Vol.2017-AAC-5 No.3
2017/12/8

スライドパズルにおける回転型操作とアクセシビリティ

藤本 光史[1,a)]

概要：著者はスライドパズルの一種である 15 パズルとその拡張である $mn-1$ パズルが loop generators という回転型操作によって必ず解くことができることを数学的に証明した．そして，完成までの手数が通常の操作と比較して劣らないことを計算実験により示した．本稿ではその詳細を解説すると共に，その操作系を用いた 15 パズルのわかりやすい解法アルゴリズムを示す．さらに，視覚に障害を持つ中高生のために開発した点字ラベル付き 15 パズルとタブレット用 $mn-1$ パズルアプリのアクセシビリティについて解説する．

Loop Generators in a Sliding Puzzle and its Accessibility

Fujimoto Mitsushi[1,a)]

1. はじめに

2008 年夏，筆者らは全国から視覚障害を持つ中高生を募り，宿泊型のサイエンスキャンプ「科学へジャンプ・サマーキャンプ」[1] を企画・実施した．以来，この活動はサマーキャンプに留まらず，全国 8 地域での日帰り型の科学イベントを含む科学体験支援事業「科学へジャンプ」[2] へと発展し，現在も継続中である．このイベントでは，化学実験，動物骨触察，モーター工作，音楽プログラミングなど多くのワークショップが大学研究者・教育関係者・福祉機器企業により提供され，参加生徒同士の交流や社会で活躍する先輩による講演と共に大きな柱となっている．

筆者はこの事業において，ルービックキューブに関するワークショップを担当し，サマーキャンプや九州地区イベントでこれまでに 6 回実施している（[3]）．そこでは，紫外線硬化樹脂製の「・，＋，ー，□，○」の点字シールを貼った 2 × 2 のルービックキューブ（**図 1**）と，キューブの動きを点図で表現した資料を利用した．ルービックキューブは操作の種類が少なく（3 × 3 で 6 種，2 × 2 で 3 種），操作の周期性の確認が容易であり，数学とパズルの関係を説明する良い教材と言える．しかし，一度間違えると前の状態に戻すことが困難であったり，3 次元のルービックキューブの動きを配布資料において 2 次元の点図で表現したため，理解しづらいという課題があった．これはルービックキューブが 3 次元パズルであることが大きな要因と考えられる．そこで，2 次元パズルの教材化ができないかと考え，15 パズルに着目した．

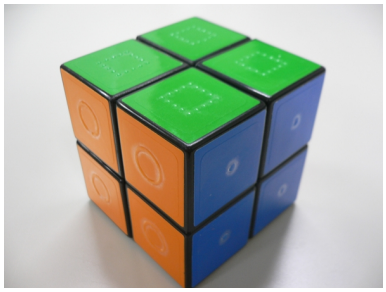

図 1 UV ラベルを用いたルービックキューブ
Fig. 1 Rubik's cube with UV labels

2. 15 パズルの回転型操作

15 パズルは 19 世紀から知られるスライドパズルの一種であり，ランダムに配置された 1 から 15 までの番号付きピースを正方形の台座の中でスライドして整列させるパズル（**図 2**）である．

[1] 福岡教育大学
University of Teacher Education Fukuoka, Munakata, Fukuoka 811-4192, Japan
[a)] fujimoto@fukuoka-edu.ac.jp

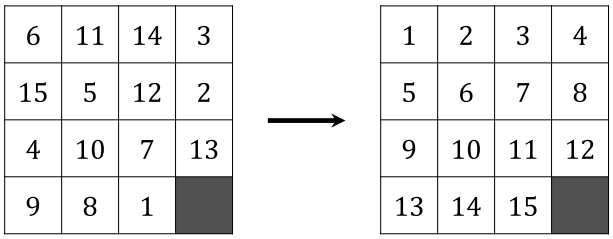

図 2 15 パズル
Fig. 2 15 puzzle

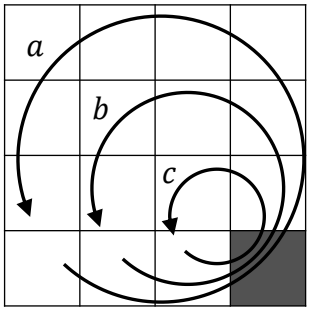

図 3 回転型操作
Fig. 3 Loop generators

数学者の間では,「15 パズルの操作は偶置換であり，操作の全体は 15 次交代群[*1]である」ことが良く知られている．交代群の生成元は何種類もあるが，教材化にはルービックキューブのような解法提示に適したわかりやすい生成元が必要となる．ここではある回転型操作を定義する．

定義 1 (Loop generators) *15* パズルにおける 3 つの巡回置換 (1 2 3 4 8 12 15 14 13 9 5), (6 7 8 12 15 14 10), (11 12 15) をそれぞれ a, b, c とおき，これらに対応する反時計回りの回転型操作（**図 3**）を *loop generators* と呼ぶ．

筆者は，この操作に対して次の結果を得た（[4]）．

定理 1 (15 パズル定理) P_{15} を *15* パズルの操作全体がなす群，a, b, c をその *loop generators* とすると，

$$P_{15} = \langle a, b, c \rangle$$

が成り立つ．

この結果は，任意の 15 パズルが loop generators のみで解けることを我々に教える．また，この定理は $m \geq n \geq 2, (m,n) \neq (2,2)$ を満たす m を列数，n を行数とする長方形の $mn-1$ パズルにも拡張できる．

3. 回転型操作による 15 パズルの解法

ここでは，前節で定義した loop generators を用いて 15 パズルを解くアルゴリズムについて解説する．これ以降，a^{-1}, b^{-1}, c^{-1} で a, b, c の逆操作（時計回りの回転型操作）を，$n > 0$ のとき a^n で a を n 回実行を，$n < 0$ のとき a^n で a^{-1} を n 回実行を表すこととする．15 パズルの解法の流れは次の通りである．

- **図 4** の A ブロック→ B ブロック→ C ブロックの順に完成させていく．
- 挿入するピースを回転で影響を受けない内側の位置に移動する．
- 挿入される側のブロックを回転し連結する．

図 5 の問題を例に 2 のピースを 3 のピースの左側に移動するための操作手順を示す．

(1) a^2（a を 2 回実行）．
(2) b^{-1}（b の逆を 1 回実行）．（**図 6**）
(3) a^3（a を 3 回実行）．（**図 7**）

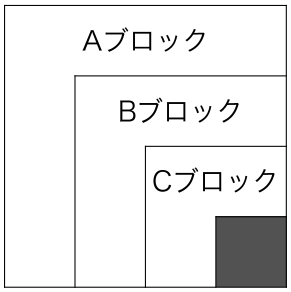

図 4 解法ブロック
Fig. 4 Blocks for solving

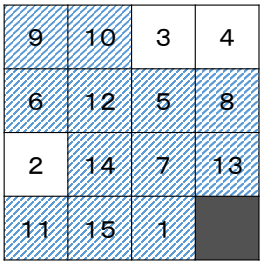

図 5 問題例
Fig. 5 Example

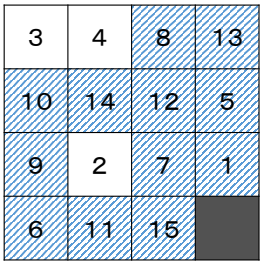

図 6 a^2b^{-1}
Fig. 6 a^2b^{-1}

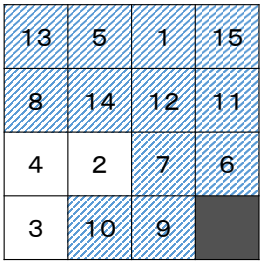

図 7 $a^2b^{-1}a^3$
Fig. 7 $a^2b^{-1}a^3$

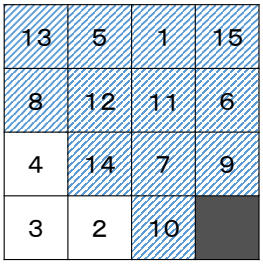

図 8 $a^2b^{-1}a^3b$
Fig. 8 $a^2b^{-1}a^3b$

[*1] 群とは演算が定義された集合のことで，その構造を研究する分野を群論と呼ぶ．

(4) b（b を 1 回実行）.（**図 8**）
(5) a^{-5}（a^{-1} を 5 回実行）.

つまり，操作 $a^2b^{-1}a^3ba^{-5}$ で $\boxed{2}$ のピースを $\boxed{3}$ のピースの左側に移動できる．このような操作を繰り返すことにより，A ブロック→ B ブロック→ C ブロックの順に全ブロックを揃えることができる．

4. 数式処理システムによる解法

群論用数式処理システム GAP [5] を用いて，loop generators による 15 パズルの解を求める．まず，次のように 15 パズル群を定義する．

```
gap> a:=(1,2,3,4,8,12,15,14,13,9,5);;
gap> b:=(6,7,8,12,15,14,10);;
gap> c:=(11,12,15);;
gap> P:=Group(a,b,c);
```

これ以降，GAP 上で 15 パズル群に関する計算が可能となる．以下では，15 パズル群の位数を求めている．15 次交代群の位数に一致していることが確認できる．

```
gap> Size(P);
653837184000
gap> Factorial(15)/2;
653837184000
```

問題 1 *次の 15 パズルを GAP を用いて解け.*

13	9	2	6
7	11	3	12
15	4	5	8
14	1	10	

この解を求める関数を GAP で書くと次のようになる．

```
gap> Solve15:=function(G,GenName,x)
gap>  local gen,F,hom;
gap>  F:=FreeGroup(GenName);
gap>  gen:=GeneratorsOfGroup(G);
gap>  hom:=GroupHomomorphismByImages(
            F,G,GeneratorsOfGroup(F),gen);
gap>  return PreImagesRepresentative(hom,x);
gap> end;
```

この関数 `Solve15` を用いて問題を解く．

```
gap> x:=PermList(
    [13,9,2,6,7,11,3,12,15,4,5,8,14,1,10]);;
gap> Solve15(P,["a","b","c"],x);
a*c^-1*a^-1*c^-1*a*c^-1*a*c^-1*a^-2*c^-1*b
*c^-1*a*c^-1*a^-1*b^-1*c^-1*b^2*a*b^-1*a^-1
*b^-1*a*b^-2*a^-1*b^3*a^2*b*a^-4*b*a^2*b*a^3
*b*a^-6*b*a*b^2*a^6*c^-1*a^4
```

この出力結果は整列された状態から問題の配置を実現するための操作手順である．よって，実際に 15 パズルを解く際は，出力結果の左から順に loop generators を逆向きに（つまり，a ならば a^{-1} を）実行すればよい．

5. Loop generators による手数の長さ

3 節で見たように loop generators による 15 パズルの解法は操作の種類が 3 種と少ないだけでなく，解法アルゴリズムの提示にも適している．しかし，通常の 1 ピース毎に移動する操作と比較して完成に至るまでの手数が著しく長ければ良い操作系とは言えない．そこで本節では，loop generators による 15 パズルの解法手数について述べる．

置換パズルに対して，問題を解くために必要となる手数の最小数は God's number と呼ばれる．3 × 3 のルービックキューブの God's number は FTM[*2] で 20，QTM[*3] で 26 であり（[6]，[7]），15 パズルの God's number は single-tile（連続するピースの移動もそれぞれカウント）で 80，multi-tile（連続するピースの移動は 1 手とカウント）で 43 であることが知られている（[8]，[9]）．Loop generators による 15 パズルの God's number は現在まだ知られていない．

置換パズルを解くことは，置換群における語の問題（Word Problem）を解くことと同値であり，計算群論（Computational Group Theory）からのアプローチにより最適解ではないが比較的短い解を求める Minkwitz アルゴリズム [12] と数式処理システム GAP 上の実装があり，これを前節では利用した．この実装を使用して，以下の 547 個のテストパターンに対する計算実験を行った．

- **ランダムパターン** (K_{100}): Korf [10] は計算実験のために 15 パズルのランダムなテストパターンを与えた．ここでは，これらの空白位置を右下に移動し，100 個のパターンを用意した．
- **最長手数パターン** (KN_{31}): Kociemba [11] は single-tile で 80 手かかる 17 個のパターンを与えた．また，Norskog [9] は multi-tile で 43 手かかる 16 個のパターンを与えた．ここでは，これらの空白位置を右下に移動し，重複を除いた 31 個のパターンを用意した．
- **魔方陣パターン** (MS_{416}): 空白を 16 と考えると，右下に空白を持つ 15 パズルで魔方陣になるものは 416 個存在する．これを数式処理システムを用いて求めた．

表 1 は loop generators による解法の実験結果であり，平均手数は 64 手であった．一方，**表 2** は [13] を用いた通常の操作（single-tile moves）による解法の実験結果（最適解）であり，平均手数は 56.7 手であった．以上の実験結果から，loop generators による解は通常操作より長くなる傾向があるが，Minkwitz アルゴリズムの出力が最適解でな

[*2] face-turn metric の略で，連続する 180 度の回転を 1 手とカウントする．
[*3] quarter-turn metric の略で，連続する 180 度の回転を 2 手とカウントする．

表 1　loop generators を用いた実験結果
Table 1　The computation result using loop generators

moves	30s	40s	50s	60s	70s	80s	90s–
K_{100}	2	12	33	25	17	7	4
KN_{31}	1	2	8	7	4	5	4
MS_{416}	2	29	116	164	63	33	9

表 2　通常操作での実験結果
Table 2　The computation result using single-tile moves

moves	30s	40s	50s	60s	70s	80s
K_{100}	0	23	62	15	0	0
KN_{31}	0	0	0	4	27	0
MS_{416}	0	1	359	56	0	0

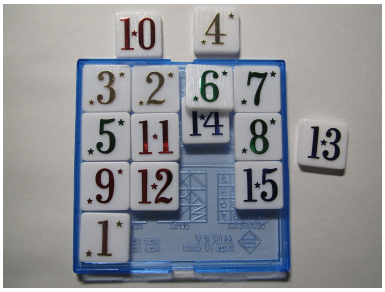

図 9　ピースが外れる 15 パズル
Fig. 9　Removable 15 puzzle

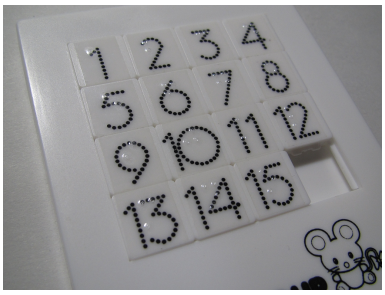

図 10　ピースが外れない 15 パズル
Fig. 10　Non-removable 15 puzzle

いことを考慮に入れれば，両者は同程度の操作系と言える．

6.　点字ラベル付き 15 パズルの試作

Loop generators による操作は単純で解法アルゴリズムの提示にも適している．視覚障害を持つ子ども達が使用できるよう配慮された 15 パズルがあれば，筆者がルービックキューブで行ったように科学イベントで利用可能な教材となり得るだろう．ここでは，筆者が試作した点字ラベル付き 15 パズルについて解説する．

通常の 15 パズルは **図 9** のように各ピースが台座に単に置かれているだけであり，操作中に簡単に台座から外れてしまう．ピースが外れないようにするには，スライドの際の力加減に注意して操作する必要があり，視覚障害者にとって取り扱いが難しい．つまり，ピースが台座から簡単

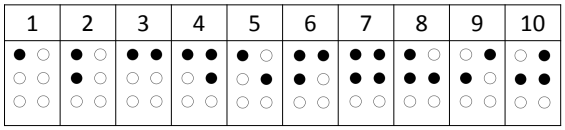

図 11　点字ラベル 1〜10
Fig. 11　Braille labels of 1 – 10

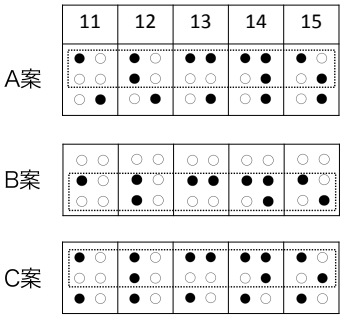

図 12　点字ラベル 11〜15
Fig. 12　Braille labels of 11 – 15

に外れないものが求められる．一方，**図 10** の 15 パズルは，ピースが台座から外れないよう配慮されたものである．ここでは，このタイプの 15 パズルを用いることにする．

次に，これに貼る点字ラベルについて述べる．筆者が入手できたピースが外れない 15 パズルは，ピースが 13mm 四方であり，貼り付け可能な点字ラベルは 1 枚である．[1] 〜 [10] のピースについては，**図 11** のように数符なしの 1 〜 9，0（ a 〜 j ）を用いた．[11] 〜 [15] のピースについては，次の 3 種のラベル（**図 12**）を作成し，4 名の視覚障害者に試用してもらった．

- **A 案**: 1 〜 5 の点字に 6 の点を追加
- **B 案**: 1 〜 5 の点字を下がり数字にしたもの
- **C 案**: k，l，m，n，o の点字を使用

その結果，4 名全員から「C 案が最もわかりやすい」という回答を得たため，C 案を採用した．

7.　タブレット用 $mn-1$ パズルアプリの作成

15 パズルにはランダムに配置されたピースを元の状態に戻す遊び方の他に，「与えられた配置が実現可能かどうか判定する」という遊び方がある．15 パズルにおいて実現可能な配置は偶置換であることが必要十分であるから，その配置が実現できるかどうかは数字の配列が偶置換か奇置換かを調べればよく，大学初学年で行われる線形代数の教材としても利用可能である．

前節で試作した 15 パズルは，物理的なピースの動きを自分の手で確認でき，制作費も安価であることから教材として適していると言える．しかし，これはピースが外れな

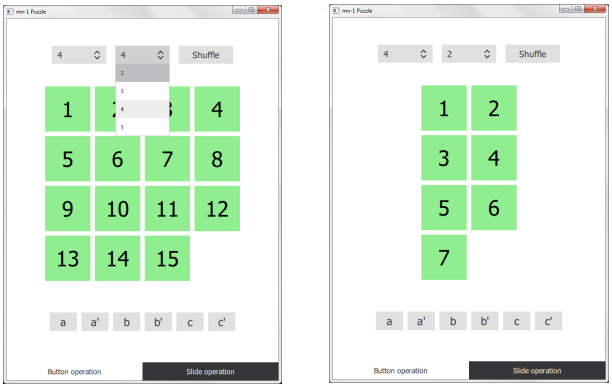

図 13 Loop generators モード
Fig. 13 Loop generators mode

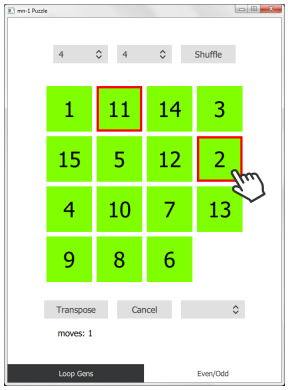

図 14 11 と 2 を選択して Transpose で交換
Fig. 14 Transposition of 11 and 2

いようになっているため，偶置換の配置しか作ることができず，「与えられた配置が実現可能かどうか判定する」という遊びができない．また，2 節の最後に述べたように，loop generators による解法は長方形の $mn-1$ パズルにも適用できるが，前節で試作した 15 パズルはサイズが固定されているので，$5\times4-1$ パズルや $2\times5-1$ パズルは試せない．これらの欠点を克服するために，マルチプラットフォームの $mn-1$ パズルアプリを開発した．ここではその概要について述べる．

7.1 開発環境 Qt について

Qt は 1994 年にノルウェーの Trolltech が開発したクロスプラットフォーム UI フレームワークである．Qt 自体は C++のクラスライブラリ群として提供されており，これを用いることで同一のコードで Windows / Linux / Mac OS X / QNX / VxWorks / Android / iOS などのソフトウェアを開発することが可能である（[14]）．2010 年にリリースされた Qt4.7 から新しい UI 開発技術 Qt Quick が導入され，C++に加えて QML(Qt Meta-object Language) という CSS によく似た UI 記述言語が使用できるようになった．これによって UI のロジックに JavaScript が使用可能となり，スマートフォンやタブレット用アプリに見られる動的な UI の作成が容易となった．また，Qt Quick にはアクセシビリティ機能も提供されている．ここで述べるタブレット用 $mn-1$ パズルアプリは，[15] 及び [16] で用いた Qt Quick による実装手法で開発を行った．

7.2 ダイナミック UI

本アプリにおいて，プレイヤーはまず $mn-1$ パズルのサイズを設定しなければならない．列数と行数はコンボボックスを利用して設定できるようにした．設定は直ちに反映され，$mn-1$ 個のピースが出現する．ピースの下には，loop generators に対応した loopgen ボタンが配置される．このボタンの個数もパズルサイズに連動して自動的に変更されるようになっている．

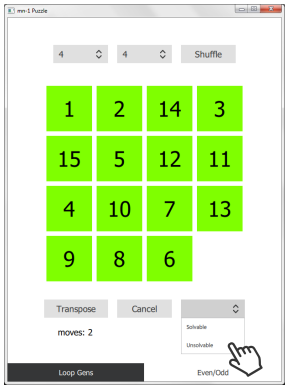

図 15 偶奇を選択
Fig. 15 Selection of even or odd

7.3 2 種類のプレイモード

本アプリのプレイ画面は 2 モードあり，最下部のタブで切り替えることが可能である．「Loop generators モード」は，ランダムに配置されたピースを画面下部の loopgen ボタンを使って移動させ，1 ～ 15 の順に整列させることを目指すモードである（**図 13**）．

もう一方の「偶奇判定モード」は，与えられた配置が実現可能かどうか判定するモードである．実現可能な配置かどうかは偶数回の互換で整列できるかを調べれば良い．このモードでは，2 個のピースをクリックで選択し Transpose ボタンを押すことでピースを交換できる（**図 14**）．プレイヤーはできる限り Transpose ボタンを利用しないようにして実現可能かどうかを見つけることを目指す．実現可能（偶置換）であることがわかった段階でコンボボックスから「Solvable」を，実現不可能（奇置換）であることがわかったら「Unsolvable」を選択する（**図 15**）．すると，正解なら「Correct!」不正解なら「Incorrect!」が表示される．

7.4 アクセシビリティ

本アプリには，Qt Quick のアクセシビリティ機能を用いて UI の各構成要素にスクリーンリーダーで読み上げ可能な文字列を埋め込んでいる．これにより，指でなぞるこ

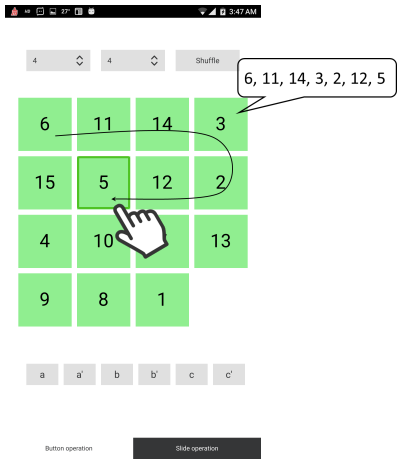

図 16 ピースの読み上げ機能
Fig. 16 Accessibility of UI

とでスクリーンリーダーがピースを読み上げ，パズル内のピース配置を把握できる（**図 16**）．指でなぞるとき，各ピースは移動しないように設定されており，画面下部の各種ボタンの実行はダブルタップで行う．また，操作が実行されたことを把握するために，ピースが移動した際には移動音を，パズル完成時には正解音を出すように配慮した．

8. おわりに

本稿で我々は 15 パズルにおける loop generators という回転型操作を導入した．この操作は単純であり，わかりやすい解法提示が可能であるだけでなく，約 500 個のテストパターンに対して，解に至るまでの手数が通常操作と同程度の平均 64 手であることを計算実験により示した．このことから loop generators を用いた 15 パズルは，中高生向け科学イベントでの教材になり得ると考えられる．そして，本稿では視覚障害を持つ中高生が利用できるように，点字ラベルを貼った 15 パズルと，サイズ変更や配置の偶奇判定が可能なタブレット用 $mn-1$ パズルアプリを作成した．

次のステップとして，この 2 種類のパズルを視覚障害を持つ中高生向け科学イベントで実際に利用し，有効性を検証することが今後の課題である．また，$mn-1$ パズルアプリについては，数式処理システムと連携して解を求め，解法を助言・提示するアプリへ発展させることも重要と考える．

謝辞 本研究は JSPS 科研費 JP17K01133 の助成を受けたものである．

参考文献

[1] 科学へジャンプ・サマーキャンプ 2008, http://www.sciaccess.net/JSSC2008/
[2] 科学へジャンプ, http://www.jump2science.org/
[3] 藤本光史: さわって解けるルービックキューブ, 視覚障がいをもつ生徒たちのための科学へジャンプ・サマーキャンプ 2008 報告, 31–35, 2009, http://www.sciaccess.net/JSSC2008/JSSC2008Report.pdf
[4] M. Fujimoto: Loop Generators in the $mn-1$ Puzzle and Factorization Problem, *Proceedings of the Workshop on Mathematical User-Interfaces 2017*, to appear.
[5] The GAP Group: GAP – Groups, Algorithms, Programming – a System for Computational Discrete Algebra, http://www.gap-system.org/
[6] T. Rokicki, H. Kociemba, M. Davidson, and J. Dethridge: The diameter of the Rubik's cube group is twenty, *SIAM J. Discrete Math.*, 27, 1082–1105, 2013.
[7] T. Rokicki and M. Davidson: God's Number is 26 in the Quarter-Turn Metric, http://cube20.org/qtm/
[8] A. Brüngger, A. Marzetta, K. Fukuda and J. Nievergelt: The parallel search bench ZRAM and its applications, *Annals of Operations Research*, 90, 45–63, 1999.
[9] B. Norskog: The Fifteen Puzzle can be solved in 43 moves, http://cubezzz.duckdns.org/drupal/?q=node/view/223
[10] R. E. Korf: Depth-first iterative-deepening: An optimal admissible tree search, *Artificial Intelligence*, 27, 97–109, 1985.
[11] H. Kociemba: 15-Puzzle Optimal Solver, http://kociemba.org/fifteen/fifteensolver.html
[12] T. Minkwitz: An Algorithm for Solving the Factorization Problem in Permutation Groups, *Journal of Symbolic Computation*, 26, 89–95, 1998.
[13] B. Borowski: Optimal 8/15-Puzzle Solver, http://brian-borowski.com/software/puzzle/
[14] The Qt Company, http://www.qt.io/
[15] M. Fujimoto: An implementation method of a CAS with a handwriting interface on tablet devices, *Proceedings of the 4th International Congress on Mathematical Software*, Lecture Notes in Computer Science vol.8592, Springer, 545–548, 2014.
[16] 藤本光史: タブレット端末への数式処理システムの実装手法, 京都大学数理解析研究所講究録 1927, 89–102, 2014.