェック操作の定義は,

・2 つの頂点が隣接しているかどうかを確認すること,

・ある頂点に隣接するすべて頂点を確認すること,

・ある頂点に隣接する頂点が幾つあるかを確認すること,

・ある頂点に何種類の色が隣接しているかを確認すること,

としている.

5.2 色数の比較

すべての表で,同じ行(すなわち問題例)における最良の値を太字で表した.

Greedy,DSATUR,リサイクル法を初期解生成法として用いたときの近似解法が求めた解の色の数を表 2 から表 4 に示す.それぞれ表 2 が TabuCol,表 3 が PartialCol,表 4 が HEA に対応している.

表 2 : TabuCol に対して 3 つの異なる初期解生成アルゴリズムを適用したときに得られた解の色数

Files	頂点数	Greedy	DSATUR	リサイクル法
C2000.5	2000	194.7	193.1	**176.5**
C4000.5	4000	381.9	372.9	**334.9**
dsjc250.5	250	28.9	**28.8**	29.0
dsjc500.1	500	**12.9**	13.0	13.0
dsjc500.5	500	51.8	51.8	**51.2**
dsjc500.9	500	133.3	133.0	**130.1**
dsjc1000.1	1000	**21.0**	**21.0**	**21.0**
dsjc1000.5	1000	97.0	97.0	**95.3**
dsjc1000.9	1000	273.3	272.3	**240.6**
dsjr500.1c	500	90.8	88.2	**86.3**
dsjr500.5	500	130.6	129.9	**126.8**
flat300_28_0	300	**32.0**	**32.0**	**32.0**
flat1000_50_0	1000	95.0	95.0	**91.6**
flat1000_60_0	1000	95.7	95.3	**93.8**
flat1000_76_0	1000	96.0	96.0	**94.3**
latin_square	900	113.9	113.4	**112.0**
le450_25c	450	**26.3**	**26.3**	26.4
le450_25d	450	26.4	**26.1**	26.4
r250.5	250	69.1	67.3	**66.5**
r1000.1c	1000	103.0	101.4	**100.7**
r1000.5	1000	259.2	250.0	**244.5**

表 3：PartialCol に対して 3 つの異なる初期解生成アルゴリズムを適用したときに得られた解の色数

File	頂点数	Greedy	DSATUR	リサイクル法
C2000.5	2000	186.4	185.4	**173.1**
C4000.5	4000	376.9	365.4	**333.7**
dsjc250.5	250	**29.0**	**29.0**	**29.0**
dsjc500.1	500	**13.0**	**13.0**	**13.0**
dsjc500.5	500	51.9	52.0	**51.5**
dsjc500.9	500	132.9	133.1	**130.7**
dsjc1000.1	1000	**21.0**	**21.0**	**21.0**
dsjc1000.5	1000	96.6	96.7	**93.7**
dsjc1000.9	1000	256.2	255.9	**242.3**
dsjr500.1c	500	92.6	88.6	**87.0**
dsjr500.5	500	130.6	129.9	**126.9**
flat300_28_0	300	**32.1**	**32.1**	32.2
flat1000_50_0	1000	94.8	94.6	**91.6**
flat1000_60_0	1000	95.0	95.1	**92.1**
flat1000_76_0	1000	95.6	95.4	**92.4**
latin_square	900	117.9	117.6	**113.1**
le450_25c	450	**27.0**	27.2	**27.0**
le450_25d	450	**27.0**	**27.0**	**27.0**
r250.5	250	68.2	67.3	**66.4**
r1000.1c	1000	110.5	102.8	**101.1**
r1000.5	1000	261.1	250.0	**244.9**

表 4：HEA に対して 3 つの異なる初期解生成アルゴリズムを適用したときに得られた解の色数

Files	頂点数	Greedy	DSATUR	リサイクル法
C2000.5	2000	175.0	174.7	**172.3**
C4000.5	4000	339.6	339.0	**318.0**
dsjc250.5	250	**28.0**	28.1	28.1
dsjc500.1	500	**12.2**	12.3	12.6
dsjc500.5	500	50.0	50.1	**49.6**
dsjc500.9	500	130.1	130.3	**128.5**
dsjc1000.1	1000	**21.0**	**21.0**	**21.0**
dsjc1000.5	1000	93.2	93.4	**92.2**
dsjc1000.9	1000	248.5	248.3	**239.6**
dsjr500.1c	500	87.8	**87.4**	87.6
dsjr500.5	500	124.2	124.5	**123.9**
flat300_28_0	300	**31.0**	**31.0**	**31.0**
flat1000_50_0	1000	91.1	91.1	**88.0**
flat1000_60_0	1000	92.0	92.0	**90.5**
flat1000_76_0	1000	92.0	92.0	**90.9**
latin_square	900	110.0	110.1	**108.6**
le450_25c	450	27.0	27.0	**26.8**
le450_25d	450	27.0	27.0	**26.9**
r250.5	250	**65.8**	65.9	66.0
r1000.1c	1000	102.7	102.1	**100.5**
r1000.5	1000	**241.7**	242.1	242.3

表 2,表 3 を見ると TabuCol と PartialCol の双方には同じ傾向があることが判明した.頂点数の少ない小規模なグラフ(頂点数 250～450)においては,リサイクル法は既存のアルゴリズムと同等の性能を出している一方で,頂点数の多い大規模なグラフ(頂点数 2000 以上)においては,リサイクル法は既存のアルゴリズムを大きく上回る性能を出しているというものである.また,中規模なグラフ(頂点数 500～1000)においても大規模グラフほどの差は出ていないものの,ほとんどが既存のアルゴリズムを上回る性能を出している.このことか

ら，リサイクル法は一点探索の近似解法において既存の初期解生成アルゴリズム(Greedy，DSATUR)より有効であることが示された．

また，表 4 から HEA も同様に，小規模なグラフにおいては既存のアルゴリズムと同等の性能を叩き出し，大規模なグラフにおいては既存のアルゴリズムを大きく上回る結果となった．このことは多点探索の近似解法においてもリサイクル法が有効であることを示している．

さらに，大きな性能差が出た大規模グラフを取り上げ，リサイクル法によって色数の改善が加速されていることを視覚的に示す．C4000.5 の問題例を取り上げ，色数の改善を折れ線グラフで表したものが図 13 である．縦軸は色数，横軸はチェック回数を示している．同じ色数でも，リサイクル法は既存のアルゴリズムに比べて，少ないチェック回数で解を求めることが出来ている．

以上より，色数の改善という観点では従来法よりリサイクル法が優れていると考えられる．

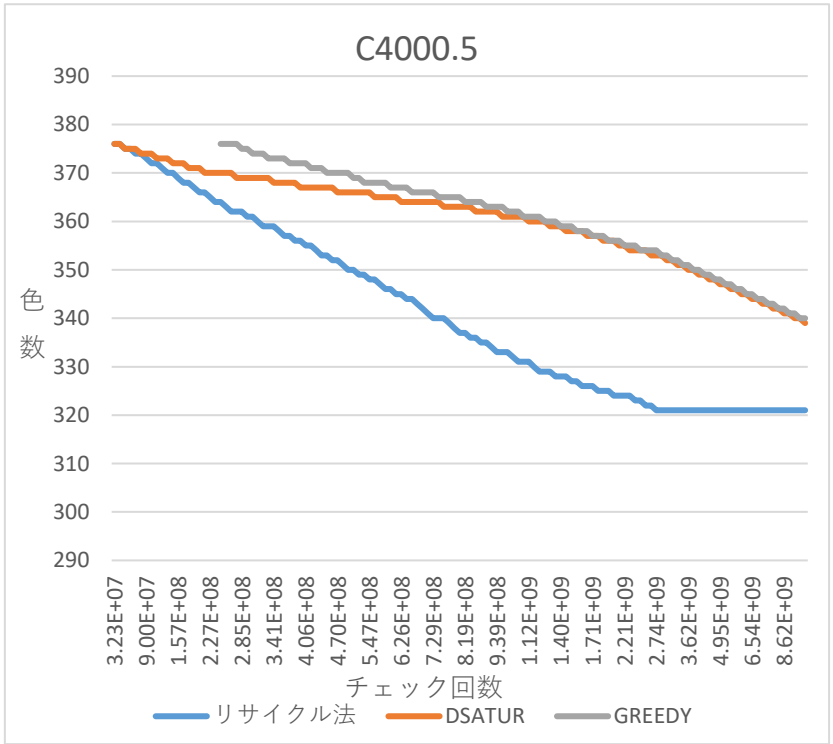

図 13：C4000.5 における色数とチェック回数の比較

5.3 ペナルティ値の比較

続いて,従来法とリサイクル法によって生成される初期解のペナルティ値の比較を行う.

比較に使用した問題例は図 13 でも取り上げた C4000.5 である.図 14 が TabuCol におけるペナルティ値の比較,図 15 が PartialCol におけるペナルティ値の比較を示している.いずれの場合においても従来法では色数が改善されていくに従い初期解のペナルティ値が大きくなっていることが分かる.これは色数の改善に伴い初期解の質が悪くなっていることを示している.

一方でリサイクル法は色数が改善に向かっても,初期解のペナルティ値は大きく変化せずにほぼ一定の値を維持していることが分かる.このことから,リサイクル法は色数の改善に関係なく初期解の質はほぼ一定に保たれることが示された.これは色数の改善に伴いリサイクル法の優位性が大きくなることにつながる.

以上より,初期解のペナルティ値という観点においてもリサイクル法は従来法より優れていると考えられる.

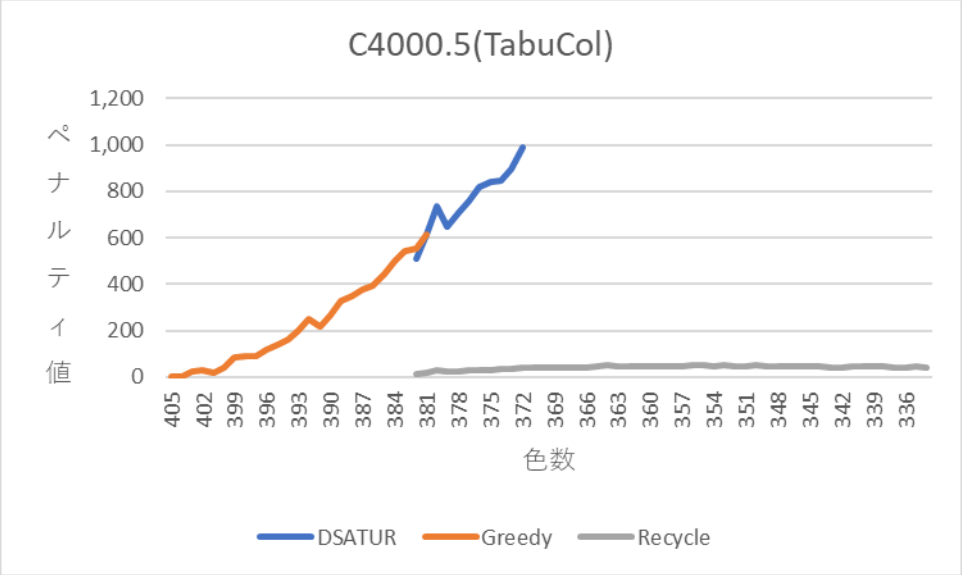

図 14：TabuCol における色数とペナルティ値の比較

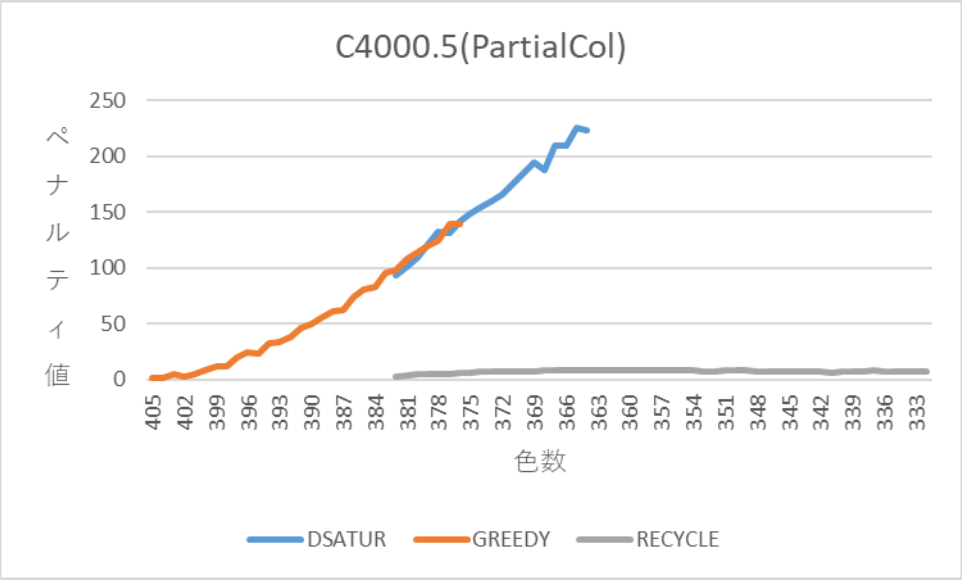

図 15：PartialCol における色数とペナルティ値の比較

6 まとめ

本論文では,頂点彩色問題に対する近似解法のための新たな初期解生成アルゴリズムとしてリサイクル法を提案した. TabuCol[6]や PartialCol[7]で提案したリサイクル手法の有効性を示すことが出来たことに加えて,より性能の良い上位近似解法でもリサイクル法の有効性が示された.このことから,リサイクル法が近似解法のための新たな初期解生成アルゴリズムの候補に成り得る可能性が十分にあるのではないだろうか.

今後の課題としては,HEA[10]のようにより複雑で性能の良い近似解法にリサイクル法を組み込み,リサイクル法の有効性をさらに細かく調べていくことである.同じアルゴリズムであっても,組み込む近似解法を変えることによって,より少ないチェック回数やペナルティ値で色数の改善ができる可能性がある.従って,本研究では取り上げなかった近似解法にリサイクル法を組み込んで実験を行うことは十分有益であると言うことが出来る.

また,今回使用した DIMACS の問題例の執筆時点(2018/12/15)で判明している最良解を紹介する.提案したリサイクル法がどこまで現状の最良解まで迫ることができたかということを確認していただきたい.

Files	best K	リサイクル法	Files	best K	リサイクル法
C2000.5	146	171	flat300_28_0	28	31
C4000.5	260	316	flat1000_50_0	50	87
dsjc250.5	28	28	flat1000_60_0	60	90
dsjc500.1	12	12	flat1000_76_0	76	90
dsjc500.5	48	49	latin_square	97	108
dsjc500.9	126	127	le450_25c	25	26
dsjc1000.1	20	21	le450_25d	25	26
dsjc1000.5	83	91	r1000.1c	98	99
dsjc1000.9	222	238	r1000.5	234	240
dsjr500.1c	84	86	r250.5	65	66
dsjr500.5	122	123			

謝辞

本論文は多くの方々からのご協力を受け執筆することが出来ました.ご指導いただいた先生,一次審査で貴重なご意見をくださった先生方に心から感謝致します.

参考文献

[1] Leighton, F. (1979). A graph coloring algorithm for large scheduling problems, Journal of Research of the National Bureau of Standards 84: 489-503.

[2] Lewis, R.M.R: A Guide to Graph Colouring. Springer.

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

[4] Wu, Q. and Hao, J.K. (2012): Coloring large graphs based on independent set extraction, Computers & Operations Research, 39-2: 283-290.

[5] 久保幹雄, J.P.ペドロソ(2009). メタヒューリスティクスの数理, 共立出版.

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

[7] Blöchliger, I. and Zufferey, N. (2008). A graph coloring heuristic using partial solutions and a reactive tabu scheme, Computers & Operations Research 35: 960-975.

[8] Galinier, P. and Hertz, A. (2006). A survey of local search methods for graph coloring, Computers & Operations Research 33(9): 25472562.

[9] DIMACS Graphs:Benchmark Instances and Best Upper Bounds: http://www.info.univ-angers.fr/pub/porumbel/graphs/ (2018年12月15日確認)

[10] Galinier, P. and Hao, J. (1999). Hybrid evolutionary algorithms for graph coloring, Journal of Combinatorial Optimization 3: 379-397.

[11] Brélaz, D. (1979). New methods to color the vertices of a graph, Communications of the ACM 22(4): 251-256.