4S-01

ルービックキューブの部分パターンを用いた評価関数の構成

草間 正喜 †　　山本 修身 ‡
名城大学 †　　名城大学 ‡

1　はじめに

15 パズルなどのスライディングパズルをコンピュータで効率的に解く場合，IDA*アルゴリズムなどのヒューリスティックサーチアルゴリズムが利用することができる．その際，評価関数の性質によって効率が決まる．ルービックキューブは一度の動作で複数のブロックが移動するため 15 パズルと比べ評価関数の構成は困難である．本稿では，[1] と同様の方法であるランダムな試行で得られた盤面の状態をニューラルネットワーク (NN) で学習することでルービックキューブの評価関数の構成をした．

ルービックキューブは一般的に $3\times3\times3$ の立方体でブロックの各面は 6 色のいずれかとなっており，1 つの面を回転させることによってブロックの位置を変化させることができる．本稿で用いる回転操作は 1 つの面に対して 90°，180°，270° の回転がある．さらに 6 面全て同様の操作ができるため計 18 通り操作できる．操作を繰り返しブロックをゴールとなる状態にすることがこのパズルの目的である．図 1 にルービックキューブの外観を示し，本稿では左図をゴール状態とする．前述の 18 通りの操作によって，どの状態からでも最大 20 回転でゴール状態にすることが可能であることが 2010 年に Rokicki らによって証明されている [2].

また，本稿では膨大なルービックキューブの状態空間を削減するために学習データを作成する際に状態をいくつかの部分パターンに分解することによってより広い範囲で精度の高い NN の構成を目指した．

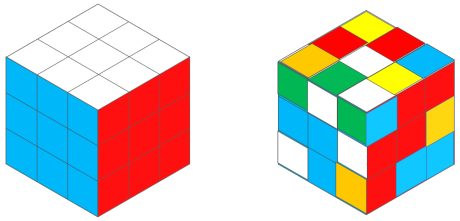

図 1　左図はゴール状態，右図はある状態

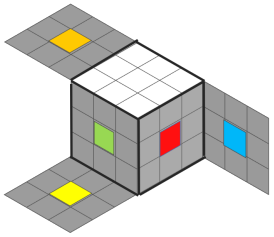

図 2　学習データ作成に用いた簡易な状態

2　構成した学習データと NN

本稿では，ルービックキューブを解くためのアルゴリズムとして IDA*アルゴリズム [3] を使用している．ルービックキューブはゴール状態から何回の操作でそれぞれの状態に遷移したかを記録することによって，与えられた状態からの操作回数を得ることができるため，これを NN に学習させるデータとして用いる．なるべく少ない操作回数を得るために，既出の状態が出現した場合回転数が少ない値を学習データに用いる．ルービックキューブの状態数は約 4.3×10^{19} 個と大きいため，すべての状態を一様に覆うような学習が困難である．これを改善するため図 2 のような縮退したルービックキューブを考える．この縮退した状態をランダムに回転させることで得られたデータを用いて NN を構成した．

前述の操作によって得られるルービックキューブの状態を入力，ある状態に至るまでの回転数を出力として NN に学習させる．NN は，入力層，中間層 5 層，出力層の計 7 層で構成した．本稿で使用した縮退した状態は 2 色であり色は 2 ビットで表して

Construction of evaluation function using partial pattern of Rubik's cube
† Masayoshi Kusama, Grad. Sch. of Sci. and Tech, Meijo Univ
‡ Osami Yamamoto, Fac of Sci. and Tech, Meijo Univ

いる．このため入力は中心を除く 48 か所を 96 ビット，出力層は softmax 関数を用いて 25 以下の整数値を出力している．中間層は，各 500 ノードで構成され，各層は ReLU 関数を用いて全結合している．またデータを学習させる回数を 30 回とした．

3 実行結果と考察

Python で学習モデルを構成し, JavaScript で IDA* と学習モデルによる出力値の計算を行った[*1]．学習データは図 2 の状態からランダムに 25 回転させる試行を 150,000 回繰り返し得られた状態を用いた．また，評価関数には縮退した状態を学習させた NN を 6 面全ての評価値を求め足し合わせ 6 で割った値を使用した．比較に用いる評価関数は元の状態 (図 1 の左図の状態) を学習させ作成した評価関数を用いた．最小回転数が 2〜10 回転となる状態を IDA* に用いてゴールの状態までの回転手順を求めた．

表 1　探索した状態数と各評価関数の正解率

最小回転数	状態数	正解率 (%)	
		degenerate	original
2	1	100	100
3	2	100	100
4	1	100	100
5	7	86	100
6	7	57	100
7	8	100	100
8	13	84	54
9	10	80	40
10	4	50	50

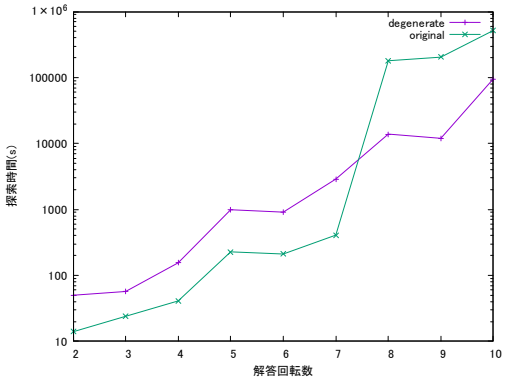

図 3　解答回転数による探索時間の変化

縮退した状態を学習した評価関数 (degenerate) と元の状態を学習したは評価関数 (original) によるアルゴリズムの正解率はそれぞれ 82% と 67% であった．図 3 にそれぞれのアルゴリズムの探索時間を示す．

図 3 より，解答が 7 回転以下の状態は，元の状態を学習させた NN は学習データが正確であるため評価値も正しい値を出力する．一方で縮退した状態の NN では，6 面の和を求めた際に誤差が生じるため，7 回転以下の解答では元の状態を学習させた方が良い結果である．しかし，8 回転以降の状態では元の状態の学習データは，ある状態の回転数が最小より大きくなる傾向 であり正しい回転数が得ることができていない状態が多くなるため評価関数の性能が悪くなる．一方で縮退した状態を学習させた場合同様の状態の出現頻度が高いため回転数が正しい状態になりやすく元の状態を学習させるよりも性能が悪化しないため，縮退した状態の NN の方が良い結果である．

4 今後の課題

本稿では評価値を縮退した状態を学習した NN を 6 面に適応させることで求めた．しかし，縮退した状態はいくつか考えることができ，より良い評価関数ができる可能性がある．より良い評価関数の構成が今後の課題である．

参考文献

[1] 山本修身，伊藤康太: ランダムな試行の学習による 15 パズルの評価関数の構成．電気学会論文誌 C (電子・情報・システム部門) Vol. 139，No. 12，pp. 1420–1428 (2019)

[2] Rokicki, T., Kociemba, H., Davidson, M., Dethridge, J.: The diameter of the Rubik's cube group is twenty, *SIAM Rev. 56*, pp. 645–670 (2014)

[3] Korf, R. E.: Finding optimal solutions to Rubik's cube using pattern databases, *Proceedings of the Fourteenth National Conference on Artifcial Intelligence and Ninth Conference on Innovative Applications of Artifcial Intelligence*, pp. 700–705 (1997)

[*1] 計算環境を以下に示す．OS Windows 10, CPU: Intel Core i7 8565U, メモリ:8GByte, 学習データ作成, IDA*: Node.js 12.18.3(JavaScript)，学習:Python 3.8.5 上の TensorFlow 2.3.0