6M-08

15パズルのための評価関数のランダムな学習による構成

伊藤康太 †　　山本修身 ‡

名城大学理工学研究科情報工学専攻 †　　名城大学理工学部情報工学科 ‡

1　はじめに

本稿では，15 パズルを IDA*アルゴリズムなどのヒューリスティックサーチ (HSA) で解くための効率的な評価関数をゴール状態からの逆回しの試行による学習により構成し，許容的な評価関数であるマンハッタン距離に基づく評価関数とその性能を比較した．

ここで扱う 15 パズルはスライディングブロックパズル (SBP) の一種である．他に SBP として 24 パズル，箱入り娘パズルなどがあり有名なルービックキューブも SBP に含めることができる．これらのパズルは 理論的にはサイズに関して PSPACE 完全であり [1] 一般的には解くのは困難であるが，サイズの小さなものに限れば計算機で適当な評価関数を利用して HSA で解くことができる．その際，効率的にパズルを解くにはゴール状態までのステップ数を見積もる高性能な評価関数が必要となる．これまでそれぞれのパズルについて，個々のパズルの特徴を利用して色々な評価関数が作られてきた [2][3]．本稿では実際に計算機が SBP のコマを動かすという動作を通じてその「経験」を蓄えることにより評価関数を自動生成することを試みた．ここで述べた評価関数は現在のところそれほど性能の高いものではないが，もし効率的な構成方法が見つかれば比較的小さな変更でこの手法を他の SBP に適用することも可能であると考えられる．

本稿で示す手法は実際に 15 パズルの状態を変化させた結果を蓄えることにより評価関数を構成することから評価関数は必ずしも許容的とならない．したがって必ずしも最適解を与えるものではないが，後述するように多くの場合 IDA*によって最適解が得られている．

2　15 パズルについて

15 パズルは 4×4 のコマの入る枠とその中に置かれる 1 から 15 までの数が書かれたコマから構成される．コマの置かれていない一箇所を利用してコマをスライドさせることによって状態を変化させることができる．この操作を繰り返して，与えられた配置からゴール状態へ変化させることがこのパズルの目的である．ここでは図 1 左図で示す状態をゴール状態とする．

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

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

図 1　15 パズルの 2 つの状態．左はゴール状態であり，右はランダムにスライドさせたある一つの状態である．

15 パズルを解く場合，適当な評価関数により A*や IDA*[4] などの HSA で解を求めることができる．15 パズルについては良く知られたマンハッタン距離の和による評価関数がある．コマ k の位置が (i,j) である場合，ゴール状態でのコマ k の位置が $(k \bmod 4, \lfloor k/4 \rfloor)$ であることからマンハッタン距離を $d(k) = |k \bmod 4 - i| + |\lfloor k/4 \rfloor - j|$ と定義する．これを用いてマンハッタン距離による評価関数は $\mathrm{MD}(p) = \sum_{i=1}^{15} d(i)$ と定義される．ただし，p はコマの配置とする．ゴール状態 g の評価関数値は $\mathrm{MD}(g) = 0$ となる．また配置 p から p' へ 1 回のスライドで変化させたとき，$|\mathrm{MD}(p) - \mathrm{MD}(p')| = 1$ であることから，任意の状態 p からゴール状態まで変化させるのに少なくとも $\mathrm{MD}(p)$ ステップ必要である．このような性質をもつ評価関数を許容的と呼び，A*や IDA*により解を探索するときその解が最適解であることが保証される．

3　ランダムな学習による評価関数の構成

15 パズルのような SBP では，ゴール状態からランダムに状態変化させたコマの動作の系列を逆回しにすることによって，ある状態からゴール状態に至る動きを求めることができる．この動きが最適な動きであるとは限らないが，多くの試行の中で，それぞれの動きに現れる状態とゴール状態からその状態に至るまでのスライドの回数 (ステップ数) を評価関数値として状態とともにデータベース（DB）に書き込み，後でそれを利用する．ただし，DB とはパターンをキーとし，ステップ数が出力されるものである．たとえば，p_0 をゴール状態としたとき，それが DB に存在すれば $\mathrm{DB}(p_0) = 0$ が出力される．

p_0, n_0 のゴール状態のパターンとステップ数 0 から始め，N 回ランダムに動かし DB に記録していく．さらにその試行を M 回行う（ここでは $N = 200, M = 10^8$ とする）．また，何回かの試行で同じ状態が現れた場合，そのうちの最小値で更新する．このアルゴリズムを Algorithm 1 に示す．ただし，5 行目の $move(p_j)$ は状態 p_j からランダムに 1 手動かすことを表している．

また，15 パズルのすべてのパターンをキーとして DB を構成すると，その全てのパターンは，$p_i \in S_{16}$ ($|S_{16}| \fallingdotseq 4\times10^{12}$) となり，DB のサイズが爆発してしまう．そこで，パターンの一部を取り出すことによりパターンを圧縮し，それをキーとして DB を構成する方法を用いる．本稿では，図 2 のように，空白と下の段の 6 つのコマの位置を取り出しそれをキーとし記録する．これは人間が下の段から揃えていく解き方を参考にしている．この圧

Construction of evaluation functions for the fifteen puzzle using random examples of moves

† Kota Ito, Division of information Engineering, Graduate School of Science and Technorogy, Meijo University

‡ Osami Yamamoto, Department of Information Engineering, Faculty of Science and Technology, Meijo University

Algorithm 1 DB の学習アルゴリズム

```
function makeDB(N, M)
    for (i = 0; i < M; i++) do
        n_0 = 0, p_0 = p_finalState
        for (j = 0; j < N; j++) do
            p_{j+1} = move(p_j)
            n_{j+1} = n_j + 1
            if p_{j+1} ∈ DB(p_{j+1}) and n_{j+1} < n_{DB(p_{j+1})} then
                n_{DB(p_{j+1})} = n_{j+1}
            else if p_{j+1} ∈ DB(p_{j+1}) and n_{DB(p_{j+1})} < n_{j+1} then
                n_{j+1} = n_{DB(p_{j+1})}
            else if p_{j+1} ∉ DB(p_{j+1}) then
                DB(p_{j+1}) に p_{j+1} と n_{j+1} の組を追加
            end if
        end for
    end for
    return DB
end function
```

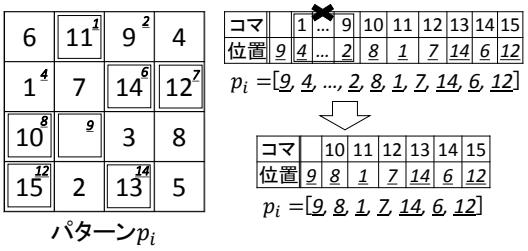

図 2　空きスペースと 10〜15 のコマを取り出した圧縮.

縮方法の場合，すべてのパターンは約 5.8×10^7 パターン存在し，$|S_{16}|$ と比べて約 $1/10^5$ に圧縮できる．また，このようなパターンを簡略化するような圧縮をしたとき，DB にステップ数 s を記録するより $|s - \mathrm{MD}(p)|$ を記録した方がよりよい評価関数を与える DB となることがわかっているのでそれを利用する [5]．その理由として，まずパターンの圧縮をすることで，さまざまなパターンが同一のものとして扱われ，大きなステップ数がかかるパターンに対しても，より小さいステップ数が書き込まれる．しかし，ステップ数とマンハッタン距離の和には，ステップ数が大きくなればマンハッタン距離の和も大きくなるような相関があると考えられ，その差を取ることで大きな値が得られる．

4　計算機実験

実際に DB を構成し，DB を使用する際に評価関数として，$V(p) = \mathrm{MD}(p) + \mathrm{DB}(p)$ を用いる．ただし，一度も出現していないパターン p については $\mathrm{DB}(p) = 0$ とする．DB の構成には約 1 時間かかった．そして，IDA* に DB を使い構成した評価関数と，マンハッタン距離の和を用いた評価関数を使うことで，ランダムに作成した 500 個のパターンを解いたときの探索ノード数と計算時間の計算結果を図 3 に示す[1]．その結果全体として，探索ノード数は 85%，計算時間は 76%削減できた．また，最適解が得られた割合は 90%であった．

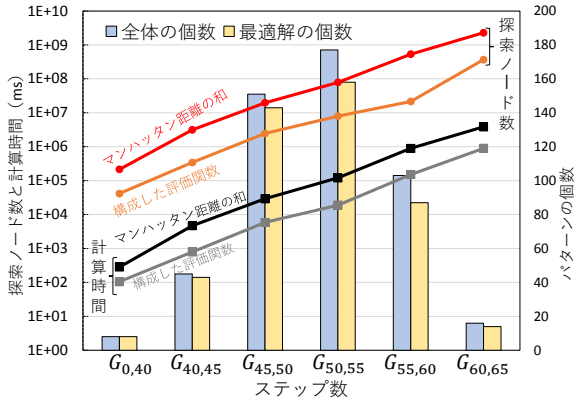

図 3　評価関数別の平均探索ノード数と平均計算時間（$G_{i,j}$ は最適解 d が $i < d \leq j$ の範囲にあるパターンの集まり）.

5　まとめと今後の課題

本稿では，SBP の中で比較的解きやすい 15 パズルを用いて，A*や IDA*で解くための評価関数を構成した．この評価関数は許容的ではなく，ゴール状態からの逆回しの試行による学習によって，パターンとステップ数からなる DB で構成されている．この際 DB のサイズを考慮し，空白と 6 つのコマの位置を取り出したパターンの圧縮によって，DB のサイズを小さくした．

さらに探索ノード数や計算時間を短くする方法として，パターンの圧縮方法を変えることや圧縮率を下げること，N と M を変化させ学習させることなどが考えられる．また，ニューラルネットワークでパターンからステップ数を得るようなものも考えている．このような評価関数がうまく構成できれば，今後 15 パズルだけではなく，同様のパズルについて，パズルを動かして状態とステップ数を記録するという行為を通して，パズルの解き方を学習することに対応する評価関数を構成することが可能になると考えられる．

参考文献

[1] Robert A. Hearn, Erik D. Demaine: ゲームとパズルの計算量. 上原隆平 訳, 近代科学社 (2011)

[2] 山本修身, 佐藤根寛: ギャップ集合を用いた 15 パズルの最適解探索の高速化. 人工知能学会論文誌, Vol. 26, No. 2, pp. 419-426 (2011)

[3] 山本修身, 加藤貴之: 箱入り娘型スライディングブロックパズルのためのパターンデータベースの構築とその性能評価. 電気学会論文誌, Vol. 137, No. 9, pp. 1-10 (2017)

[4] Korf, R. E.: Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, Vol. 27, No. 1, pp. 97-109 (1985)

[5] 伊藤康太, 山本修身: 15 パズルのための評価関数の「学習」による構成について. 第 15 回情報学ワークショップ WiNF, A-6 (2017)

[1] ここで示す結果は以下の計算環境で行った．CPU：Core i7-6700K，メモリ：16GByte，OS：Windows 10，DB 作成：Node.js 7.10.0（JavaScript），IDA*アルゴリズム：Visual C++ 2017 14.10（C++）