数理解析研究所講究録
第1652巻 2009年 125-131

GAPを用いたRubik's Cube解法表示ソフトについて

田崎 拓馬
TAKUMA TASAKI
オービック
OBIC CO., LTD.

藤本 光史
MITSUSHI FUJIMOTO
福岡教育大学
FUKUOKA UNIVERSITY OF EDUCATION*

1 Rubik's Cube と数式処理

Rubik's cube は、ハンガリーの建築学者 Ernö Rubik によって 1974 年に考案された立方体パズルである。1980 年代に世界的なブームとなり、6 面を揃える速さを競う「スピードキュービング」は世界大会が開かれるほど愛好者が多い。一般に 3 × 3 × 3 のタイプが Rubik's Cube と呼ばれるが、この他に 2 × 2 × 2 (Pocket Cube)、4 × 4 × 4 (Rubik's Revenge)、5 × 5 × 5 (Professor's Cube) のタイプがある。

数式処理分野における Rubik's cube の研究としては、D. Kunkle と G. Cooperman が ISSAC2007 で、「3 × 3 × 3 の Rubik's cube を解くために必要な手数の上限は 26 手[1)]以下である」ことを大規模な計算によって示した [1]。

Rubik's cube の操作は置換で表現でき、群論を用いて解法を求めることが可能である。我々は群論計算が可能な数式処理ソフト GAP [2] を用いて Rubik's cube の解法を求め、それを Java3D を使ってグラフィカルに提示するソフトウェアを開発した。本稿では、このソフトウェアについて解説する。

2 Rubik's Cube と群

Rubik's cube の緑・黄・青・白・赤・橙の各面を時計と逆周りに 90 度回転させる操作 (各面の中心は回転するだけで移動しないことに注意) をそれぞれ g, y, b, w, r, o とおく。また、緑・黄・青・白・赤・橙の各面を時計周りに 90 度回転させる操作をそれぞれ $g^{-1}, y^{-1}, b^{-1}, w^{-1}, r^{-1}, o^{-1}$ で表す。Rubik's cube のすべての操作 (言い換えればすべての状態) は、この $g, y, b, w, r, o, g^{-1}, y^{-1}, b^{-1}, w^{-1}, r^{-1}, o^{-1}$ の積で表現できる。

さらに、Rubik's cube の各面の中心を除くピースに図 1 のように 1 から 48 の番号を付ける。この番号によって、操作 g, y, b, w, r, o は次のような置換表示を得る。

$$
\begin{aligned}
g &= (17,19,24,22)(18,21,23,20)(6,25,43,16)(7,28,42,13)(8,30,41,11)\\
y &= (33,35,40,38)(34,37,39,36)(3,9,46,32)(2,12,47,29)(1,14,48,27)\\
b &= (25,27,32,30)(26,29,31,28)(3,38,43,19)(5,36,45,21)(8,33,48,24)\\
w &= (9,11,16,14)(10,13,15,12)(1,17,41,40)(4,20,44,37)(6,22,46,35)\\
r &= (1,3,8,6)(2,5,7,4)(9,33,25,17)(10,34,26,18)(11,35,27,19)\\
o &= (41,43,48,46)(42,45,47,44)(14,22,30,38)(15,23,31,39)(16,24,32,40)
\end{aligned}
$$

*fujimoto@fukuoka-edu.ac.jp

1) ある面を 180 度回転する場合も 1 手と数える。この定義は face-turn metric と呼ばれる。

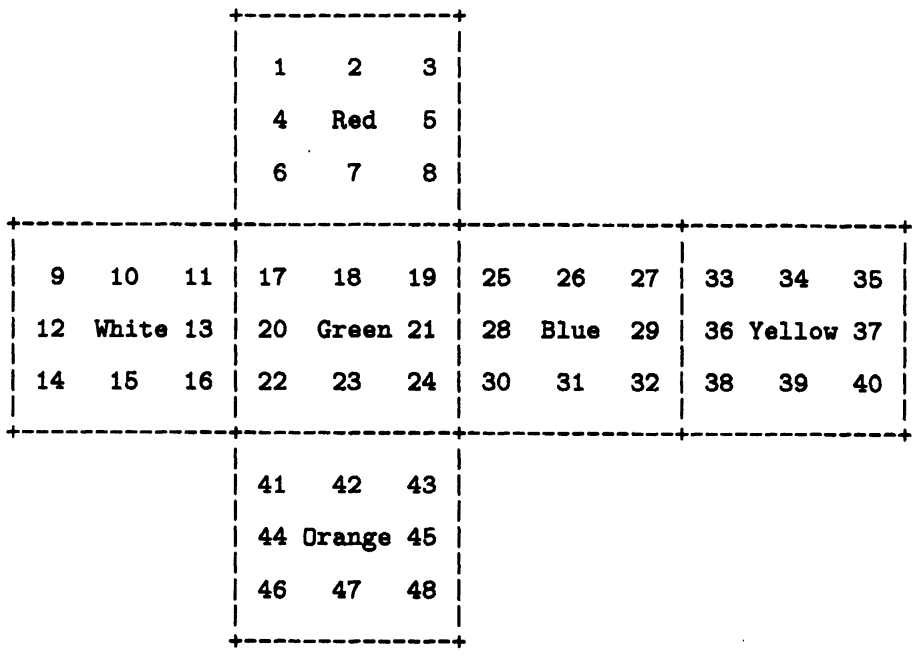

図 1: 各面の番号付け

これらを生成元とする置換群（48 次対称群の部分群）として Rubik's cube 群が定義できる。GAP 上で定義するには、次のように入力すればよい。

```
gap> g := (17, 19, 24, 22)(18, 21, 23, 20)(6, 25, 43, 16)(7, 28, 42, 13)(8, 30, 41, 11);;
gap> y := (33, 35, 40, 38)(34, 37, 39, 36)(3, 9, 46, 32)(2, 12, 47, 29)(1, 14, 48, 27);;
gap> b := (25, 27, 32, 30)(26, 29, 31, 28)(3, 38, 43, 19)(5, 36, 45, 21)(8, 33, 48, 24);;
gap> w := (9, 11, 16, 14)(10, 13, 15, 12)(1, 17, 41, 40)(4, 20, 44, 37)(6, 22, 46, 35);;
gap> r := (1, 3, 8, 6)(2, 5, 7, 4)(9, 33, 25, 17)(10, 34, 26, 18)(11, 35, 27, 19);;
gap> o := (41, 43, 48,46)(42, 45, 47, 44)(14, 22, 30, 38)(15, 23, 31, 39)(16, 24, 32, 40);;
gap> cube := Group(g,y,b,w,r,o);
```

これで GAP 上で Rubik's cube 群に関する計算を行うことができる。例えば、Rubik's cube 群の位数は次で求められる。

```
gap> Size(cube);
```

ここで得られる数 43,252,003,274,489,856,000 は Rubik's cube の製品パッケージにも組み合わせ総数として記載されている。

3 Rubik's Cube 問題を GAP で解く

ここでは、Rubik's cube に関するいくつかの問題を GAP を使って解いてみる。

3.1 位数問題 – 一連の操作を何回行うと元に戻るか

完成した状態の Rubik's cube で、例えば $y \to g^{-1} \to w^{-1} \to b^{-1}$ という一連の操作を繰り返すと必ず元の状態に戻る。これは Rubik's cube 群が有限群であるため、すべての元は有限位数を持つからである。一連の操作を何回繰り返せば元に戻せるかという問題は、その操作に対応した元の位数を求めることに他ならない。GAP で元 $y*g^{-1}*w^{-1}*b^{-1}$ の位数 ($(y*g^{-1}*w^{-1}*b^{-1})^m = 1$ となる m) を求めるには、次のように入力すればよい。

```
gap> Order(y*g^-1*w^-1*b^-1);
12
```

つまり、この例では 12 回で元に戻る。

3.2 メンバーシップ問題 – 与えられた模様は実現可能か

Rubik's cube には、完成させた後も様々な模様を作るという遊び方がある。しかし、自分で考えた模様が実現可能かどうかを判断するのは容易ではない。これは、その模様の置換表現が Rubik's cube 群に属するかどうかというメンバーシップ問題である。

次の 2 つの模様が実現可能かどうか調べてみる。図中の R は赤、G は緑、B は青、O は橙を表している。

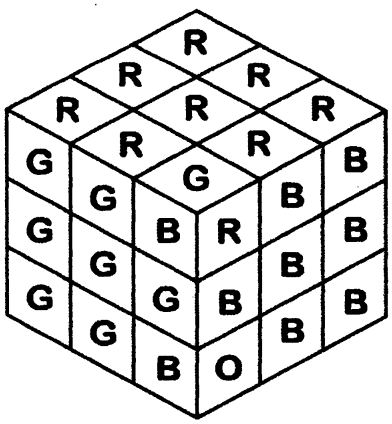

図 2: Cube 1

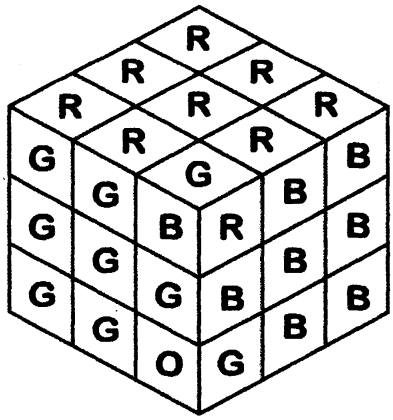

図 3: Cube 2

Cube 1、Cube 2 の置換表現はそれぞれ (8,19,25)(24,30,43)、(8,19,25)(24,43,30) である。GAP では、以下のように入力する。

```
gap> (8,19,25)(24,30,43) in cube;
true
gap> (8,19,25)(24,43,30) in cube;
false
```

この結果により、Cube 1 は実現可能で、Cube 2 は実現不可能であることがわかる。

3.3 生成元の積で表す問題 – Rubik's cube の完成手順を求める

バラバラになった Rubik's cube を完成させる手順を求めるには、バラバラの Rubik's cube の状態を置換で表現し、それを Rubik's cube 群の元と見て、生成元 g, y, b, w, r, o の積による表示を求めればよい。その表示の逆元が完成手順となる。

ここでは、次の脇克志氏による GetWordOfElements 関数 [4] を利用する。

```
GetWordOfElements:=function(G,GenName,x)
  local gen,F,hom;
  F:=FreeGroup(GenName);
  gen:=GeneratorsOfGroup(G);
  hom:=GroupHomomorphismByImages(F,G,GeneratorsOfGroup(F),gen);
  return PreImagesRepresentative(hom,x);
end;
```

(8,19,25)(24,30,43) で表される状態は、前節で計算したように実現可能である。これの生成元の積による表示を GAP で求めるには、次のように入力する。

```
gap> p:=GetWordOfElements(cube,["g","y","b","w","r","o"],(8,19,25)(24,30,43));
g*r*w*g*w^-1*g^-1*r^-1*g^-1*b^-1*g^-1*o^-1*g*o*b*g^2*b*g*b^-1
*r*g^-1*r^-1*g*b*g^-1*b^-1*g*r^-1*b*r*b^-1*g
```

ここで得られたものの逆元が完成手順を表す。

```
gap> p^-1;
g^-1*b*r^-1*b^-1*r*g^-1*b*g*b^-1*g^-1*r*g*r^-1*b*g^-1*b^-1
*g^-2*b^-1*o^-1*g^-1*o*g*b*g*r*g*w*g^-1*w^-1*r^-1*g^-1
```

実際に操作する場合は、この結果を右から順に実行していけばよい。

4 Rubik's cube 解法表示ソフト

4.1 ソフトウェアの概要

3.3 で解説した方法により解法手順を求めることは可能であるが、入力として必要な Rubik's cube の状態を表す置換表現を求めることは簡単ではない。また、GAP により得られた解法手順をグラフィカルに表示することは、結果の正当性を確認するためにも重要である。今回開発した Rubik's cube 解法表示ソフトはこれらを解決するためのインターフェースを有する。

Rubik's cube 解法表示ソフトが有する機能は以下の通りである。

- マウスやタブレットでドラッグすることにより、回転・移動・拡大が可能。
- 各面の色のボタンをクリックすることにより、その色の面を反時計回りに 90 度回転。
- Reset ボタンを押すことで初期状態に戻す。
- Send ボタンを押すことで、現在の Rubik's cube の状態 (置換表現) を GAP に渡し、計算を開始。
- Result ボタンで GAP が計算した解法手順に従い Rubik's cube を自動操作。

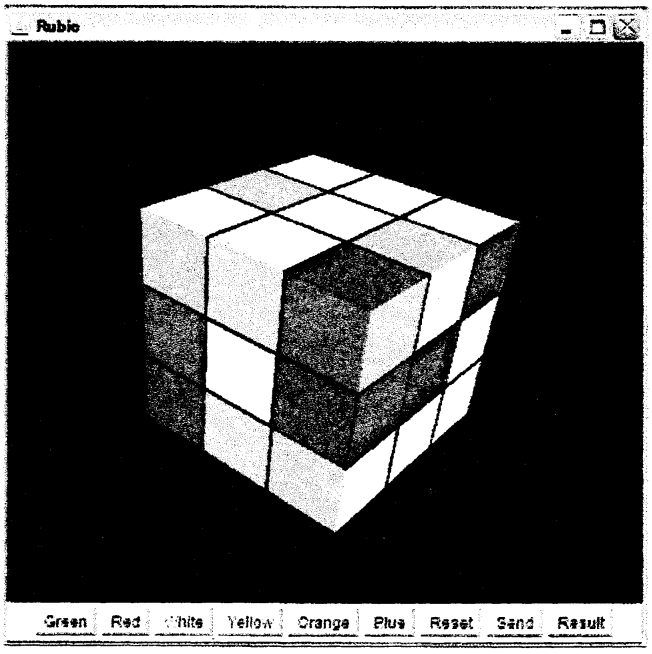

図 4: Rubik's cube 解法表示ソフトのスクリーンショット

4.2 実装について

本ソフトウェアは、次の３つのコンポーネントから成る。

1. **ユーザーインターフェース部** Java3D を用いて Rubik's cube を 3D オブジェクトとして画面内に実現。ドラッグによる回転・移動・拡大は、Java3D の機能を利用。各面の色のボタンをクリックする度に、Rubik's cube の状態 (置換表現) を計算する。

2. **解法エンジン部** Rubik's cube の状態 (置換表現) が入力されると、GAP と通信して、3.3 で解説した方法により解法手順を計算する。

3. **GAP との通信部** OpenMath や OpenXM は使用せず、原始的なファイルを介した通信方法で実現。

5 本ソフトウェアの活用法

本ソフトウェアは Rubik's cube の解法に群論を用いている。よって、大学での群論の講義で、置換の話や Lagrange の定理の理解を解説する時の学習ツールとして活用できる。しかしながら、本ソフトウェアを開発した動機は別にある。我々は、パズルの解法に数学が利用できることを子ども達に体験してもらうためにこのソフトを開発した。数学を学ぶ意義が理解できず、「なぜ数学を学ぶのか」と疑問を感じる子ども達は多い。この疑問に対して、一般には「世の中の見えない所で役立つから」、「論理的な思考力を養うため」、「受験に必要だから」といった答えが唱えられている。数学にこのような側面があることは否定しないが、「数学を学ぶのは面白いからである」という原点を伝える必要があるのではないだろうか。我々は、この点において本ソフトが利用できるのではないかと考えている。

例えば、ジュニアサイエンスなどの子ども達のための科学イベントにおいて本ソフトを使用すれば効果的だろう。また、高校生に対しては、「群論」という数学分野に興味を持ってもらうことが期待できる。受

験勉強を通じて「式の計算を行うことが数学である」という認識を持つ高校生には新鮮に感じられるのではないだろうか。

6 今後の課題と発展

本ソフトウェアの改良すべき点としては、次が挙げられる。

- GAP との通信に OpenMath や OpenXM を利用する。
- Rubik's cube の現在知られている最も高速な解法である LBL 法を用いた高速解法モードを追加する。
- ボタンではなく、マウスやペンによるキューブ操作を実現する。

最後に、本ソフトウェアをベースとして今後開発を予定している 2 つのシステムについて紹介する。

■ Rubik's cube 模様作成支援システム

3.2 でも取り上げたことであるが、Rubik's cube で新しい模様を作ることは面白い遊び方の一つである。しかし、勝手に色の配置を変更して新しい模様をデザインしても、その模様が実現できる (完成状態からその模様に変形できる) とは限らない。また、Rubik's cube を分解した経験のある人なら容易に理解できると思われるが、分解したキューブを闇雲に組み上げても完成状態に戻せないキューブができることもある。

このように、Rubik's cube には物理的な制約があるため、実現できる模様をデザインすることは難しい。そこで、コンピュータの画面上で、キューブの 20 個のピースをバラバラにした状態からマウスやペンを使って組み上げ、様々な模様を作成できるようにする。そして、その模様が実現可能かどうかを GAP で判定する模様作成支援システムを開発したい。

■ Rubik's cube ナビゲーションシステム

Rubik's cube の解法を覚えるのは非常に難しい。そこで、実物の Rubik's cube の状態を Web カメラで撮影し、画像解析により色の配置を求め、その情報から解法を計算し、次の状態までの操作方法を音声で指示するナビゲーションシステムを構築したい。

これが実現できれば、途中で間違えた場合でも、再び Web カメラで撮影すれば、その状態からの新しい解法を提示してくれる。まさにカーナビと同じ感覚である。また、手順を音声で提示できれば、視覚障害者が解法をマスターする際にも利用できると考えている。

参 考 文 献

[1] D. Kunkle, G. Cooperman, Twenty-six moves suffice for Rubik's cube, Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC2007), ACM Press, (2007) 235–242.

[2] GAP – Groups, Algorithms, Programming – a System for Computational Discrete Algebra, http://www.gap-system.org/

[3] Martin Schönert, Analyzing Rubik's Cube with GAP, http://www.gap-system.org/Doc/Examples/rubik.html

[4] 脇 克志, 月刊 GAP, http://sci.kj.yamagata-u.ac.jp/~waki/jpn/MonthlyGAP/

[5] 田崎拓馬, ルービックキューブ解法支援ソフトの開発について, 福岡教育大学卒業論文, (2007)