JAIST Repository
https://dspace.jaist.ac.jp/

Title	15パズルの変形とその最大の最短手数に関する研究
Author(s)	佐藤，隆太郎
Citation	
Issue Date	2021-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/17100
Rights	
Description	Supervisor: 上原 隆平，先端科学技術研究科，修士（情報科学）

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

修士論文

15 パズルの変形とその最大の最短手数に関する研究

佐藤隆太郎

主指導教員 上原隆平

北陸先端科学技術大学院大学
先端科学技術研究科
（情報科学系）

令和 3 年 3 月

Abstract

The 15-puzzle is one of the sliding block puzzles. It consists of a grid of 4×4 with 15 tiles numbered 1 to 15 and one blank space. The goal of this puzzle is to change arrangement of the tiles by repeatedly sliding a tile adjacent to the blank space. The 15-puzzle can be generalized from 4×4 to $m \times n$. In $m \times n$ puzzle, there are $(mn)!$ arrangement patterns. Considering permutations by arrangement of the tiles, it is known that the number of possible states that can be reached by sliding tiles from a given initial state is exactly $(mn)!/2$.

The rule of the 15-puzzle is simple and easy to understand, and the essence of the problem is not lost when the size is changed. The larger puzzle size, the larger the search space becomes. Thus, finding the shortest number of moves is computationally difficult. For this reason, search methods for finding the shortest number of moves have been used in various studies. Typical methods include the A* algorithm and the IDA* algorithm. These algorithms are called heuristic search and use a heuristic function. A heuristic function is a function that gives a prediction of the shortest number of moves required to get from the current state to the target state. A function that returns a value that is always smaller than the actual minimum number of moves is said to be acceptable. One of the heuristic functions for the 15-puzzle is the use of the Manhattan distance. Using the Manhattan distance, it is possible to determine at least how many moves are required for a tile to be in the desired position. The sum of the Manhattan distances of all the tiles is an acceptable heuristic function.

In the n^2-1 puzzle, the minimum number of moves for $n \leq 4$ has been analyzed. The 8-puzzle (3×3) can be solved within 31 moves, and the 15-puzzle (4×4) can be solved within 80 moves. These values are the results of computer computations. The problem of finding the shortest number of moves for an n^2-1 puzzle is known to be NP-hard, and the maximum number of moves for n is not known. Previous research has shown that the lower bounds on the maximum number of shortest moves for n^2-1 puzzles are $n^3 - O(n^2)$ and the upper bounds are $5n^3 + O(n^2)$.

In this study, we limit the length of one side of the puzzle to a small value $(m = 2)$. The purpose of this study is to find the maximum number of shortest moves for n in such a size of $2 \times n$. As a result, we showed the lower and upper bounds of the maximum shortest moves.

As an experimental method, for $n \leq 6$, where all configurations are enumerable, we found the configurations that can reach the target state. Then, we computed the sum of the Manhattan distances of all tiles for that configuration. From the experimental results, we observed a pattern in the arrangement of the tiles that were expected to have the maximum number of moves. Using the arrangement

pattern obtained for $2 \times n$, we showed that the lower bound on the maximum number of shortest moves is $n^2 + \lfloor (3/2)n \rfloor - 1$. We also divided the problem of placing tiles in the desired position into subproblems. Considering that the whole puzzle can be solved by solving the subproblems recursively, we found regularity in the solutions for the subproblems. From the regularity, we showed that the upper bound of the maximum number of shortest moves is $(9/2)n^2 - (9/2)n - 6$.

目 次

第 1 章　はじめに **7**
1.1　研究背景 7
1.2　研究目的 7
1.3　本論文の構成 8

第 2 章　関連研究 **9**
2.1　偶奇性 9
2.2　探索手法 11

第 3 章　下界 **13**
3.1　マンハッタン距離 13
3.2　アルゴリズムの詳細 14
3.3　実験結果 15

第 4 章　上界 **23**
4.1　手法 23
4.2　実験結果 24

第 5 章　考察 **28**

第 6 章　おわりに **30**

図 目 次

1.1 15 パズル 7

2.1 15 パズルの到達不可能な配置の例 9

3.1 本論文で用いる $2 \times n$ パズルの目的状態 13
3.2 最大の $MD(s)$ となる配置 s $(n \leq 6)$ 18
3.3 図 3.2 に関するタイルの配置パターン 19
3.4 $2 \times n$ に対するタイルの配置パターン 20
3.5 目的状態に対するタイルの配置パターン 20
3.6 初期配置 21

4.1 部分問題の目的状態 23
4.2 $f(n)$ 手必要な配置 $(n \leq 6)$ 24
4.3 $f(n)$ 手かかる手順における配置の変化 26

表 目 次

1.1 $m \times n$ パズルにおける最大の最短手数 [3] 8

3.1 実験結果による $2 \times n$ パズルの下界 $(n \leq 6)$ 16

4.1 $2 \times n$ に対する部分問題の計算結果 24
4.2 $f(n)$ 手かかる手順のパターン数 25

5.1 最大の最短手数の下界値と上界値および実際の値 28
5.2 部分問題のサイズを変化させたときの解 28

第1章　はじめに

1.1　研究背景

15 パズルとは，スライディングブロックパズルの一つである．4×4 のグリッドに，1 から 15 までの番号が書かれた 15 枚のタイルと 1 つの空きマスによって構成される (図 1.1)．空きマスに隣接したタイルは，空きマスにスライドさせることができる．このパズルの目的は，スライドする操作を繰り返し行い，与えられたタイルの配置を特定の配置に変化させることである．

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

図 1.1: 15 パズル

15 パズルのサイズを 4×4 から $m \times n$ に一般化したものは，$mn - 1$ 個のタイルと一つの空きマスによって構成され，同様のパズルとして成り立つ．$m = n$ のときのパズルは $n^2 - 1$ パズルと呼ばれる．

このパズルはルールが簡潔でわかりやすく，サイズを変形させても問題の本質を失わない．また，$m \times n$ のサイズに対して，パズルの配置パターンは $(mn)!$ 通り存在し，m, n の値によって指数的に増加する．そのため，初期状態から目的状態に変化させるための最短手数を求める探索手法は，さまざまな研究の対象として利用されている．

1.2　研究目的

$n^2 - 1$ パズルの最短手数を求める問題は NP 困難であることが知られている [1][2]．そのため，n に対する最大の最短手数は分かっていない．先行研究によって，次の表 1.1 に示すサイズは既に解析されており，最大の最短手数が得られている．

m \ n	1	2	3	4	5	6	7	8	9
1	0	1	2	3	4	5	6	7	8
2	1	6	21	36	55	80	108	140	
3	2	21	31	53	84				
4	3	36	53	80					

表 1.1: $m \times n$ パズルにおける最大の最短手数 [3]

本研究では，パズルの一方の長さを小さい値 $(m = 2)$ に制限する．そのような $2 \times n$ のサイズにおいて，最大の最短手数の下界と上界に関する次の 2 つの定理を示す．

定理 1.2.1 $2 \times n$ パズルを解くために少なくとも $n^2 + \lfloor \frac{3}{2}n \rfloor - 1$ の手数が必要である．

定理 1.2.2 $2 \times n$ パズルを解くために必要な手数は高々 $\frac{9}{2}n^2 - \frac{9}{2}n - 6$ である

1.3 本論文の構成

本論文は，本章を含めて 5 つの章で構成する．第 2 章では，15 パズルに関する先行研究を紹介する．第 3 章では，$2 \times n$ パズルの最大の最短手数の下界について示す．第 4 章では，$2 \times n$ パズルの最大の最短手数の上界について示す．第 5 章では，第 4 章で示した上界に対する改善案を考察する．最後に，第 6 章に本論文のまとめと今後の課題を述べる．

第2章　関連研究

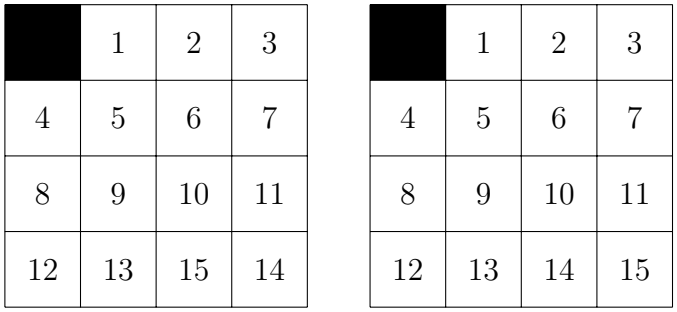
本章では，15 パズルに関する性質や最短手数を求める探索手法について述べる．ここでは 15 パズルについて記載するが，基本的に $m \times n$ のサイズに関しても適用できるものである．

2.1　偶奇性

パズルの配置には，タイルのスライドによって目的状態に到達できる配置と到達できない配置が存在する．この性質に関する定理を次に示す [5].

定理 2.1.1 初期状態 s から目的状態 g に到達できる $\Leftrightarrow$ s と g の空きマスの位置を適当なタイルのスライドによって合わせた後，s から g へ変化させるためのタイルの交換回数が偶数である

15 パズルの配置パターンは $16! = 20922789888000$ 通り存在する．そのうち，タイルのスライドによって目的状態に到達可能な配置は，上記の定理よりちょうどその半数である．残りの半数の配置は，目的の配置に変化させることができない．到達できない 2 つの配置の例を，図 2.1 に示す．



(a) 初期状態　　(b) 目的状態

図 2.1: 15 パズルの到達不可能な配置の例

与えられたパズルに対して，ブランクを含め，タイルに関する配置を数列で表す．空きマスの位置は，はじめに目的の位置と一致させておく．与えられた配置から目的の配置への変化は，数列の各要素を目的の位置に移す操作であり，これ

は置換に対応する．また，タイルの 1 回のスライドは，数列の 2 つの要素の入れ替えであり，互換に対応する．1 回のスライドにより，空きマスは 1 マス移動する．空きマスが目的の位置に戻るには，必ず偶数回移動する．すなわち，偶数回の互換で表される偶置換によってのみ，目的状態に到達できる．

例として，図 2.1 の初期状態から目的状態に到達できるかどうかの判定を記述する．ここで，空きマスは 0 とおく．

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 15 & 14 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{pmatrix} = \begin{pmatrix} 15 & 14 \\ 14 & 15 \end{pmatrix} = \begin{pmatrix} 14 & 15 \end{pmatrix}$$

上記の式より奇置換であるため，到達できないと判定できる．

2.2 探索手法

最大の最短手数を求めるための基本的な探索方法として，幅優先探索が挙げられる．パズルの配置を頂点，タイルのスライドによって遷移可能な配置に対応する頂点同士を辺で結んだグラフを考える．グラフにおける 2 頂点を結ぶ最短の辺の長さは距離と呼ばれ，任意の 2 頂点の組合せに対する距離の最大値はグラフの直径と呼ばれる．15 パズルから考えたグラフの直径は，15 パズルにおける最大の最短手数である．目的状態に対応する頂点から探索を始めることで，すべての頂点との最短距離が得られる．15 パズルに対する幅優先探索の疑似コードを以下に示す．

Algorithm 1 $BreadthFirstSearch(g)$

Input: 目的状態 g
Output: g に変化させるために必要な配置の最大の最短手数
  $OpenList \leftarrow$ 空のキュー
  $ClosedList \leftarrow$ すべての配置の最短手数を ∞
  $OpenList$ に g を追加
  $ClosedList$ に g の最短手数は 0 であることを記録
  while $OpenList$ が空でない **do**
    $v \leftarrow OpenList$ から取り出す
    for v からタイルをスライドして到達できる配置 w **do**
      if $ClosedList$ に記録されている w の最短手数 $>$ ($ClosedList$ に記録されている v の最短手数 $+ 1$) **then**
        $OpenList$ に w を追加
        $ClosedList$ に w の最短手数を更新
      end if
    end for
  end while
  return $ClosedList$ に記録されている最大値

頂点数を $|V|$ とすると，幅優先探索に必要な記憶容量は，$ClosedList$ にすべての配置に関する手数を記録しておく必要があるので $O(|V|)$ である．小さいサイズに対しては全探索が可能だが，パズルのサイズが大きくなると，それに伴って計算時間も記憶容量も増加する．15 パズルにおいて，$|V| = \frac{16!}{2} \simeq 1.0 \times 10^{13}$ である．先行研究によって，幅優先探索より省メモリで動作する探索アルゴリズムとして，Frontier Search が提案されている [6]．$ClosedList$ は，既に探索済みの配置を記録するために必要なデータである．Frontier Search は，$OpenList$ 内に含まれれる配置情報に対して，その配置がどの配置から到達したものかを表現しておく．これにより，タイルをスライドさせたとき，以前の配置に再び変化することをを防ぐ．そ

のため，$ClosedList$ を必要としない．このアルゴリズムを用いて，$2\times7, 2\times8, 4\times4$ パズルにおける最大の最短手数が解かれている．

初期状態が決まっているならば，目的状態までの最短手数を求める探索手法として A*アルゴリズム [7] や IDA*アルゴリズム [8] などの発展的な探索アルゴリズムを用いることができる．ヒューリスティック探索とも呼ばれ，現在の状態から目的状態までの最短手数の予測値を与えるヒューリスティック関数を扱う．この予測値をもとに探索を行うため，アルゴリズムの性能はヒューリスティック関数の性能に依存する．15 パズルに対するヒューリスティック関数で扱われる代表的な方法としては，マンハッタン距離がある．

本研究では，最大の最短手数の下界を求めるために，目的状態までの予測値を与えることができるマンハッタン距離を用いる．また，上界を求めるために，探索空間が小さい問題を考え，その問題に対して幅優先探索を行う．

第3章　下界

本章では，$2 \times n$ パズルに対する最大の最短手数の下界について述べる．本論文では，図 3.1 に示す配置を目的状態とする．

	1	2	$\cdots$	$n-1$
n	$n+1$	$n+2$	$\cdots$	$2n$ -1

図 3.1: 本論文で用いる $2 \times n$ パズルの目的状態

$2 \times n$ パズルにおいて，配置パターンは $(2n)!$ 通りだけ存在する．配置の偶奇性より，与えられた初期状態からタイルのスライドによって目的状態に到達できる数は，ちょうどその半数 $(2n)!/2$ である．最大の最短手数の下界値を求めるうえで，目的状態と同じパリティの配置のみを考えればよい．そのような配置から目的状態に変化させるまでに少なくとも必要な手数を，マンハッタン距離を用いて求める．

3.1　マンハッタン距離

$2 \times n$ パズルにおいて，行 i，列 j に位置するタイル t の位置を (i, j) $(0 \leq i \leq 1, 0 \leq j \leq n-1)$ とすると，図 3.1 で示した目的状態 g のタイル $t(t = 1, \cdots, 2n-1)$ の位置は $(\left\lfloor \frac{t}{n} \right\rfloor, t \bmod n)$ と表せる．このとき，マンハッタン距離 $md(t)$ を次のように定義する．

$$md(t) = |\left\lfloor \frac{t}{n} \right\rfloor - i| + |(t \bmod n) - j|$$

タイルの配置に関するパズルの状態を s とおく．s のすべてのタイル t に対する $md(t)$ の総和は，

$$MD(s) = \sum_{t=1}^{2n-1} md(t)$$

である．位置 (i, j) に存在するタイル t は，目的の位置に対して垂直方向に $|\left\lfloor \frac{t}{n} \right\rfloor - i|$，水平方向に $|(t \bmod n) - j|$ だけ離れている．そのため，t を目的の位置に配置するためには，$md(t)$ 以上の手数が必要である．つまり，$MD(s)$ は s を g に変化させるために必要な手数の下界値といえる．

3.2 アルゴリズムの詳細

ここでは，パズルのサイズ n を入力として与え，最も大きい $MD(s)$ の値とそのパズルの状態 s を出力するアルゴリズムを考える．

まず，目的状態を表す配列 $g = (0, 1, ..., 2n-1)$ を用意する．空きマスは 0 とする．

次に，タイル t が (i, j) に位置するときの $md(t)$ を事前計算しておく．タイルの数は $2n-1$，タイルが位置できる箇所は $2n$ しかない．そのため，任意のタイルと位置のペアに対するマンハッタン距離をすべて配列に保存しておいても，多くの記憶容量を必要としない．

そして，0 から $2n-1$ までの整数で表される $2n$ 個の要素の順列 s を一つずつ列挙し，以下の手順を行う．

1. s が g に到達可能であるかどうかを判定する．

   これは，定理 2.1.1 より，s を g に変化させるための，タイルの交換回数の偶奇と，空きマスの移動回数の偶奇を計算することで求めることができる．

   まず，タイルの交換回数の偶奇を求める．$2n$ 個 の要素からなる配列 A を用意し，s の各タイルがどこに位置するかを対応付けておく．次に，s の位置 (i, j) をひとつずつ確認し，g の位置 (i, j) に存在するタイル t と一致しているか調べる．一致していない場合は，s のタイルを交換する必要がある．(i, j) に位置するタイルを，A を参照して t と交換する．その際，交換したタイルに対応する A の要素を更新し，交換回数をカウントする．この操作をすべてのタイルに対して行い，s を g に変化させる．

   そして，空きマスの移動距離の偶奇についても求める必要がある．本論文では，目的状態の空きマスの位置は $(0, 0)$ としている．そのため，空きマスをタイル $t = 0$ としたときの $md(t)$ から得ることができる．

2. 到達可能ならば，$MD(s)$ を求める．これは，s のタイル t に対して，$md(t)$ の総和を計算すればよい．$md(t)$ は事前計算した値を利用する．

Algorithm 2 CheckSolvability(s)

Input: パズルの配置を表す配列 s
Output: s が目的状態 g に到達可能ならば true, そうでないならば false

```
for s のタイル t do
    配列 A ← t の位置
end for
タイルの交換回数 x ← 0
for s の位置 (i, j) do
    if s の (i, j) に存在するタイル s[(i, j)] ≠ g の (i, j) に存在するタイル t then
        swap(s[(i, j)], s[A[t]])
        A[交換したタイル] ← 交換した位置
        x ← x + 1
    end if
end for
return x の偶奇 = md(0) の偶奇
```

上記のアルゴリズムを用いて，最大の最短手数の下界を求める．パズルのサイズ $N = 2 \times n$ に対して，それぞれの時間計算量は，

- $md(t)$ の事前計算 $O(N^2)$
- $CheckSolvability$ $O(N)$
- $MD(s)$ の計算 $O(N)$

である．パズルの配置パターンは $N!$ 通りあるので，アルゴリズム全体の時間計算量は $O(N \cdot N!)$ である．また，事前計算した $md(t)$ の値を保持するために，$O(N^2)$ の記憶容量を必要とする．

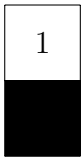
3.3 実験結果

前節で示した手法を，C++で実装し，プロセッサ 2.6GHz Intel Core i5-7300U，メモリ 8GB の計算機を用いて計算を行った．

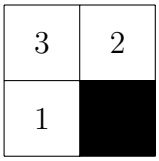
$n \leq 6$ に対する $MD(s)$ の最大値と計算時間を表 3.1 に，その配置 s を図 3.2 に示す．ただし，$2 \times 5, 2 \times 6$ についての配置の数はそれぞれ 40，216 個存在するため，そのうちの 10 個のみを記載する．

n	1	2	3	4	5	6
$MD(s)$ の最大値	1	6	12	21	31	44
計算時間 $[s]$	0.015	0.017	0.021	0.036	0.416	46.2

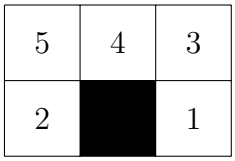
表 3.1: 実験結果による $2 \times n$ パズルの下界 $(n \leq 6)$



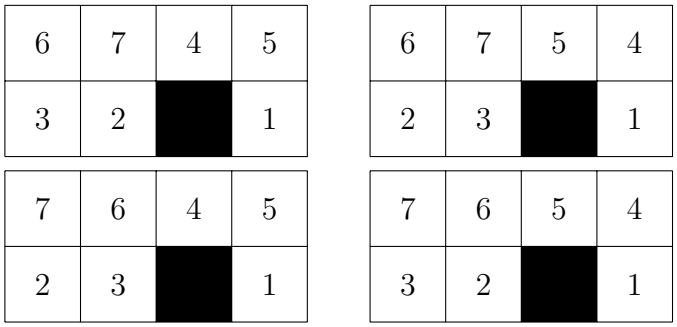
(a) 2×1



(b) 2×2



(c) 2×3



(d) 2×4

7	8	9	5	6
3	4		1	2

7	8	9	5	6
4	3		2	1

7	8	9	6	5
3	4		2	1

7	8	9	6	5
4	3		1	2

7	9	8	5	6
3	4		2	1

7	9	8	5	6
4	3		1	2

7	9	8	6	5
3	4		1	2

7	9	8	6	5
4	3		2	1

8	7	9	5	6
3	4		2	1

8	7	9	5	6
4	3		1	2

(e) 2×5

9	10	11	6	7	8
3	4	5		1	2

9	10	11	6	7	8
3	5	4		2	1

9	10	11	6	7	8
4	3	5		2	1

9	10	11	6	7	8
4	5	3		1	2

9	10	11	6	7	8
5	3	4		1	2

9	10	11	6	7	8
5	4	3		2	1

9	10	11	6	8	7
3	4	5		2	1

9	10	11	6	8	7
3	5	4		1	2

9	10	11	6	8	7
4	3	5		1	2

9	10	11	6	8	7
4	5	3		2	1

(f) 2×6

図 3.2: 最大の $MD(s)$ となる配置 s $(n \leq 6)$

タイル $t(t = 1, ..., 2n-1)$ を，次の 4 つの集合 A, B, C, D に分類する．

- $A = \{t \mid 1 \leq t \leq \lfloor \frac{n}{2} \rfloor - 1\}$
- $B = \{t \mid \lfloor \frac{n}{2} \rfloor \leq t \leq n-1\}$
- $C = \{t \mid n \leq t \leq \lfloor \frac{3}{2}n \rfloor - 1\}$
- $D = \{t \mid \lfloor \frac{3}{2}n \rfloor \leq t \leq 2n-1\}$

t が属する集合によってタイルをラベル付けすると，図 3.2 で示した配置はすべて，次の図 3.3 のようになっている．

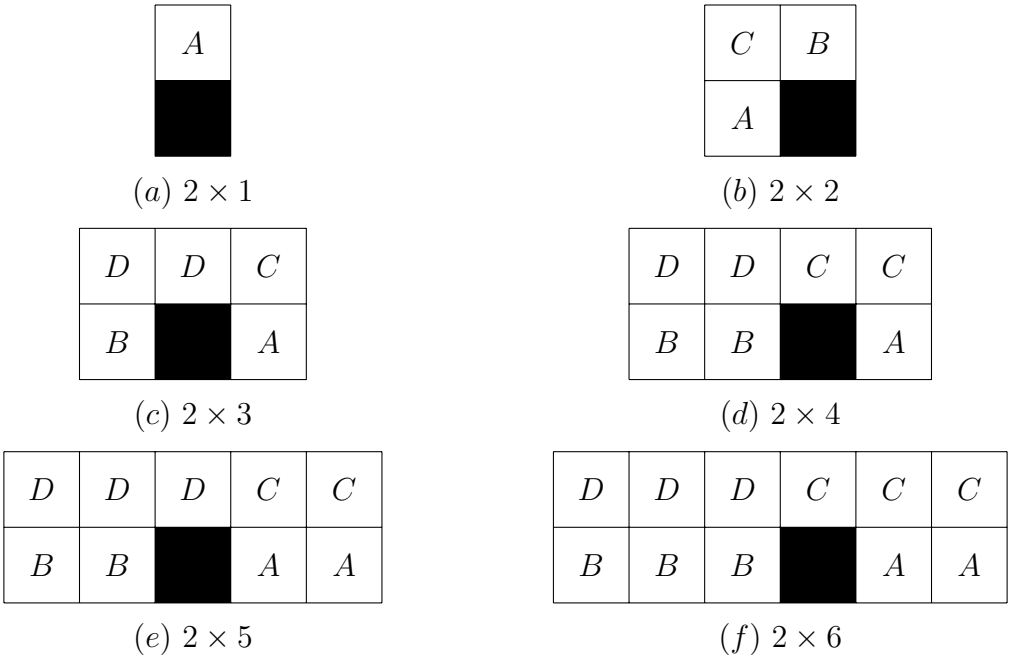


図 3.3: 図 3.2 に関するタイルの配置パターン

図 3.3(c)(d)(e)(f) は，次の図 3.4 のように配置されていることが読み取れる．また，目的状態に対し，各タイルが属する集合によりラベル付けした状態を次の図 3.5 に示す．

D	$\cdots$	$\cdots$	D	C	$\cdots$	$\cdots$	C
B	$\cdots$	$\cdots$	B		A	$\cdots$	A

(a) n が偶数

D	$\cdots$	$\cdots$	D	C	$\cdots$	C
B	$\cdots$	B		A	$\cdots$	A

(b) n が奇数

図 3.4: $2 \times n$ に対するタイルの配置パターン

	A	$\cdots$	A	B	$\cdots$	$\cdots$	B
C	$\cdots$	$\cdots$	C	D	$\cdots$	$\cdots$	D

(a) n が偶数

	A	$\cdots$	A	B	$\cdots$	B
C	$\cdots$	C	D	$\cdots$	$\cdots$	D

(b) n が奇数

図 3.5: 目的状態に対するタイルの配置パターン

図 3.4 の配置から図 3.5 の配置への変化に必要な手数について，タイルのマンハッタン距離の総和を用いて考えることで，以下の定理を示す.

定理 3.3.1 $2 \times n$ パズルを解くために少なくとも $n^2 + \lfloor (3/2)n \rfloor - 1$ の手数が必要である.

証明

$2 \times n$ パズルに対して，初期配置として次の図 3.6 のような状態を考える．

$\frac{3}{2}n$	$\cdots$	$\cdots$	$2n$ -1	n	$\cdots$	$\cdots$	$\frac{3}{2}n$ -1
$\frac{n}{2}$	$\cdots$	$\cdots$	n -1		1	$\cdots$	$\frac{n}{2}$ -1

(a) n が偶数

$\lfloor\frac{3}{2}n\rfloor$	$\cdots$	$\cdots$	$2n$ -1	n	$\cdots$	$\lfloor\frac{3}{2}n\rfloor$ -1
$\lfloor\frac{n}{2}\rfloor$	$\cdots$	n -1		1	$\cdots$	$\lfloor\frac{n}{2}\rfloor$ -1

(b) n が奇数

図 3.6: 初期配置

n が偶数の場合

タイル 1 は，目的の位置と水平方向に対して $\frac{n}{2}$，垂直方向に対して 1 だけ離れている．よって，マンハッタン距離は $\frac{n}{2}+1$ となる．ほかの 2 から $2n-1$ までのすべてのタイルについても同様である．

したがって，図 3.6(a) の配置に関するマンハッタン距離の総和は，

$$(2n-1)(\frac{n}{2}+1) = n^2 + \frac{3}{2}n - 1$$

である．

n が奇数の場合

タイル 1 は目的の位置と水平方向に対して $\frac{n-1}{2}$，垂直方向に対して 1 だけ離れている．よって，マンハッタン距離は $\frac{n-1}{2}+1$ となる．2 から $\lfloor\frac{n}{2}\rfloor-1$ まで，$\lfloor\frac{3}{2}n\rfloor$ から $2n-1$ までのタイルについても同様である．

タイル $\lfloor\frac{n}{2}\rfloor$ は，目的の位置と水平方向に対して $\frac{n+1}{2}$，垂直方向に対して 1 だけ離れている．よって，マンハッタン距離は $\frac{n+1}{2}+1$ となる．$\lfloor\frac{n}{2}\rfloor+1$ から $n-1$，n から $\lfloor\frac{3}{2}n\rfloor-1$ までのタイルについても同様である．

したがって，図 3.6(b) の配置に関するマンハッタン距離の総和は，

$$\frac{n-1}{2}(\frac{n-1}{2}+1) + (n-1)(\frac{n+1}{2}+1) + \frac{n+1}{2}(\frac{n-1}{2}+1) = n^2 + \lfloor\frac{3}{2}n\rfloor - 1$$

である．

以上より，$2 \times n$ パズルを解く最大の最短手数の下界として，$n^2 + \lfloor \frac{3}{2}n \rfloor - 1$ が得られる．

$\square$

第4章　上界

本章では，$2 \times n$ パズルに対する最大の最短手数の上界について述べる．

4.1　手法

本論文では，$2n-1$ 個すべてのタイルを目的の位置に配置する問題を部分問題に分割して再帰的に解く．部分問題は次のようなものを考える．

「$2 \times n$ パズルの目的状態に対して，右端の列 1 列以外のタイルの区別を無視する．そのようなパズルを考えたとき，目的状態に変化させるために必要な手数の最大値はいくつか」

この部分問題を解くうえでは，配置したい右端のタイルのみ区別できていれば良いので，すべてのタイルに異なる番号をラベル付けする必要はない．そのため，図 4.1 のようにラベル付けしたパズルをこの部分問題における目的状態とする．ここでは，区別する必要のないタイルはすべて X とラベル付けしておく．

また，このようないくつかのタイルのみの配置について考える手法はパターンデータベースと呼ばれる．タイル全体を解く元の問題の部分問題であるため，許容的なヒューリスティック関数としても用いられる [9].

	X	$\cdots$	X	$n-1$
X	X	$\cdots$	X	$2n$ -1

図 4.1: 部分問題の目的状態

図 4.1 のパズルにおける配置パターンは，$2n(2n-1)(2n-2)$ 通りしか存在しない．そのため，目的状態を初期状態とし，到達できるすべての配置とその最短手数を，幅優先探索による全探索で求める．

$2 \times n$ パズルに対する部分問題の解を $f(n)$ とおくと，パズル全体を解くために必要な手数の上界値は，$f(n) + f(n-1) + f(n-2) + \cdots + f(2) + f(1)$ として得ることができる．

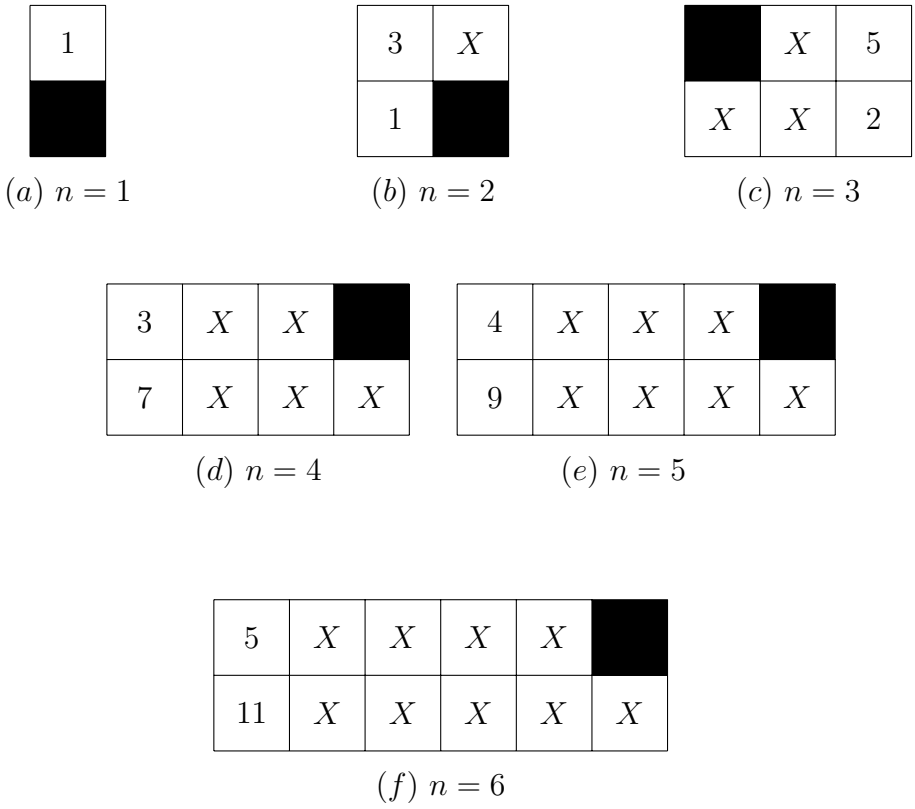
4.2 実験結果

前節で示した手法を用いて，$2 \times n$ パズルに対する部分問題の解 $f(n)$ の結果と計算時間を，表 4.1 に示す．

n	1	2	3	4	5	6	7	$\cdots$	100
$f(n)$	1	6	20	27	36	45	54	$\cdots$	891
計算時間 $[s]$	0.012	0.015	0.018	0.026	0.027	0.026	0.021	$\cdots$	510

表 4.1: $2 \times n$ に対する部分問題の計算結果

目的状態へ変化させるために $f(n)$ 手必要な配置を，図 4.2 に示す．



(a) $n = 1$　(b) $n = 2$　(c) $n = 3$

(d) $n = 4$　(e) $n = 5$

(f) $n = 6$

図 4.2: $f(n)$ 手必要な配置 $(n \leq 6)$

表 4.1 より，$n \geq 4$ に対して，$f(n) = 9(n-1)$ であることが読み取れる．また，図 4.2(d)(e)(f) を見ると，目的のタイルが左端の列，空きマスが右上の位置であることがわかる．そのため，部分問題を解くために最も手数のかかる配置に関する手順には，規則性があると考えられる．図 4.2 で示した $f(n)$ 手必要な配置から目的状態に変化させるためのタイルの動きに関する手順は，1 つだけでなく，いくつか存在した．その手順のパターン数を，表 4.2 に示す．

n	1	2	3	4	5	6	7	$\cdots$	100
手順の数	1	2	8	12	24	40	60	$\cdots$	19404

表 4.2: $f(n)$ 手かかる手順のパターン数

表 4.2 で示した通り数のうち，$n = 4, 5, 6, 7$ における手順の一つを，以下に示す．ここでは，ブランクの上下左右の動きをそれぞれ UDLR とおき，それを手順として表す．なお，手順の一連の流れを A, B, C, D, E の 5 つに区分する．

2×4

$$\underbrace{DLLLU}_{A}\underbrace{RRDLLU}_{B}\underbrace{RRDRULDLURDRU}_{D}\underbrace{LLL}_{E}$$

2×5

$$\underbrace{DLLLLU}_{A}\underbrace{RRDLLU}_{B}\underbrace{RRRDLLU}_{C}\underbrace{RRDRULDLURDRU}_{D}\underbrace{LLLL}_{E}$$

2×6

$$\underbrace{DLLLLLU}_{A}\underbrace{RRDLLU}_{B}\underbrace{RRRDLLURRRDLLU}_{C}\underbrace{RRDRULDLURDRU}_{D}\underbrace{LLLLL}_{E}$$

2×7

$$\underbrace{DLLLLLLU}_{A}\underbrace{RRDLLU}_{B}\underbrace{RRRDLLURRRDLLURRRDLLU}_{C}\underbrace{RRDRULDLURDRU}_{D}\underbrace{LLLLLL}_{E}$$

上記の手順について，2×5 を例に，図 4.2(e) の配置から目的状態までの変化を次の図 4.3 に示す．

	X	X	X	X
4	9	X	X	X

(a) 図 4.2(e) の配置から手順Aによる変化

	X	X	X	X
X	4	9	X	X

(b) (a) の配置から手順Bによる変化

4		X	X	X
9	X	X	X	X

(c) (b) の配置から手順Cによる変化

X	X	X	4	
X	X	X	X	9

(d) (c) の配置から手順Dによる変化

	X	X	X	4
X	X	X	X	9

(e) (d) の配置から手順Eによる変化

図 4.3: $f(n)$ 手かかる手順における配置の変化

この一連の手順は，$n \geq 4$ に対して見つかった．区分したそれぞれの手順の手数について，A は $n-1$ 手，B は 6 手，C は $7(n-4)$ 手，D は 13 手，D は $n-1$ 手である．よって全体で，

$$n-1+6+7(n-4)+13+n-1=9(n-1)$$

であることが示せる．$n \leq 3$ のサイズにおいて，区分 D の動きはできないため，上記で示した手順は見つからなかったと考えられる．

定理 4.2.1 $2 \times n$ パズルを解くために必要な手数は高々 $\frac{9}{2}n^2 - \frac{9}{2}n - 6$ である

証明

$2 \times n$ パズルを解くために必要な手数の上界を $UpperBound(n)$ とすると，

$$UpperBound(n) = 9(n-1) + UpperBound(n-1) \quad (n \geq 4)$$

と表せる．ただし，$UpperBound(3) = 21$ とおく．なぜなら，表 1.1 より，2×3 パズルにおける最大の最短手数は 21 なので，これが最良だといえる．この式について解くと，

$$\begin{aligned}
UpperBound(n) &= 9n - 9 + Upperbound(n-1) \\
&= 9(n + n - 1 + n - 2 + \cdots 4) - 9(n-3) + UpperBound(3) \\
&= 9\sum_{k=4}^{n} k - 9n + 27 + 21 \\
&= 9(\sum_{k=1}^{n} k - \sum_{k=1}^{3} k) - 9n + 48 \\
&= \frac{9}{2}n(n+1) - 9(1+2+3) - 9n + 48 \\
&= \frac{9}{2}n^2 - \frac{9}{2}n - 6
\end{aligned}$$

が得られる． □

第5章　考察

まず，前章により得られた下界，上界を表 5.1 に示す．

n	4	5	6	7	8	9	10
下界値	21	31	44	58	75	93	144
最大の最短手数	36	55	80	108	140		
上界値	58	84	129	183	246	318	399

表 5.1: 最大の最短手数の下界値と上界値および実際の値

表 5.1 から，実際の最大の最短手数と比べると，今回示した下界と上界には改善の余地があると思われる．

ここでは，上界の改善案を考える．第 4 章で示した部分問題は，パズルの右端の列 1 列に対するタイルのみ考えた．ここでは，区別するタイルの数を増やすために，右端の列から k 列に対して同様のことを考える．前述した部分問題を，次のように言い換える．

「$2 \times n$ パズルの目的状態に対して，右端の列から k 列以外タイルの区別を無視する．そのようなパズルに対し，目的状態に変化させるために必要な最大の手数はいくつか」

表 5.2 にその結果を示す．なお，計算時間は $n = 7$ に対して，$k = 1$ のとき $0.016[s]$, $k = 2$ のとき $1.01[s]$，$k = 3$ のとき $151[s]$ だった．

n \ k	1	2	3
2	6	6	
3	20	21	21
4	27	35	36
5	36	48	54
6	45	61	71
7	54	74	88

表 5.2: 部分問題のサイズを変化させたときの解

$k = 1$ のときは，$n \geq 4$ に対して $9(n-1)$ 手であった．表 5.2 を見ると，$k = 2$ の

とき $n \geq 4$ に対して $13n - 17$ 手，$k = 3$ のとき $n \geq 5$ に対して $17n - 31$ 手となっていることが読み取れる．このことから，$k = 1$ のときと同様に，$k \geq 2$ でも空きマスの動きに関する規則性があり，そのような手順から再帰的に解くことで上界が改善されると推測できる．

第6章　おわりに

本研究では，15 パズルと知られるパズルのサイズを 4×4 から $2 \times n$ のサイズに限定し，その最大の最短手数についての考えた．その結果，

下界 $n^2 + \lfloor \frac{3}{2} n \rfloor - 1$

上界 $\frac{9}{2} n^2 - \frac{9}{2} n - 6$

であることを示した．しかし，既に解かれている $n \leq 8$ における最大の最短手数と比べると，改善の余地があると考えられる．

参考文献

[1] D. Ratner and M. K. Warmuth. The (n^2-1)-puzzle and related relocation problems, Journal for Symbolic Computation, 10:11-137 (1990)

[2] Erik D. Demaine, Mikhail Rudoy. A simple proof that the (n^2-1)-puzzle is hard, Theoretical Computer Science, Vol.732, pp.80-84 (2018)

[3] http://oeis.org/A151944 (最終閲覧日：2021 年 1 月 28 日)

[4] I. Parberry. A real-time algorithm for the (n^2-1)-puzzl, Information Processing Letters 56 (1) 23-28 (1995)

[5] Johnson, W. W and Story, W. E. Notes on the “15” puzzle, American Journal of Mathematics, Vol.2, No.4, pp.397-404 (1879)

[6] R.E. Korf. Linear-time Disk-based implicit Graph Search, Journal of the ACM 55 No.6 (2008)

[7] Hart, P.E., Nilsson, N.j., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths, IEEE transactions on Systems Science and Cybernetics, Vol.4 No.2, pp.100-107 (1968)

[8] R.E. Korf. Depth-First Iterative-Deepening: An optimal admissible tree search, Artificial Intelligence, Vol.27, No.1, pp.97-109 (1985)

[9] R.E. Korf and Felner, A. disjoint pattern database heuristics, Artificial Intelligence, Vol.134, No.1-2, pp.9-22 (2002)