3S-1

BDDを用いた15パズルのギャップ集合の効率的な表現について

長谷川 冴香† 山本 修身‡

名城大学大学院 理工学研究科 † 名城大学 理工学部 情報工学科‡

1 はじめに

本稿は，スライディングパズルの一種である15パズルを対象としている．15パズルは，4×4 に仕切られた盤面と15個の駒によって構成されているここでは図1の盤面を最終状態とし，この状態までスライドさせる経路を効率的に探索する．ここで対象としているのは最少手順数で求めた解，つまりは最適解である．

2 ギャップ集合を利用した15パズルの最適解の探索

ある駒の配置の状態を c とし，c における i 番目の駒の配置のベクトル値を $P(c,i)$ としたとき，マンハッタン距離関数 $M(c) = \sum_{i=1}^{15} ||P(c,i) - P(g,i)||_1$ を定義する．ただし，g は最終状態，$||\cdot||_1$ は，2次元ベクトルの縦横の絶対値の和を表す．この関数は許容的な評価関数となる．ここで，15パズルの最適解はIDA*アルゴリズムによって効率的に解けることが分かっているため [3]，本稿ではIDA*アルゴリズムを利用している．

状態 c_1 から c_2 への1回の駒のスライドによる状態遷移のうち，$M(c_2) = M(c_1) - 1$ となっているものをマンハッタン遷移と呼ぶ．また，ある配置からマンハッタン遷移のみでゴール状態に到達できるとき，その配置をギャップゼロ配置と呼ぶ．

全ての15パズルの配置の中で，ギャップゼロ配置となっている配置の集合をゼロ集合 G_0 と呼び，$G_0 = \{c \in C | \exists (c = c_1, c_2, \ldots, c_N = g)$ かつ $c_i \rightarrow c_{i+1}$ はマンハッタン遷移である．$\}$ とかける [5]．任意の配置 c が与えられたとき，ゼロ集合 G_0 に c が含まれるかどうかがわかれば，以下のように評価関数を改良できる:

$$h_G(c) = \begin{cases} M(c) & (c \in G_0) \\ M(c) + 2 & (c \notin G_0) \end{cases} \quad (1)$$

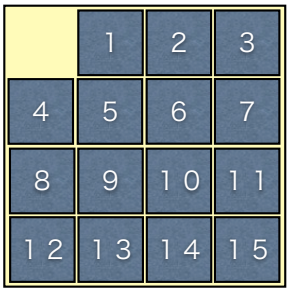

図 1: 15パズルの最終状態

これをゼロ集合評価関数とする．ここで，ゼロ集合 G_0 に c が含まれない場合に＋2されているのは，15パズルの性質 [5] により，真の最少手順数はマンハッタン距離に0を含む偶数回足された数となるためである．

さらに，最少手順数とマンハッタン距離との差がゼロとなる配置の集合であるゼロ集合に対して，2以上の偶数となる場合の配置の集合をギャップ2N集合と呼ぶ．この集合を利用した評価関数をギャップ2N集合評価関数とよび, 以下のように定義する:

$$h_{G:2N}(c) = M(c) + \begin{cases} 0 & (c \in G_0) \\ 2 & (c \notin G_0, c \in G_2) \\ 4 & (c \notin G_2, c \in G_4) \\ \ldots & \ldots \\ 2N & (c \notin G_{2N-2}, c \in G_{2N}) \\ 2N+2 & (c \notin G_{2N}) \end{cases} \quad (2)$$

しかし，ゼロ集合のサイズが膨大のため，これらの手法では集合を近似してハッシュ法を用いているにすぎない．

3 BDDを用いたギャップ集合表現の効率化

BDDとは，論理式を表現するデータ構造であり [1][2]，BDDを用いることでギャップ集合をコンパクトに表現することができる．つまり，BDDは集合の圧縮表現とみなすことができ，さらにその圧縮表現のまま大規模な論理演算をおこなうことが可能となる．この性質を

*On an efficient method for representing the gap sets of the fifteen puzzle using BDD

†Saeka Hasegawa, Graduate School of Science and Technology, Meijo University

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

利用し，15パズルの最適解の探索での計算量の削減を試みた．本稿ではさらに近似された集合をBDDを用いることでより効率的な表現とする手法を提案した．

4 計算機実験

深さ優先探索によって生成したギャップ集合をBDDで表すために，盤面をエンコーディングした．15パズルの駒それぞれを4bitで表現することで1つの盤面を64bitで表した(空欄は0)．その64bitのうち8駒32bit，10駒40bit，12駒48bitの情報量のみを考慮した集合をそれぞれ作成した．ゼロ集合を近似した集合はそれぞれ，$Gap0_8$，$Gap0_{10}$，$Gap0_{12}$とする．また，ギャップ2集合も同様に$Gap2_8$，$Gap2_{10}$とする．$Gap2_{12}$については，メモリが足りなくなり今回作成することができなかった．作成したBDDのファイルサイズ，実際のノード数，BDDが表している実際の盤面のパターン数を表1に示す．BDDの作成にはJava言語とPythonを用いた．

表1のBDDを利用した評価関数を用いて，ランダムに生成した300パターンの15パズルの問題において最適解の探索をおこなった．表2は，その実験結果であり，この300パターンの結果の平均値である．計算は全てC言語でおこない，GCC version 4.2.1を用いた．計算時間は全て同一の計算機上*の実行結果である．M(c)を用いた場合と比べて，ゼロ集合を用いた場合は探索ノード数を約0.26，実行時間を約0.45にまで，2集合を用いた場合はノード数を0.06，実行時間を約0.19にまで削減することができた．

表1: 作成したBDD

BDD	file size(byte)	ノード数	パターン数
$Gap0_8$	7,870,996	1,489,108	4,233,704
$Gap0_{10}$	54,289,825	9,965,935	23,696,364
$Gap0_{12}$	132,693,298	24,643,389	61,559,068
Gap0	120,098,272	24,867,599	88,728,780
$Gap2_8$	49,439,511	8,514,680	32,844,091
$Gap2_{10}$	527,378,793	87,720,568	256,541,479

表2: 評価関数別の探索ノード数と実行時間．

評価関数 (BDD)	探索ノード数	実行時間 (s)
M(c) (-)	1.32×10^8	51.8
$h_G(c)(Gap0_8)$	3.44×10^7	22.3
$h_G(c)(Gap0_{10})$	3.42×10^7	23.9
$h_G(c)(Gap0_{12})$	3.42×10^7	23.6
$h_G(c)(Gap0)$	3.28×10^7	12.9
$h_{G:2}(c)(Gap0_8/Gap2_8)$	8.65×10^6	9.1
$h_{G:2}(c)(Gap0_{10}/Gap2_{10})$	8.32×10^6	10.9

5 今後の課題とまとめ

BDDを用いることによって15パズルのためのギャップ集合を効率的に表現することができた．また，提案したデータベースを使用することで，従来法よりも少ない探索ノード数で計算することができた．しかし，従来のパターンデータベースを用いた手法の場合のほうがはるかに効率的であり[5]，さらなる計算の効率化を試みる必要がある．そのためにはギャップ2集合をさらに拡張したギャップ4，6，8集合の利用が必要となる．

しかし，近似した集合のパターンをそれぞれBDDに格納し，それをさらに1つのBDDとするにはかなりのメモリ量がかかってしまう．ギャップ2集合の$Gap2_{12}$については，64Gメモリ仕様のコンピュータを用いても作成することができなかった．今後のアイデアとして，BDDと同様に論理式を表すデータ構造であるZDD (Zero-Suppressed BDD)[4]の利用が挙げられる．

*計算機仕様 ： 2 × 2.93GHz 6 Core Intel Xeon/ Mem: 64G/ OS: Mac OS X 10.7

参考文献

[1] Andersen, H. R.: An introduction to binary decision diagrams. manuscript, available from `http://www.itu.dk/people/hra`, 1999.

[2] 石浦菜岐佐：BDD(二分決定グラフ). 情報処理学会 Information processing Society of Japan, vol. 34, No. 5, 1993.

[3] Korf, R. E.：Depth-first iterateve-deepening: An optimel admissible tree search. *Artificial Intelligence*, vol. 27, No. 1, pp. 97-109, 1985.

[4] Minato, S.: Zero-Suppressed BDDs for set manipulation in combinatorial problems. *Proceedings of 30th ACM/IEEE Design Automation Conference.* pp. 272-277, 1993.

[5] 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.