情報処理学会論文誌　Vol. 0　1–8 (??? 1959)

# 定数次数グラフの同型性判定アルゴリズムの実装

園田尚人[†1]　山内由紀子[†1]
来嶋秀治[†1]　山下雅史[†1]

二つのグラフが与えられた時，それらが同型であるか否かを判定する問題をグラフ同型性判定問題という．この問題が一般に多項式時間で計算可能であるか否かは，著名な未解決問題である．3 正則グラフの同型性判定問題に対する多項式時間アルゴリズムは Luks によって与えられている．Luks のアルゴリズムでは群論を利用して計算を行うことを述べている．しかし具体的な計算機における実装については述べていない．そこで本研究では，Luks のアルゴリズムを計算機上に実装し，その動作について確認と考察を行う．

# Implementation of Isomorphism Test Algorithm for Regular Graph

Naoto Sonoda,[†1] Yukiko Yamauchi,[†1] Shuji Kijima[†1] and Masafumi Yamashita[†1]

The graph isomorphism (GI) problem is to decide whether two given graphs are isomorphic or not. It has not been known whether the GI problem can be solved in polynomial time. A polynomial time algorithm for the GI problem for 3 regular graphs is provided by Luks whichis based on the group theory. However, Luks focuses on the mathematical argument and does not refer to the implementation on the computer. In this work, we implemented Luks' algorithm and examined its behavior.

†1 九州大学
Kyushu University

## 1. はじめに

二つのグラフが与えられた時，それらが同型であるか否かを判定する問題をグラフ同型性判定問題という．グラフ同型性判定問題と相互に還元可能な計算問題は GI 完全という計算複雑性クラスに属している．GI 完全な問題が多項式時間で解決可能かどうかは長いあいだ未解決問題のままであり，さらに NP 完全であるか否かもわかっていない．そこでグラフ同型性判定問題については，入力グラフに条件を与えた場合の計算時間についても多くの研究がされており，その結果いくつかのグラフについては多項式時間で計算可能であることがわかっている[6]．例えば木や平面グラフを入力として与えた場合の同型性判定問題は多項式時間計算が可能である．そこで本研究では 3 正則グラフが入力として与えられた場合の多項式時間アルゴリズム[4] について研究する．

Luks によって与えられたアルゴリズム[4] は，3 正則グラフの同型性判定問題を 2 つの入力グラフを連結したグラフ $X$ の自己同型群を求める問題へと還元する．このアルゴリズムは 3 正則グラフの同型性判定問題について多項式時間で解を与える．しかし，このアルゴリズムは数学的な議論に終始しており，実際の群の計算については不明瞭であった．そこで，実際に Luks のアルゴリズムを計算機上で実装し，いかにして群の計算が行われているかを確認した．

## 2. 準　　備

グラフ同型性判定問題の定義や群論の基本概念についてはここでは省略する．本章ではアルゴリズムの実装を行うために必要な概念について述べる．

任意の有限集合を $X$ とおく．このとき $X$ から $X$ への全単射全体の集合は写像の合成を積とする群となる．これを $X$ の**対称群**とよび，$\mathrm{Sym}(X)$ と表記する．対称群の部分群を**置換群**という．対称群の単位元は各 $x \in X$ が自身に写像するような全単射であり，これを**恒等置換**という．

$\mathrm{Sym}(X)$ の任意の部分群を $G$，$X$ の任意の要素を $x$ とするとき，

$$G(x) = \{g(x) \mid g \in G\}$$

を $x$ の $G$ による**軌道** (あるいは $G$-軌道) という．また，任意の $Y \subseteq X$ に対して，

$$\{g(y) \mid g \in G, y \in Y\} = Y$$

が成り立つとき，$Y$ は***G*-安定**といわれる．さらに，$G(x) = X$ であるならば，群 $G$ は $X$ 上で**推移的**であるという．

$G$ を $X$ 上で推移的な群と仮定する．空でない部分集合 $Y \subset X$ が任意の $\sigma, \tau \in G$ に対して $\sigma(Y) = \tau(Y)$ または $\sigma(Y) \cap \tau(Y) = \emptyset$ であるならば，$Y$ を**$G$-block** という．また，$Y$ が $G$-block であるとき，集合族 $\{\sigma(Y) \mid \sigma \in G\}$ を $Y$ についての**$G$-block system** という．$G$-block system の各要素を **block** と呼ぶ．$Y \subset Z \subset X$ を満たすどの $Z$ も $G$-block でないとき，$Y$ による $G$-block system を**極小**と呼ぶ．

## 3. 3 正則グラフの同型性判定アルゴリズム

本章では Luks[4] によって設計された 3 正則グラフに対する同型性判定アルゴリズムを説明する．Luks のアルゴリズムでは，まずグラフ同型性判定問題を二つの入力グラフを連結したグラフ $X$ の自己同型群を求める問題へと還元する．次に，$X$ の自己同型群 $\mathrm{Aut}(X)$ を求めるために，本アルゴリズムでは $X$ の部分グラフの列 $X_1, ..., X_{n-1} = X$ について考える．この部分グラフの列を用いて $\mathrm{Aut}(X_{r-1})$ が分かっているときに $\mathrm{Aut}(X_r)$ を計算していき，帰納的に自己同型群を求めることができる．最後に，$\mathrm{Aut}(X_{r-1})$ から $\mathrm{Aut}(X_r)$ を求める際に，CAP と呼ばれる問題を利用する．

### 3.1　自己同型群を求める問題への還元

グラフ同型性判定問題の入力である二つの 3 正則グラフを $Q_1, Q_2$ とする．$Q_1$ と $Q_2$ の頂点数が異なる場合，これらのグラフが同型であることはない．ゆえに二つのグラフの頂点数は同じであると仮定し，それを $n$ とおく．このときグラフ同型性判定問題は次の定理が示すような還元を行うことができる．

**定理 3.1.** *3 正則グラフの同型性判定問題は 3 正則な連結グラフの自己同型群の生成系を求める問題に多項式時間で還元可能である．*

**証明**

$Q_1$ の適当な辺を $e_1$ として固定する．このとき $Q_2$ の各辺 $e_2$ に対して次のような操作を行い $Q_1$ と $Q_2$ を連結する．

1．$e_1, e_2$ のそれぞれの中間点に新しい頂点 $v_1v_2$ を導入する．

2．$v_1$ と $v_2$ を新たな辺でつなぎ，その辺の中間点に新たな頂点 $s$ を導入する．

このようにして連結したグラフを $X$ とする．$s$ を自身へ写像する (以下，これを固定するという) ような $X$ の自己同型群を $\mathrm{Aut_s}(X)$ と表記する．$\mathrm{Aut_s}(X)$ は対称群 $\mathrm{Aut}(X)$ の部分群である．なぜなら，$\mathrm{Aut_s}(X)$ の二つの元による積はやはり $s$ を固定するような全単射であるし，逆元も $s$ を固定する全単射である．これは $\mathrm{Aut_s}(X)$ が部分群であるという定義を満たしている．$\mathrm{Aut_s}(X)$ は恒等写像を含むのでからでないことに注意されたい．

このとき，$e_1$ を $e_2$ へ写像するような $Q_1$ から $Q_2$ への自己同型写像が存在する，かつその時に限り，$v_1$ を $v_2$ へ写像するような $\mathrm{Aut_s}(X)$ の元が存在する．さらに，もしそのような自己同型写像が存在するならば，そのうちのひとつは $\mathrm{Aut_s}(X)$ の生成元である．したがって，$X$ の自己同型群 $\mathrm{Aut_s}(X)$ の生成系を求めることで $Q_1$ と $Q_2$ が同型であるかを判別することができる．どのような $e_2$ を選んだ場合に対してもそのような自己同型写像が存在しないならば，$Q_1$ と $Q_2$ は同型ではないということになる．　□

以上から，3 正則グラフの同型性判定問題を解くために，連結グラフ $X$ の自己同型群を計算すればよいことがわかる．

### 3.2　自己同型群の計算

ここからは $\mathrm{Aut_s}(X)$ の生成系を求めるアルゴリズムについて述べる．

$X$ は前節の定理 3.1 の証明中で構成されたグラフである．$X_r$ を頂点 $s$ を始点とする長さ $r$ の全ての道を集めて得られる部分グラフとする．すなわち，$X_1$ は $s, v_1, v_2$ の三点とその間の辺のみから成るグラフとなり，$X_2$ は $X_1$ に加え $v_1, v_2$ に接続する辺とその端点から成るグラフとなる．そして $X_{n-1} = X$ となる．$X_r$ の例を図 1 に示す．ただし $V(X_{r-1})$ の 2 点間にあるような辺は $E(X_{r-1})$ には含まれず，$E(X_r)$ に初めて含まれることに注意する．このような部分グラフ $X_r$ を考えれば，$\mathrm{Aut_s}(X)$ の生成系が $\mathrm{Aut_s}(X_1)$ から $\mathrm{Aut_s}(X_{n-1})$ まで帰納的に求められることを述べる．

まず，$\mathrm{Aut_s}(X_1)$ は恒等置換と互換 $(v_1v_2)$ のみから成る群であることは明らかである．次に $\mathrm{Aut_s}(X_r)$ がわかっていることを仮定して $\mathrm{Aut_s}(X_{r+1})$ の求める方法を示す．そのために次の補題を導入する．

**補題 3.2.** $\mathrm{Aut_s}(X_r)$ に属する自己同型群のうち，$X_{r-1}$ の各頂点を固定するような全単射全てから成る集合を $K_r$ とする．すなわち，任意の $k \in K_r$ について，任意の $v \in v_{r-1}$ に対して，$k(v) = v$ が成り立つ．このとき $K_r$ は $\mathrm{Aut_s}(X_r)$ の正規部分群である．

$K_r$ は $\mathrm{Aut_s}(X_r)$ の正規部分群であるから，$K_r$ の生成系と商群 $\mathrm{Aut_s}(X_r)/K_r$ の完全代表系となるような適当な集合を求めることができれば，$\mathrm{Aut_s}(X_r)$ を求めることができる．

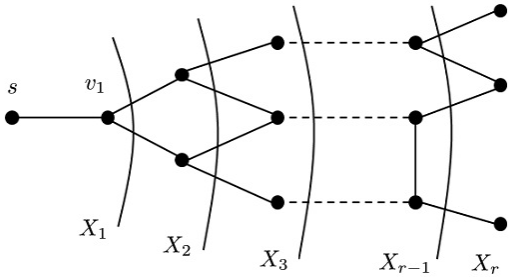


**図 1**　グラフ $X_r$ の例

まず，$K_r$ の生成系は次の補題で与えられる．

**補題 3.3.** 任意の $v \in V(X_r) \setminus V(X_{r-1})$ に対して

$$f(v) = \{w \in V(X_{r-1}) \mid vw \in E(X_r)\}$$

と定義する．このとき，$X_r$ 上の互換から成る集合 $T_r$ を

$$T_r = \{(u\ v) \mid u, v \in V(X_r) \setminus V(X_{r-1}), u \neq v, f(u) = f(v)\}$$

と定義する．ただし，$(u\ v)$ は $u$ と $v$ の互換を表す．このとき，$K_r$ は $T_r$ によって生成される．

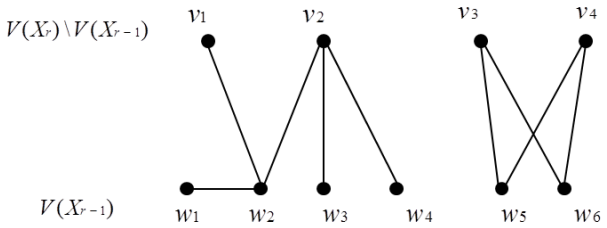


**図 2**　$f(v)$ の例

例えば，図 2 の各 $v \in V(X_r) \setminus V(X_{r-1})$ 対して $f(v)$ は，

$$f(v_1) = \{w_1\}, \qquad f(v_2) = \{w_1, w_2, w_3\}, \qquad f(v_3) = f(v_4) = \{w_4, w_5\}$$

である．また $T_r$ は $v_3, v_4$ のように，$f(v)$ が等しい 2 点の互換によって成り，$T_r = \{(v_3\ v_4)\}$ となる．

次に $\mathrm{Aut_s}(X_r)/K_r$ の完全代表系を求める方法について述べる．$\mathrm{Aut_s}(X_r)$ の各自己同型写像の定義域を $X_{r-1}$ に制限して得られる集合を $L_r$ とおく．このとき $L_r$ は $\mathrm{Aut_s}(X_{r-1})$ の部分群である．なぜなら，$\mathrm{Aut_s}(X_r)$ は $s$ を固定している写像であるから $V(X_{r-1})$ の頂点が $V(X_r)$ に写像されるようなことはなく，$V(X_{r-1})$ から $V(X_{r-1})$ への写像は全て $\mathrm{Aut_s}(X_{r-1})$ に含まれているからである．そこで $\mathrm{Aut_s}(X_{r-1})$ の生成系から適当な $L_r$ の生成系を求めることを考える．$L_r$ の生成系が求められれば，定義域の拡大はグラフが 3 正則であることから容易である．次の補題で $L_r$ が $\mathrm{Aut_s}(X_{r-1})$ のうちどのような特徴をもつものであるかを示す．

**補題 3.4.** $A = \{a \subseteq V(X_{r-1}) \mid 1 \leq |a| \leq 3\}$ とおく．また，$A_1, A_2, A'$ を次のように定義する．

$$A_1 = \{a \in A \mid \exists! u \in V(X_r) \setminus V(X_{r-1})[a = f(u)]\}$$
$$A_2 = \{a \in A \mid \exists u, v \in V(X_r) \setminus V(X_{r-1})[u \neq v \wedge a = f(u) = f(v)]\}$$
$$A' = \{\{u, v\} \mid u, v \in V(X_{r-1}), uv \in E(X_r)\}$$

このとき，$L_r$ は $\mathrm{Aut_s}(X_{r-1})$ に属する自己同型写像 $\sigma$ のうち，

$$\{\sigma(a) \mid a \in A_1\} = A_1 \quad \text{かつ}$$
$$\{\sigma(a) \mid a \in A_2\} = A_2 \quad \text{かつ}$$
$$\{\sigma(a) \mid a \in A'\} = A'$$

を満たすもの全てから成る．

例えば，図 2 においては，$A_1 = \{\{w_1\}, \{w_1, w_2, w_3\}\}$，$A_2 = \{\{w_4, w_5\}\}$ である．さらに，図 3 においては，$A_1 = \{\{w_2\}, \{w_3, w_4\}\}$，$A_2 = \{\{w_1\}\}$，$A' = \{\{w_2, w_3\}\}$ である．

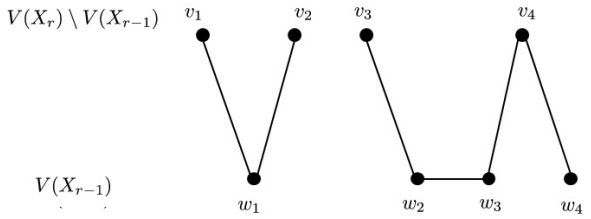


**図 3**　$A_1, A_2, A'$ の例

この補題から，$L_r$ は $\mathrm{Aut_s}(X_{r-1})$ がわかっていれば計算できることが分かる．次の節では $L_r$ の生成系を求める問題を別の計算問題へと還元して計算していくことを説明する．

### 3.3 Color Automorphism Problem (CAP)

Color Automorphism Problem (以下 CAP) と呼ばれる問題を次のように定義する．

**定義 3.1.** 各要素が“ 色 ”によって類別された任意の集合を色付き集合と呼ぶことにする．色付き集合の任意の要素 $a, b$ に対して，$a$ の色と $b$ の色が等しいことを $a \sim b$ と表記する．

いま $G$ を色付き集合 $A$ 上の任意の置換群とする．このとき $\sigma \in \mathrm{Sym}(A)$ および $B \subseteq A$ に対して，

$$C_B(\sigma G) = \{\pi \in \sigma G \mid \forall b \in B[\pi(b) \sim b]\}$$

と定義する．以上の準備の下，次のような入力と出力を持つ問題を $CAP$ と定める：

入力：色付き集合 $A$ 上の置換群 $G$ の生成系，$G$-安定集合 $B \subseteq A$，置換 $\sigma \in \mathrm{Sym}(A)$

出力：$C_B(\sigma G)$.

次に $L_r$ の生成系を求める問題を CAP に還元する方法を述べる．まず色付き集合 A は補題 3.4 で定義した $V(X_{r-1})$ の集合族を用いる．ここで A の各要素の色とは次のような 6 つのものを指す．ただし $A_0 = A - (A_1 \cup A_2)$ である．

$$A_0 \cap A',\quad A_1 \cap A',\quad A_2 \cap A',\quad A_0 \setminus A',\quad A_1 \setminus A',\quad A_2 \setminus A'.$$

実際には，$A_2 \cap A' = \emptyset$ であるので色は 5 つとなる．

また，写像 $\psi : \mathrm{Aut_s}(X_{r-1}) \to \mathrm{Sym}(A)$ を任意の $\gamma \in \mathrm{Aut_s}(X_{r-1})$ に対して

$$\psi(\gamma)(a) = \gamma(a) \quad (a \in A)$$

と定義すると，$\psi$ は $\mathrm{Aut_s}(X_{r-1})$ から $\mathrm{Sym}(A)$ の中への同型写像となる．そこで置換群 $G$ とその生成系としては $\mathrm{Aut_s}(X_{r-1})$ の像，$\psi(\mathrm{Aut_s}(X_{r-1}))$ を与える．そして，$B = A$ とし，$\sigma$ は恒等置換とする．

この問題の還元において σ を恒等置換にしたにもかかわらず，CAP でわざわざ $\sigma$ を入力として与えていることには理由がある．それは CAP を解く上で再帰的に CAP を呼び出す場合があるからであり，その中には入力に恒等置換でない $\sigma$ を与えるような CAP も解く必要があるからである．

さて，CAP の解法について述べる前に $C_B(\sigma G)$ のサイズについて言及する．$\sigma G$ は置換群 $G$ の左剰余類であり，この入力から出力される集合 $C_B(\sigma G)$ は群になっているとは限らず生成系でもつことができない．さらに $C_B(\sigma G)$ の要素数は指数関数的になる可能性がある．この問題を解決するために次の補題を導入する．

**補題 3.5.** 任意に与えられた $\sigma \in \mathrm{Sym}(A)$ に対して，もし $C_B(\sigma G)$ は空集合でないならば，$C_B(\sigma G)$ は $C_B(G)$ の左剰余類である．

この補題によって，$C_B(\sigma G)$ は $\sigma$ と $C_B(G)$ の生成系で出力できることがわかった．そこで，ここからは CAP を求めるアルゴリズムについて説明する．アルゴリズムは以下のような入力ごとに別の動作を行う．

(1)　$|B| = 1$ のとき．
(2)　$|B| > 1$　かつ　$G$ が $B$ 上で推移的でないとき．
(3)　$|B| > 1$　かつ　$G$ が $B$ 上で推移的であるとき．

まず 1 の場合について述べる．このとき $B = \{b\}$ とおき，$\sigma(b) \sim b$ であるか否かを調べる．$\sigma(b) \sim b$ のときには $B = \{b\}$ が $G$-安定集合であることから，任意の $\tau \in G$ に対して $\tau(b) = b$ となり，$\sigma\tau(b) \sim b$ が成り立つ．よって，このときは $\sigma G$ そのものを出力するとよい．一方で，$\sigma(b) \not\sim b$ であるならば，やはり $B = \{b\}$ が $G$-安定集合であることから，任意の $\tau \in G$ に対して $\sigma\tau(b) \not\sim b$ が成り立つ．ゆえに出力は空集合 $\emptyset$ である．

次に 2 の場合について述べる．このとき $B$ は G-安定な二つの部分集合 $B'$ と $B''$ の和集合となっている．このとき $C_B(\sigma_0 G)$ は次のようなものである．

$$C_B(\sigma G) = C_{B''}(C_{B'}(\sigma G))$$

これは，一方の部分集合 $B'$ と $\sigma G$ を入力として与えた CAP を先に計算した後，得られた出力である置換と部分群をもう一方の部分集合 $B''$ とともに入力した CAP を計算している．つまり，この場合には分割した各集合に対して $C_{B'}(\sigma G) = \sigma_0 C_{B'}(G)$ となる $\sigma_0$ を求めた後，$C_{B''}(C_{B'}(\sigma G))$ を求めればよい．

最後に 3 の場合について述べる．このときは $B$ のどんな部分集合も G-安定ではないので CAP としてそのまま再起をかけることはできない．そこで次の二つの補題を導入する．

**補題 3.6.** 集合 $D(|D| > 1)$ 上で推移的かつ $p$ 群となっている置換群を $G$ とする．このと

き $D' \subset D$ による $G$-*block system* が極小ならば丁度 $p$ 個の block から成る．さらに，全ての block が安定となるような $G$ の部分群 $H$ が存在する．

**補題 3.7.** 置換群 $G$ の生成系と $G$ によるある軌道 $B$ が与えられた時，$B$ 上で極小な $G$-block system は多項式時間で見つけることができる．

また，$\mathrm{Aut_s}(X_{r-1})$ $(r \geq 2)$ が 2 群であることを $r$ についての数学的帰納法を用いて述べる．まず，$\mathrm{Aut_s}(X_1)$ が 2 群であることは明らかである．次に，$\mathrm{Aut_s}(X_{r-1})$ が 2 群であると仮定すると，その部分群である $L_r$ は 2 群である．さらに，$K_r$ は複数の互換を生成元とする群であるから，やはり 2 群である．$\mathrm{Aut_s}(X_r)$ は $L_r$ と $K_r$ によってつくられる群であるので 2 群となっている．よって任意の $r \geq 1$ について $\mathrm{Aut_s}(X_r)$ が 2 群であることがいえる．

$\mathrm{Aut_s}(X_{r-1})$ が 2 群となっていることから，補題 3.6 により部分集合 $B$ は極小な $G$-block system，$B'$，$B''$ が存在し，補題 3.7 により多項式時間で見つけることができる．さらに，それを安定させるような $G$ の部分群 $H$ と $\tau \in G$ で $G = H \cup \tau H$ を満たすようなものを多項式時間で見つけることができる．これらの具体的な計算方法は第 3 章で述べる．このとき $C_B(\sigma_0 G)$ は次のようなものである．

$$C_B(\sigma G) = C_{B''}(C_{B'}(\sigma H)) \cup C_{B''}(C_{B'}(\sigma\tau H))$$

すなわち，この場合には $H$，$\tau H$ の各入力と集合の各分割に対して 4 回の再帰的な計算を行うことで，$C_B(\sigma G)$ を計算することができる．

ただし，補題 3.5 で述べたように，$C_{B''}(C_{B'}(\sigma H))$ と $C_{B''}(C_{B'}(\sigma\tau H))$ の計算結果は $C_B(H)$ の左剰余類として得られるため，そのままでは和集合をとることができない．仮に計算結果から集合の要素をすべて計算して和集合を求めるとするならば，それは入力に対して指数的な要素数を持つ可能性がある．これを解決するために次の補題を述べる．

**補題 3.8.** $C_{B''}(C_{B'}(\sigma H))$ と $C_{B''}(C_{B'}(\sigma\tau H))$ の計算結果をそれぞれ

$$C_{B''}(C_{B'}(\sigma H)) = \rho_1 C_B(\sigma G), \qquad C_{B''}(C_{B'}(\sigma\tau H)) = \rho_2 C_B(\sigma G)$$

とおく．このとき，

$$C_{B''}(C_{B'}(\sigma H)) \cup C_{B''}(C_{B'}(\sigma\tau H)) = \rho_1 \langle C_B(H) \cup \rho_1^{-1}\rho_2 \rangle$$

と表せる．

以上より，3 の場合の計算方法も得られ，各場合についての計算方法が示された．

## 4. アルゴリズムの計算機上での実装

3 章では Luks[4] による 3 正則グラフにおけるグラフ同型性判定問題のアルゴリズムについて述べた．この章では前章のアルゴリズムの計算機上における実装について述べる．まず，アルゴリズムの計算機上での実装法を述べる．後に計算速度の解析や改善点についての考察を行う．

### 4.1 アルゴリズムの実装

Luks[4] によるアルゴリズムは，前章の各節のように大きく分けて 3 つの部分から成っている．このうち，前章においてアルゴリズムがよくわからないのは CAP についてであった．本節では CAP の実装について述べる．CAP を求めるアルゴリズムは以下のとおりである．

**Algorithm 1**

**Input.** 色つき集合 $A$ 上での置換群 $G$，$B \subseteq A$，$\sigma \in \mathrm{Sym}(A)$
**Output.** $C_B(\sigma G)$

1. **if** $|B| = 1$
2. 　　**then if** $\sigma(b) \sim b (b \in B)$
3. 　　　　**then return** $\sigma G$
4. 　　　　**else return** $\emptyset$
5. **else if** $G$ が $B$ 上で推移的でない
6. 　　**then** ある $b \in B$ にたいして $B' = G(b)$ とし，$B'' = B - B'$ とする．
7. 　　　　**return** $C_B(\sigma G) = C_{B'}(C_{B''}(\sigma G))$
8. 　　**else** $B$ についての極小な $G$-block system $B'$，$B''$ を計算する．
9. 　　　　$B_1$ と $B_2$ を安定させるような $H$ と $\tau \in G$ で $G = H \cup \tau H$ となるものを見つける．
10. 　　　　**return** $C_B(\sigma G) = C_{B''}(C_{B'}(\sigma H)) \cup C_{B''}(C_{B'}(\sigma\tau H))$

この Algorithm 1 は 3 章で述べたものを疑似コードとして書いただけである．このコードのうち，掘り下げる必要があるのは，5，6 行目における軌道の計算，8 行目における block

system の計算，9 行目における $\tau$, $H$ の計算である．このうち，まず軌道の計算について述べる．

ここで $G$ の生成系の集合を $\{g_1, ..., g_k\}$ とし，$B$ の部分集合を $O$ とおいておく．適当に選んだ $b \in B$ についての軌道を求めるアルゴリズムを Algorithm 2 として示す．

**Algorithm 2**

**Input.** $G$ の生成系 $\{g_1, ..., g_k\}$，集合 $B$，$b \in B$
**Output.** $G(b)$
0. キューを空にし，$O \leftarrow \emptyset$ とする．
1. $O$ に $b$ を追加し，キューにも挿入する．
2. **while** キューが空でない
3. 　　**for** $i \leftarrow 1$ **to** $k$
4. 　　キューから 1 つ要素を取り出し $b'$ とおく．
5. 　　**if** $g_i(b') \notin O$
6. 　　**then** $O$ に $g_i(b')$ を追加し，キューにも挿入する．
7. **return** $O$

次に $G$-block を求めるアルゴリズムについて述べるが，そのために次の補題を導入しておく．

**補題 4.1.** 群 $G$ のうち $b \in B$ を固定するような部分群を $G_b$ とする．軌道 $G(b)$ の各要素と $G_b$ による各左剰余類は *1* 対 *1* で対応させることができる．

この補題から次の系も導ける．

**系 4.2.** 群 $G$ と $G_b$，$G(b)$ について次の式が成り立つ．

$$|G| = |G_b| \cdot |G(b)|$$

したがって，$|G(b)|$ は $|G|$ の約数である．

さて，$G$-block system の計算を必要とする場合，$G$ は $B$ 上で推移的となっていた．すなわち $B$ は $G$ による軌道であり，$G$ が 2 群であることから $|B|$ も 2 のべき乗である．さらに，$G$-block system の定義は $G$-block であるような $B$ の部分集合について軌道をとった集合族となってることに注意する．つまり，いかなる $G$-block system の要素数も $|G|$ の約数であり，$G$ が 2 群ならば 2 のべき乗なのである．これらの事実に注意しつつ，$G$-block system を求めるアルゴリズムを集合 $B$ の分割であるような集合族を $B'$ とおいて次のようにを示す．

**Algorithm 3**

**Input.** $G$ の生成系 $\{g_1, ..., g_k\}$，集合 $B = \{b_1, ..., b_m\}$
**Output.** 極小な $G$-block system$\{B_1, B_2\}$
0. $B' \leftarrow \{\{b_1\}, \{b_2\}, ..., \{b_m\}\}$
1. **for** $i \leftarrow 2$ **to** $m$
2. 　　Atkinson のアルゴリズムにより集合族 $B'$ 上で $b_1$ と $b_i$ を含む最小の $G$-block を求める．
3. 　　その $G$-block から $G$-block system を計算する．
4. 　　**if** 求めた $G$-block が $B$ 全体から成るものではない
5. 　　**then** $B' \leftarrow$ 計算で得られた $G$-block system
6. **return** $B'$

このようにして求めた $B'$ は極小な $G$-block system となっている．

最後に $H$ と $\tau$ を求めるアルゴリズムを述べる．補題 4.1 より $B_1$ を固定するような部分群 $H$ を持ってくれば $H$ による左剰余類は $B_1$, $B_2$ と 1 対 1 で対応することが分かる．この $H$ による左剰余類を $H$, $\tau H$ とすれば，これは確かに $B_1$, $B_2$ が $H$-安定であり $G = H \cup \tau H$ となっている．そこで $H$ と $\tau$ を求めるアルゴリズムは次の Algorithm 4 のようになる．

**Algorithm 4**

**Input.** $G$ の生成系 $\{g_1, ..., g_k\}$，$G$-block system$\{B_1, B_2\}$
**Output.** $H$，$\tau$
0. $H = \emptyset$
1. **for** $i \leftarrow 1$ **to** $k$
2. 　　**if** $g_i$ が $B_1$ を安定させる
3. 　　　　**then** $g_i$ を $H$ に追加する．
4. 　　**else** $g_i$ がはじめて $H$ に入らなかったものならば $\tau$ に選ぶ．
5. 　　　　そうでなければ $\tau^{-1}g_i$ を $H$ に追加する．
6. **return** $H$，$\tau$

$g_i \notin H$ の時，$H$ による左剰余類は二つしかないので $g_i \in \tau H$ である．ゆえに $\tau^{-1}g_i \in H$ である．したがって $H$ と $\tau$ も得ることができたので，Algorithm 1 が計算できるようになり，CAP の計算を実装できた．

### 4.2　計算時間と考察

次に，計算の実行時間について述べる．まず実験環境を述べる．

**実験環境**

| | |
|---|---|
| 使用言語 | C 言語 |
| OS | Windows7 Home Premium |
| CPU | Intel Core i5 2.80GHz |
| メモリ | 4GB |
| 時間の計測法 | C 言語の clock 関数の使用 |

また本アルゴリズムは，入力された 2 つのグラフが同型である場合と，そうでない場合には出力時間に差が出る．なぜなら，$Q_2$ のある辺を選んだときに同型であると判断できるならば即“ Yes ”を出力して実行を終了するが，同型でない場合には，$Q_2$ の全ての辺に対して，自己同型群を求める必要があるからである．そこで，まず“ Yes ”と“ No ”の出力別に入力インスタンスの作り方を述べる．

$Q_1$ の構成法は共通である．まず $V(Q_1)$ に対する完全グラフを考え，このグラフ上でランダムに完全マッチングを作成し，$E(Q_1)$ とする．次に，もう一度この完全グラフ上でランダムに完全マッチング $E'$ を作成し，$E(Q_1) \cap E' = \emptyset$ ならば，$E(Q_1) = E(Q_1) \cup E'$ とする．そうでなければもう一度マッチングを行う．同じことをもう一度繰り返すと，各頂点は 3 つの異なる頂点との辺をもつことになり，3 正則なグラフが完成する．このようにして構成したグラフを $Q_1$ として入力する．

次に“ Yes ”の出力時間を計測する際の $Q_2$ の構成法について述べる．これは $V(Q_1)$ から $V(Q_2)$ へのランダムな全単射 $\varphi$ をつくり，任意の $u$，$v \in V(Q_1)$ について，$uv \in E(Q_1)$ ならば，$\varphi(u)\varphi(v) \in E(Q_2)$ となるように，辺を構成する．すると，$Q_1$ から $Q_2$ への同型写像は $\varphi$ のようなものが必ず一つは存在することになるので“ Yes ”を出力する．

最後に“ No ”の出力時間を計測する際の $Q_2$ の構成法について述べる．これは最初の $Q_2$ と同様の方法でランダムな 3 正則グラフを構成する．ただし，このとき偶然に同型なグラフができることもある．このときは“ Yes ”を出力するので“ No ”の計算時間としてカウントはしない．

さて，これらの入力グラフを実験を行う．実験は上記の入力グラフを毎回作成し，“ Yes ”と“ No ”でそれぞれ 50 回ずつ計算を行った．その平均実行時間をまとめた結果が図 4 である．

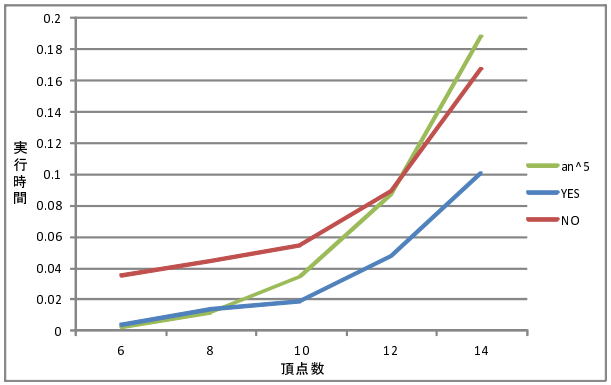


**図 4**　実行時間

図 4 中の近似曲線は $an^5$( ただし $a = 3.5 * 10^{-7}$) である．本実験では，頂点数 16 以上の場合にはメモリがオーバーフローを起こすため，頂点数 14 までの結果しか得られていない．

3 正則グラフの頂点数は偶数であることに注意されたい．このオーバーフローの理由については後述する．

さて，いま我々が欲しい結果は，本アルゴリズムが多項式時間で動作しているか否かであった．そこで，実行時間に対する片対数グラフを図 5 に両対数グラフを図 6 に示す．

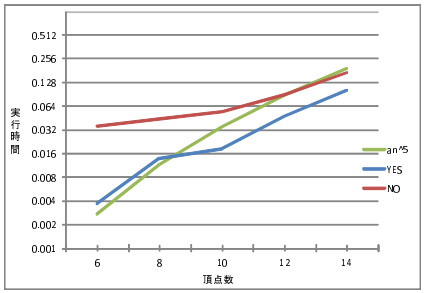


**図 5**　実行時間についての片対数グラフ

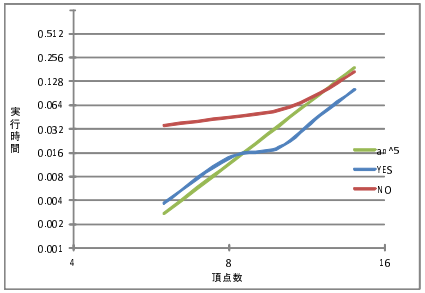


**図 6**　実行時間についての両対数グラフ

実行時間が多項式時間であるならば，片対数グラフでは上に凸の線となり，両対数グラフでは直線となるはずである．しかし，実験の結果では図 5 では下に凸の線となっており，図 6 でも同様に下に凸の線となっている．ところが，頂点数 12 から 14 に限ってみれば近似曲線である $an^5$ よりも緩やかになっている．今回は頂点数の少ない場合にしかアルゴリズムの実行時間を確認できなかったため，結論を得るためにはより多くの頂点で実験を行うことが必要であると考えられる．

もし計算時間が指数的に増加しているとするならば，CAP の計算に指数時間かかっている可能性が考えられる．CAP を解くアルゴリズムは再帰的に CAP を解くものであった．この再帰回数は入力である色つき集合の大きさで抑えられることを節で述べた．しかし，その実装が正しくなければ指数時間かかっている可能性がある．

また，より多くの頂点で実験を行うためには，記憶量の改善が必要となる．記憶量域として問題となっているのは，CAP の入力として与えられる $A$ であると考えられる．$A$ は $V(X)$ についてのサイズ 3 以下の集合族であり，そのサイズは $O(n^3)$ である．この値は記憶領域としては非常に大きい数字である．さらに，前述の CAP が再帰的に指数回呼ばれているとするならば，CAP に用いる関数の変数が指数個必要となっている可能性がある．これは，さきほどの $A$ のサイズよりも非常に大きな問題である．したがって，CAP の実装について見直すことが今後の課題である．

## 5. おわりに

本論文では 3 正則グラフの同型性判定問題を解くアルゴリズムについて述べ，そのアルゴリズムの計算機上での実装と実験による計算時間の確認を行った．その結果，現在の実装では頂点数が 6〜14 までの実験しかできなかったため，多項式時間で計算できているかどうかの確認を行うことができなかった．実装を行った結果，本アルゴリズムを改良することによって多項式時間でグラフの同型写像の数え上げを行うことが可能であると考える．詳細は今後の課題とする．

実装についての今後の課題は，まず問題が疑われる CAP の実装についての確認を行う．また頂点数が大きなグラフを扱うにはメモリのオーバーフローが発生することが判明したので，計算中に保持する記憶量が少なくなるようにアルゴリズムを改善する必要がある．

## 参　考　文　献

1) M. D. Atkinson, An algorithm for finding the blocks of a permutation group, Mathematids of Computation, 29 (1975), pp.911–913.
2) Z. Galil, C. F. Hoffmann, E. M. Luks, C. P. Schnorr and A. Weber, An $O(n^3 \log n)$ deterministic and an $O(n^3)$ Las Vegas isomorphism test for trivalent graphs, Journal of the Association for Computing Machinery, 34 (1987), pp.513–531.
3) 国吉秀夫（高橋豊文　改訂），群論入門 [新訂版]，サイエンス社，2004.
4) E. M. Luks, Isomorphism of graphs of bounded valence can be tested in polynomial time, Journal of Computer and System Sciences, 25 (1981), pp.42–65.
5) 志賀浩二，群論への 30 講，朝倉書店，1989.
6) 戸田誠之助，グラフ同型性判定問題，日本大学文理学部叢書，2001.