ATLANTIS PRESS

Advances in Intelligent Systems Research, volume 134

2nd International Conference on Control, Automation, and Artificial Intelligence (CAAI 2017)

State Space and Optimal Solution of Jigsaw Puzzle

Zhongping Liu and Xiaoyuan Liu

[1]College of Computer Science and Engineering, Yulin Normal University, Yulin, China, 537000

[2]Library, Yulin Normal University, Yulin, China, 537000

***Abstract*—this paper hopes to find a planning algorithm for puzzle which is not based on search. The main idea is using the method of reducing dimension to decompose the puzzle. Firstly, we defined the jigsaw puzzle as discrete state space, and provided the functions of action space generation and state transformation; secondly, we built two breadth first trees through goal-driven strategies. One is a set of state spaces which are solvable, and another is a set of state spaces which have non solution; thirdly, a theorem for determining whether state space have a solution was obtained through extracting feature of inverse number from state spaces, and then we proposed an initialization algorithm which can get solvable state surely; Finally, we suggested a method to reduce the dimension step by step, and put forward a two-dimensional bubbling algorithm. Experiments show that this algorithm is simple and effective, and the results meet expectations.**

Keywords—jigsaw puzzle; eight digits; breadth first tree; inverse number; two-dimensional bubbling

I. Introduction

Jigsaw (8 or 15 digits puzzle) belong to the family of sliders [1], which are often used as test cases for new search algorithms in AI. The game includes 8 or 15 digital pieces and a space (Fig.1). Its operation is to move a chess piece with the help of the space, that is, the chess pieces adjacent to the space can slide into the space. The goal of the game is to achieve a particular state, as shown in Fig.1 (b) or (c).

3	5	8
1		2
6	7	4

(a)

1	2	3
4	5	6
7	8	

(b)

1	2	3	4
5	6	7	8
9	a	b	c
d	e	f	

(c)

FIGURE I. CURRENT STATE AND GOAL STATE

The main contributions of this paper are as follows:

(1) The state space of the jigsaw puzzle is defined by planning algorithm as the discrete state space representation, and generating function of the action space and the state transition function are given.

(2) The whole state space of the 8 digits puzzle is structured into two breath first trees by goal-driven strategy. The root of the one is the state of "123456780", and the other nodes are solvable. The shortest path of any node to the root node can be obtained by the breath first tree. The root of another is the state of "123456870", and all nodes are unsolvable.

(3) The state space feature is characterized by the state inversion number, and the theorem is given to determine if the state is solvable, and the initialization method is proposed to ensure the state is solvable.

(4) This paper suggests the planning algorithm which reduce dimension from 8 digits to 3 digits through sorting the first row and first column. The algorithm is universal and can be extended to higher dimensions.

II. The State Space Representation

The state space of the jigsaw problem is discrete, finite and definite: the status number of 8 digits: $9! = 362880$; the 15 digits: $16! \approx 2\times10^{13}$. Although the state space of the problem is an undirected graph with a degree of 4, because of the large number of vertices and edges, it can't be completely described by the graph, so we adopt the discrete representation of the problem state space [2].

Definition 1: Puzzle state space representation

A non-empty state space X, it is a finite set of states.

For each state $x \in X$, have a limited action space *U(X)*.

For each $x \in X$ and $u \in U(x)$, a state transfer function *f* generate state $f(x,u) \in X$.

Initial state: $x_I \in X$.

Target state: $x_G \in X$.

State space: 8 digits use [0-8]; 15 digits use [0-9] and [a-f]. For state representation, the most natural idea is to use an array, but after taking into account the state retrieval convenience, we use row-major string representation. Fig.1 (a) of the state can be expressed as x= "358102674".

Action space: There are at most 4 kinds of legal moves. The digits around the space can be moved to the space towards the up, down, left or right. In order to simplify the description of the action, we use spaces instead of digits mobile (opposite direction). According to the status space of the current x, you can figure out the legal action contained in *U (x)*. We use the character constant *LEFT* to represent the left shift of the space, *DOWN* to indicate the move down, *RIGHT* to move right, and *UP* to move up. The following algorithms are available:

Algorithm 1: Action space generating function

Copyright © 2017, the Authors. Published by Atlantis Press.
This is an open access article under the CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/).

```
Input: status x
Output: action spaces U
  function generate_action(x)
  begin
      dim=round(sqrt(x.length));
      U:=[];
      index=x.find("0")
      if index mod dim>0 then U.append(LEFT)
      if floor(index/dim)<dim-1 then U.append(DOWN)
      if index mod dim<dim-1 then U.append(Right)
      if floor(index/dim)>0 then U.append(UP)
      return U
  end
```

State transfer function: The new status x' can be generated by implementing action *U* on current status x. Algorithm is as follows:

```
Algorithm 2:  State transition function
Input:  status x, action space U
Output:  new status x'
  function f(x,u)
  begin
      dim=round(sqrt(x.length));
      index:=x.find("0");
      temp:=0;
      switch(u)
      begin
          case LEFT:  temp:=index-1; break;
          case DOWM:  temp:=index+dim; break;
          case RIGHT: temp:=index+1; break;
          case UP:    temp:=index-dim; break;
      end
      x.swap(index,temp);
      return x;
  end
```

Initial status: any character arrangement that is different from the target state can be generated using random functions.

Target status: as a digits game, it can have a lot of interesting target state, but for the puzzle, target state is usually in Figure 1 (b) or (c).

III. Breadth first Tree of Puzzle Problem

Through the study found that, regardless of 8 or 15 digits, the state space of X is composed of two equal, each other is not communicated with the 4 degrees of undirected graph. That is to say, half of the states can't get to target through the transfer function. In order to facilitate the characteristics of state space for more in-depth analysis, we adopt the strategy of goal-driven [3] and the breadth first algorithm to construct the optimal solution tree on 8 digits puzzle.

To facilitate representation, we define a four tuple to represent nodes in a tree [3]:

Definition 2: A node of a breadth first tree can be represented as a four tuple [n, x, l, p], which:

n is the number of nodes, and each state x corresponds to a four tuple with a unique number.

x is a new state that is generated by the state transition function (the state that is already in the open or closed table will be discarded).

l is layer of the nodes in the tree, and the shortest path to reach its target (the root node) .

p is the number of the parent node used to construct the best path.

In the breadth first search process, we use the list of open and closed to track through space. Open is a first in first out queue that lists the status that has been generated, but the child has not yet been analyzed. The closed records have been analyzed. The specific implementation is as follows:

Algorithm 3: The establishment of breadth first tree based on goal-driven strategy.

```
Input: none
Output: closed list
function breadth_first_tree()
begin
    open:=[start];      %initialized by goal status
    closed:=[];
    while open!=[] do
    begin
        remove leftmost state from open,call it x;
        generate children of x;
        put x on closed;
        discard children of x if already on open or closed;
        put remaining children on right end of open;
    end
    return close;
end
```

With the x= "123456780" as the root node to initialize the program, we can get a 3 fork tree with the best solution. The total number of nodes is 181440, and the distribution of nodes is shown in Fig.2. Another is composed of nodes with no solution, the root node is set to x= "123456870". 15 digits puzzle is similar, but due to the huge number of nodes, it's difficult to construct the 3 fork tree.

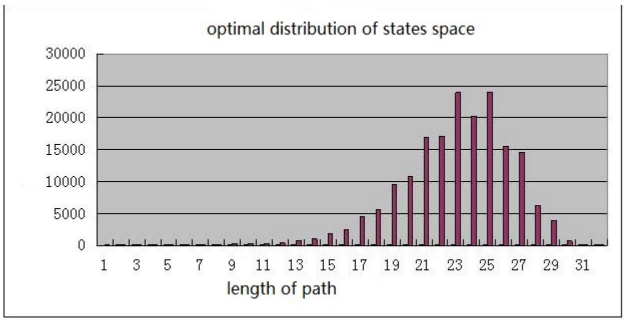

FIGURE II. NODE DISTRIBUTION OF OPTIMAL SOLUTION

The maximum value of optimal solution is 31, the average is 21.97, and the median was 24. This statistical data can be used to control the maximum depth of the depth first algorithm.

IV. THE SOLUTION EXISTENCE

Through in-depth analysis of the breadth first tree, we have the following basic facts:

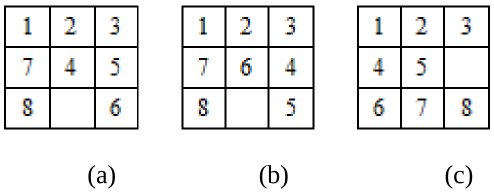

FIGURE III. SPACE SHIFT

Theorem 1: States with the same digits sequence have the same solvability (the solvability is independent of the space location)

Prove：If the two states have the same solvability, they can move from one state to another through legally moved. We regard fig.1 (b) as the target state to prove the correctness of the theorem.

(1) Obviously, the horizontal movement of the spaces does not change the order of the numbers.

(2) In Fig.1 (b), we can pull 7 and 8 aside to become fig. 3(a) through the action sequence: *action= (UP, LEFT, LEFT, DOWN, RIGHT)*. Then, make 6 and 7 next to each other implement action sequence: *action=（RIGHT, UP, LEFT, DOWN*）(fig.3(b)). Finally, we can arrive fig.3(c) through implement action sequence: *action=（LEFT, UP, RIGHT, RIGHT, DOWN, LEFT, LEFT, UP, RIGHT, RIGHT*）.From fig.1 (b) to fig.3 (c), we can move space vertically with no changing of the digits sequence.

(3) Verdict: the solvability of status has nothing to do with the space position in 8 digits puzzle.

Since space does not affect the solvability, we can study the relative order for analyzing the solvability. In order to study the digits relative position, we have the following definition:

Definition 3: Digit inverse number: In the string representation of a state, the count of digits placed in front of a digit and greater than the digit is called the inverse number. As shown in Fig.1 (a), the inverse number of the digit 1 and 2 is 3, digit 3, and the inverse number of others is 0.

Definition 4: State inverse number: The sum of the digits inverse numbers in the state (except the spaces) is called state inverse number.

As shown in Fig.1 (a), the inverse number of the state is: 3+3+1+1+4=12.

Because the digits are fewer in puzzle problem, we can get inverse number through counting.

Algorithm 4: The direct counting method for the inverse number of states.

```
Input: state x
Output: state inverse number rev.
function rev_nbr(x)
begin
rev:=0;
    for i=0 to x.length-1
    begin
        for j=0 to i-1
        begin
            if x.getChar(j)>x.getChar(i) then rev=rev+1;
        end
    end
    return rev;
end
```

By definition 4, we can obtain the theorem of whether the following solution exists.

Theorem 2: For 8 digits puzzle, necessary and sufficient conditions of the solvability is that the state inverse number is an even number.

Prove:

Necessity: If the state x is solvable, then it is possible to proceed from the target state and to achieve the status x by legal movement. The following three aspects are examined in terms of the inverse numbers of these states x.

(1) Inverse number of target state is 0, which is even.

(2) Horizontal movement is only a change in the position of the digit and space, without changing the inverse number. As shown in Fig.1 (a), horizontal movements of digit 1 or 2 do not change the order of the numbers, and the state inverse

number remains the same.

(3) In 8 digits puzzle, the space between its upper and lower digits contains two digits, and the vertical movement will change relative positions of the middle digits. So, the incremental state inverse number is -2, 0 or 2, that's to say, the parity of the state inverse number is unchanged. As shown in Fig.1 (a), if the digit 5 moves down, resulting in 5 and 1 relative position changes, the state reverse number is reduced by 1. 5 and 8 relative position changes, the state inverse number increased by 1. The total inverse number increases by 0, that is, the parity of the inverse number is not changed.

(4) From the above, if the state has a solution, then the state inverse number must be even.

Sufficiency: In 8 digits puzzle, two 3-fork-trees are constructed by algorithm 3. One has the root node state x= “123456780” and state inverse number 0 (even). Its child nodes are solvable. Another 3-fork-trees root node is x= "123456870", and the state inverse number is 1 (odd), and the child nodes are all without solution. For an arbitrary state x, if its state inverse number is even, it cannot be inverted into odd number by legal movement. Therefore, the state must be solvable. The above theorem 1, 2 and 3 are in 8 digits, in fact, it is easy to be extended to 15 digits and so on.

Corollary 1: For mode $n \times n$, if n is odd, the parity does not change with the vertical mobile of the space; if n is even, the vertical mobile of space will change parity of the state inverse number.

Corollary 2: For mode $n \times n$, if n is odd, the puzzle problem of necessary and sufficient conditions of the solutions is that the state inverse number is even; if n is even, the necessary and sufficient conditions is that the state inverse number and row number of space have different parity.

V.GAME INITIALIZATION

To ensure that the state of the initialization algorithm must be solvable, we present the following theorems:

Theorem 3: Interchange of any two non 0 (space) characters in the status string will change the parity of the state reverse number.

Prove: (1) Two non 0 digits exchange positions; the inverse number of these two characters varies by 1.

(2) The sum of inverse number of the two digits and middle of the two is even.

(3) The relative position of the two digits and outside of them has no change, so, the total inverse number is 0.

(4) In combination with (1) (2) (3), if any two non - 0 digits positions interchange, the parity of the state reverse number will be changed.

According to theorem 2, if the number of exchange is even, the interchange will not affect state solvability. So, we propose the following algorithms for game initialization:

Algorithm 4: Game initialization for 8 digits

Input: none

Output: state x

```
function initialize_3()
begin
    x:=“12345678”;
    for i=1 to x.length
    begin
        do
            index=random(1:8);
        while(index==i);
        x.swap(i,index);
    end
    x.insert(index,“0”);
    return x;
end
```

USING PLANNING ALGORITHMS TO SOLVE

Current solving strategies of puzzle are built on the basis of search. In this paper, a planning algorithm is put forward on the basis of a great deal of practical operation. The idea of the algorithm is breaking the problem down and then approaching the target state step by step.

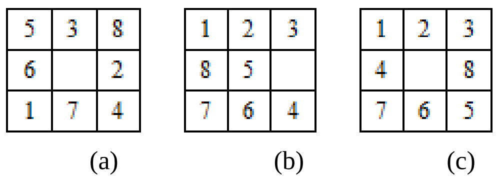

FIGURE IV. PLANNING ALGORITHM

Strategy 1: The solving process is divided into 3 steps:

(1) Making the first row orderly: from fig.4(a) to (b).

(2) Making the first column orderly: from (b) to (c).

(3) Rotating the remaining digits to right position. From fig. 4(c) to fig.1(b).

In order to from easy to difficult, we analyze the method of implementation turn to three steps. For the first row and first column orderly as fig.4(c), it's very easy to rotate three digits to right position. All we need is implement *action=(DOWN, RIGHT, UP, LEFT, DOWN, RIGHT).*

In Fig.3 (b) of the first row orderly, our goal is to move 4 and 7 to their place. A large number of experiments have found that the key to the homing of 4 and 7 is not its absolute position, but the relative position. If the following 5 numbers and spaces are considered as a circle, they can be rotated back when 4 and 7 are order relatively.

Strategy 2: Make first column orderly base on the first row orderly (That's 4 and 7 go back).

(1) If 4 and 7 are in reverse order, move 4 or 7 to right

place or their diagonal position, rotate the other three digits and disassemble the 4 and 7.

(2) If the 4 and 7 relative disorder, one of them is moved to adjacent corner of their right place, rotate the other three digits to make 4 and 7 orderly.

(3) If the 4 and 7 is order relatively, rotate 5 digits to make 4 and 7 homing.

In Figure 3 (a) of the situation, to make the first row order is not so simple. The action become complex, but the basic principle is to move the 1, 2 and 3 up. Because digits can't exchange position directly, it can only be changed by space, so it needs skill to move. We find that this problem can still be divided into two steps, one is make 1, 2, 3 bubbling(move to the upper middle), another is make 1, 2, 3 homing(make first row orderly). For the 1, 2, 3 bubbling, we propose a two-dimensional bubble strategy, the main idea is to raise the small digits 1, 2, 3 and sink the big digits 7 and 8.

Strategy 3: Two dimensional bubbling

If the 1, 2, 3 and the space are not all in the upper-middle part, repeat the following operations:

(1) If the space in the upper part, the middle small digits (1, 2, 3) go up. If there are no small digits in the middle, then the digits (4, 5, and 6) rises.

(2) If the space is in the middle, If the small numbers are all in the middle and lower part, the upper digits will sink, otherwise the lower digits will float.

(3) If the space is in the lower part, if the middle contains large digits, sink it, else if the middle contains middle digits, sink it, otherwise, sink the digits above the space.

When 1, 2, 3 float to topper, we can arrange the first row. The strategy is make 1 and 2 or 2 and 3 orderly relatively, and then joint the third digits, finally, rotate them to right place.

Strategy 4: Make the first row orderly when small digits are all in topper.

(1) If the 123 reverse order, move 1 or 3 to right place or its diagonal position, and then rotate the other three digits in order to disassemble 2.

(2) If 12 and 23 are all relativity disorder, move 1 or 3 to the adjacent corner of its target position, and rotate the other three numbers so that 12 or 23 is relatively orderly.

(3) If 12 or 23 are relatively order, move the two digits to the side, rotate the other three digits, and make the 123 relatively orderly.

(4) Rotate 5 digits, make them homing.

VI. SUMMARY

In this paper, the state space and the optimal solution of the puzzle are analyzed. The judgment theorem of whether the puzzle is solvable and initialization method are proposed. Finally, a stepwise decomposition based dimension reduction algorithm is proposed. The algorithm is clear and feasible, and can be easily extended to higher dimensions.

REFERENCES

[1] Stuart J. Russell, Peter Norvig. Artificial Intelligence: A Modern Approach, Third Edition[M]. Translated by Yin Jianping, Zhu En, Liu Yue, et al Beijin: TsingHua University Press, 2013(in Chinese)

[2] Steven M. LaValle. Planning Algorithms[M]. Translated by Zhang Qingya, Sun Dong, et al. Beijin: TsingHua University Press, 2011 (in Chinese)

[3] George F. Luger. Artificial Intelligence: Structures and Strategies for Complex Problem Solving, Sixth Edition[M]. Translated by Guo Maozu, Liu Yang, Xuan Ping, et al. Beijin: China Machine Press, 2010 (in Chinese)