彩色多項式を求めるヒューリスティック

A heuristic method of computing chromatic polynomials

谷　研究室　　今井　隆宏
Takahiro Imai

概要

P.Berthome,S.Lebresne,and K.Nguyen[1] によって設計された彩色多項式を求めるアルゴリズムの実装を試みる. 但し、clique-tree,triangulation のアルゴリズムは Pinar Heggernes and Yngve Villanger[2] のアルゴリズムを用いている.

1　はじめに

彩色とは各頂点に色を塗るとき、隣接するどの 2 頂点も異なる色で塗ることである. 彩色多項式とは無向グラフでループの無いグラフと、色の数が与えられたとき、そのグラフの異なる彩色の数であり、彩色多項式を求める問題は数え上げ問題の一種である.

本研究では P.Berthome,S.Lebresne,and K.Nguyen によって設計された彩色多項式計算アルゴリズムの実装を試み、また、昨年実装された岩川　英司氏のプログラムより高速なプログラムの開発を目指す.

2 節では定義について、3 節ではアルゴリズムついて、4 節では前節で使用している choice function について、5 節では比較実験について解説する.

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
P(G,λ):最大 λ 色使った時の、グラフ G の彩色多項式
G+e : グラフ G に辺 e を加える
G/e : グラフ G を辺 e で縮約したグラフ

2.2　chordal graph

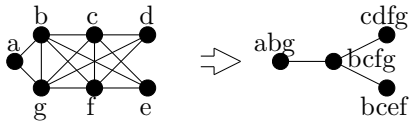
長さが 3 より大きい induce されるサイクルがないグラフ

2.3　triangulation

G=(V,E) の triangulation とは G'=(V,E ∪F) が chordal グラフになるような辺集合 F

2.4　clique-tree

chodal グラフ G の clique tree は次のような $tree\ T = (\mathcal{V}, \mathcal{E})$ である
— $\mathcal{V} = C_1, C_2, \ldots, C_j$ は G の極大クリークの集合
— G の全ての頂点 v について、v を含む極大クリークは T の subtree を induce する
例:



左図の clique-tree

2.5　augumented clique tree

graph $G = (V, E)$
F: G の minimal triangulation $T = (\mathcal{V}, \mathcal{E})$
$tree\ T = (\mathcal{V}, \mathcal{E})$: $G' = V, E \cup F$ の clique tree
このとき、ϕ は次のように定義された F の subset による $\mathcal{E}$ の labeling であり、F のための G の augumented clique tree を $T = (\mathcal{V}, \mathcal{E}, \phi)$ とする.

$$\phi(V_i, V_j) = F \cap E(G'[V_i, V_j])$$

2.6　thickness

2.6.1　triangulation F の thickness

graph $G = (V, E)$
F: G の minimal triangulation
$Th(G, F)$ をグラフ $G, triangulation F$ の $thickness$ とする.

$$Th(G, F) = \max_{(V_i, V_j) \in \mathcal{E}} |\phi(V_i, V_j)|$$

2.6.2　graph G の thickness

G の thickness を $Th(G)$ と定義する.

$$Th(G) = \min_{F\ a\ triangulation\ of\ G} Th(G, F)$$

3 アルゴリズム

3.1 代表的な彩色多項式

Empty graph $P(\bar{K_n}, \lambda) = \lambda^n$

Complete graph $P(K_n, \lambda) = \lambda^{(n)} = \prod_{i=0}^{n-1}(\lambda - i)$

Trees $P(T_n, \lambda) = \lambda(\lambda - 1)^{n-1}$

Chordal graph $P(G, \lambda) = \dfrac{\prod_{i=1}^{k} \lambda^{(|V_i|)}}{\prod_{(V_i, V_j) \in \mathcal{E}} \lambda^{(|V_i \cap V_j)|}}$

3.2 基本アルゴリズム

与えられたグラフが完全グラフになるまで、辺の追加、辺の縮約を繰り返し彩色多項式を求める.

入力　グラフ G
出力　グラフ G の彩色多項式

step 1　グラフ G が完全グラフ k_n ならば $\lambda^{(n)}$ を返す

step 2　完全グラフではないとき
　step 2.1　e $\notin E, G_1 = G + e$ *and* $G_2 = G/e$
　step 2.2　グラフ G_1, G_2を *step*1 へ

Primitive-Chromatic(G)
if G is a coplate graph K_n　then　Return　$\lambda^{(n)}$
Let e $\notin E, G_1 = G + e$ *and* G_2/e
　$P_1 \leftarrow Primitive - Chromatic(G_1)$
　$P_2 \leftarrow Primitive - Chromatic(G_2)$
　$Return P_1 + P_2$

3.3 clique-tree を用いたアルゴリズム

与えられたグラフの qlique tree を利用し、グラフが chodal グラフになるまで、グラフの分割、辺の追加、辺の縮約を繰り返し彩色多項式を求める.

入力　グラフ G,G の clique tree T
出力　グラフ G の彩色多項式

step 1　グラフ G が chordal グラフならば Return

step 2　clique tree をうまく分割できる辺が存在しないならば step3 へ

　step　2.1 グラフ G を分割して step1 へ

step 3　$Lete = ChoiceFuncion(G, T)$

　step　3.1 $G_1 = G + e, G_2 = G/e$

　step　3.2 ふたつのグラフを step1 へ

```
P(G,T)
1  begin
2     if G is trianglated then Return
3     if ∃e ∈ T such that φ(e) = ∅
4        Let G1,T1,G1,T2 and Kr
5        P1 ← P (G1,T1)
6        P2 ← P (G2,T2)
7       return P1 × P2/P(Kr)
8     Let e =  ChoiceFunction (G,T)
9        G1 = G + e , and the resulting T1 = T
10        G2 = G/e , and the resulting T2
11        return P(G1,T1)+P(G2,T2)
12 end
```

4 ChoiceFunction

4.1 岩川さんの工夫

3.3 に書かれている step3 は基本アルゴリズムの考え方であり、出来る限り step2 を行うことで高速になっていく.clique-tree を用いたアルゴリズムの 8 行目に書いてある ChoiceFunction の選び方によって、step2 へいける可能性が増える事で高速になる. そこで、岩田さんは label に多く使われているグラフの辺を選ばせる事にしていた.

4.2 岩川さんのアルゴリズムとの違い

一方、clique-tree 分割後の subtree の頂点数がが小さくなるほど、計算が高速になることが知られていたので、私は分割後の 2 つの subtree の頂点数がなるべく等しくなるよう clique-tree の辺を選び、その辺の labl の中から clique-tree の辺の label に多く使われているグラフの辺を選ばせる事にして実装した.

5 比較実験

5.1 実験方法

・C++クラスライブラリ LEDA5.1 を利用し，上記の新しいアルゴリズムを実装し、基本アルゴリズム、clique tree 作成後ランダムに辺を選び計算するプログラム、岩川さんのプログラムは去年のモノを利用.
・頂点 20 に対し、同じ辺密度のグラフを 10 回試行し、実行時間から triangulation 作成及び clique tree 作成に使用した時間を省いた平均時間をとり、５つの辺密度に対し試行した.
・実行環境
OS:Linux version 2.6.17-1.2442 FC4
CPU:Intel Pentium4 2.66GHz
RAM:1024MB
コンパイラ:gcc 4.0.2

5.2 実験結果

頂点 20					
辺密度	10.00	30.00	50.00	70.00	90.00
time(s):基本	7200 以上	7200 以上	7200 以上	625.74	0.43
time(s):ch(G,T) ChoiceFunction:ramdam	42.15	213.43	363.34	44.24	0.18
time(s):ch(G,T) ChoiceFunction:2007	0.04	26.09	46.36	9.20	0.11
time(s):ch(G,T) ChoiceFunction:2008	0.02	119.00	180.32	34.43	0.16
thickness	3.8	4.4	4.1	3.7	2.1

6 今後の課題

・プログラム全体の効率を考え、triangulation や clique-tree を作成するアルゴリズムに工夫を加える.

参考文献

[1] P.Berthome,S.Lebresne,and K.Nguyen. Computation of chromatic polynomials using triangulations and clique trees. : wg 2005 - 31st International Workshop on Graph-Theoretic Concepts in Computer Science, Metz, France, June 23-25, 2005 : http://www.lri.fr/Rapports-internes/2005/RR1403.pdf

[2] Pinar Heggernes and Yngva Villanger. Efficient Implementation of a Minimal Triangultion Algorithm. Proceedings ESA 2002 - 10th European Symposium on Algorithms, Rome, Italy, September 2002. Springer Verlag, Lecture Notes in Computer Science 2461, pages 550-561. : http://www.ii.uib.no/~pinar/LB-treedec.pdf

[3] Yngve Villanger. Efficient Minimal Triangulation of Graphs by Using Tree Decompositions, Master's Thesis. Department of Informatics, University of Bergen,Norway, 2002. : http://www.ii.uib.no/~yngvev/publications/Master_Thesis.pdf