6L-9

任天堂テンビリオンの操作手順抽出アルゴリズムの設計と実装

竹内悟 †1　西田誠幸 †2　原田紀夫 †3
拓殖大学工学部情報工学科 ††

1. はじめに

任天堂テンビリオン (以下テンビリオン) は，23個のボールを決められた形に整列するパズルゲームである．プレイヤは，このテンビリオンを解く際に，特定のボールのみを移動させる一定の操作手順を用いる．しかし，一般に知られている一定の操作手順は，その種類が少なく，テンビリオンを効率良く解くことができない．

そこで本稿では，操作手順におけるプレイヤの扱いやすさについて考察し，その考察をもとに一定の操作手順を定義する．そしてさらに，その定義に基づき，多くの一定の操作手順を抽出することを目的とする．また本稿では，プレイヤが扱いやすいことを前提に考えられている操作手順を，有効操作手順と呼ぶことにする．

2. テンビリオンについて

テンビリオンは，透明の樽型の側面に配置された5行×5列に，6色 (赤・黄・橙・緑・青・黒)・23個のボールを整列するパズルゲームである．図1で割り振られている行2から行5の中央4行に，黒を除く5色のボールを列ごとに揃え，列A・列C・列Dの行1に黒のボールを揃える．この状態が完成状態である．

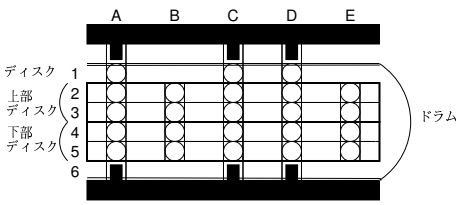

図 1: テンビリオン

テンビリオンには6つの操作が存在する (表1)．また，同操作種類の操作は，それぞれが対称操作となっていて，連続して行うと，テンビリオンの状態は操作前の状態に戻る．具体例として，左回転 U^- の後に右回転 U^+ を行うと，テンビリオンの状態は操作前の状態に戻る．

表 1: テンビリオンの操作

操作種類	操作 (対称操作)	
ドラム操作	上運動:D^+	下運動:D^-
上部ディスク操作	右回転:U^+	左回転:U^-
下部ディスク操作	右回転:L^+	左回転:L^-

3. 操作手順

既存の有効操作手順 [1][2] は，移動するボールの数が最大で7個と少ないことや，また，操作手数が最大で14手と少ないことで，プレイヤが操作手順を把握しやすい．しかし一方で，移動するボールの位置が離れており，テンビリオンの円筒形の形状のために，プレイヤの目に捉えづらいという点や，既存の有効操作手順はその種類が少なく，プレイヤが，テンビリオンの状態や扱いやすさの判断に応じて，有効操作手順を選ぶことができない点が欠点として挙げられる．

本稿では，これら既存の操作手順の利点・欠点を考慮し，以下に示すボールの移動を行う操作手順を，有効操作手順として定義する．

(1) 行1を除く単一行で任意のボールが移動する

(2) 行1を除く単一列で任意のボールが移動する

(3) 隣り合う3列間で，2個以上4個以下のボールが置き換わる

(4) 任意の行・列で，2個のボールが入れ換わる

(5) 行1を除く単一列で任意のボールが移動し，他の行・列で2個か3個のボールが移動する

(6) 行5と行4の任意のボール1個ずつが入れ換わる

また，これらの有効操作手順の操作手数の上限値は，プレイヤが記憶できる程度の長さにおさえなければならない．このことを踏まえ本稿では，上限値を20手として，20手以下の有効操作手順を対象とする．

An Algorithm of Extracting Useful Sequences of Operations for Solving Ten-Billion Barrel Puzzle

†1Satoru TAKEUCHI (r18443@st.takushoku-u.ac.jp)
†2Seikoh NISHITA (snishita@cs.takushoku-u.ac.jp)
†3Norio HARADA (nharada@cs.takushoku-u.ac.jp)
††Dept. of Computer Science, Faculty of Engineering, Takushoku University

4. アルゴリズム設計

4.1 アルゴリズム設計概要

本稿では，前節で定義した有効操作手順を抽出し，文字列として出力するアルゴリズムを設計する．

このアルゴリズムでは，テンビリオンの 6 つの操作を枝とし，操作手数を深さとする 6 分木により，テンビリオンの操作の取り得る全ての組み合わせを表現する．この 6 分木に対して，深さ優先探索により探索を行っていく．そして，テンビリオンの完成状態を初期状態とし，探索された操作を行っていき，変化したテンビリオンの状態が，3 節で定義したボールの状態と一致した場合に，この操作手順を有効操作手順として出力する．しかし，操作手数の上限値を 20 手として，全ての操作の組合せに対して探索する手間は，概算で 3.6×10^{15} となり，大きな手間がかかる．そこで，探索の必要がない操作手順を調べて，枝刈りを行うようにアルゴリズムを改良することで効率を高める．

4.2 枝刈り

4.2.1 操作手順の簡単化による枝刈り

テンビリオンの操作手順は，その意味に基づいて簡単化できる．その具体例として，$U^-U^+U^+$ は U^+ と簡単化でき，$U^+U^+U^+$ は U^-U^- と簡単化できる．本アルゴリズムでは，このように簡単化可能な操作手順に対して枝刈りを行うことにする．その理由は，本アルゴリズムは簡単化可能な操作手順 T とともに，簡単化された T' についても探索を行うのだが，この T と T' を比較すると操作手順後のボールの状態が同様となり，T の方が操作手数が長いため，T で始まる操作手順が有効操作手順となることはないためである．以上の方法を用いて，2 節で述べたテンビリオンの特徴により本アルゴリズムを改良し，枝刈りを行った．その結果，深さを 20 としたときの計算量は，概算で 3.6×10^{15} から 4.8×10^{10} に削減された．

4.2.2 アルゴリズムを利用する枝刈り

前節で述べた方法により本アルゴリズムに改良した後に，さらに，本アルゴリズムを利用し，テンビリオンの状態が元の状態に戻る一定の操作手順 T を抽出した．この T の具体例として，$D^+U^-D^-L^-D^+U^+D^-L^+$ が挙げられる．この T によって，テンビリオンの状態は T を行う前の状態に戻る．よって前節と同様に，T に対して枝刈りを行うことにする．以上の方法と，テンビリオンの状態に基づく枝刈りとを組み合わせて，アルゴリズムを改良し，枝刈りを行った．その結果，深さを 20 としたときの計算量は，概算で 4.8×10^{10} から 6.9×10^{9} に削減された．

5. アルゴリズムの実装結果

本アルゴリズムを Java により実装した．その結果として，抽出した有効操作手順を定義ごとに分け，その種類の数を表 2 に示す．また，複数の定義に該当する有効操作手順は 2 重にカウントした．

表 2: 各定義に該当する有効操作手順の種類の数

	(1)	(2)	(3)	(4)	(5)	(6)	全体
数	32	0	263	0	104	20	337

本アルゴリズムにより，多くの有効操作手順を抽出することができた．これら抽出された多くの有効操作手順を用いることで，効率良くテンビリオンを解くことができるようになると考えられる．しかし，抽出した有効操作手順は，その種類が多いため，プレイヤの判断では，これらの有効操作手順を，効果的に用いることができない恐れがある．そのため，テンビリオンを効率良く解くために，有効操作手順の組み合わせについて考える必要がある．

6. おわりに

本稿では，テンビリオンの操作手順におけるプレイヤの扱いやすさについての考察を行った．そしてその考察を受け，自らにより有効操作手順を定義した．さらには，その定義に基づき有効操作手順を抽出するためのアルゴリズムを設計し実装した．その結果として，多くの有効操作手順を抽出することができた．

今後の課題として，テンビリオンを効率良く解くために，有効操作手順の組み合わせについて考える必要がある

参考文献

[1] TumblerPuzzle
http://www.smartbytes.co.uk/Tumbler/

[2] テンビリオンの解法パターン (7 進法)
http://www.geocities.co.jp/Hollywood/9743/puzzle/