Solving Hua Rong Dao Puzzle and Its Variations using Breadth First Search

Zhikun ZHAO
School of Computer & Information Engineering
Shandong University of Finance
Jinan, China, 250014
zhaozk@sdfi.edu.cn

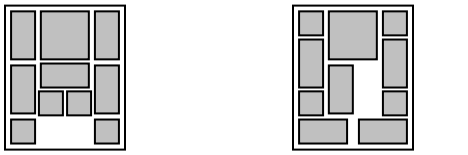
Yinglei XU
School of Computer & Information Engineering
Shandong University of Finance
Jinan, China, 250014
xuyl@sdfi.edu.cn

***Abstract*—A Hua Rong Dao algorithm is proposed in this paper, which can find best solutions for both basic version and variation version of the puzzle. A state is represented with the contents of the 4×5 grid, so that states of variation version can be also represented. Three conditions are used to detect duplicate states in the BFS. The algorithm has been implemented on Java SE platform and applied to find the best solutions of 374 configurations. The results verified the correctness and the effectiony of the algorithm.**

Keywords – Hua Rong Dao, Klotski, Breadth First Search

I. INTRODUCTION

Hua Rong Dao is a famous Chinese puzzle, which is a variation of the well known sliding block puzzle named Klotski [2]. In a basic Hua Rong Dao game, several different sized block pieces are placed in a box. The box is in 4×5 size and inside which are one 2×2, four 1×1 and five 2×1 block pieces. See Figure 1(a). The block pieces in 2×1 size might be placed horizontally or vertically. The block pieces may have different initial positions in a game configuration, see Figure 1(b), but the goal is the same: to move the largest block to the bottom middle location of the box with a minimum amount of moves.



(a) Basic Game (b) Another Configuration

Figure 1. Basic Hua Rong Dao Game

(a) Variation 1 (b) Variation 2

Figure 2. Two Variations

Besides the basic version, the Hua Rong Dao puzzle has many variations. For example, there are fourteen 1×1 block pieces and no 2×1 block piece in the version shown in Figure 2(a); and there are six 2×1 block pieces and one 1×1 block pieces in the version shown in Figure 2(b). A website lists 374 different initial configurations of the Hua Rong Dao game, which covers almost all the configurations currently known[7]. Some of them belong to the basic version and some belong to the variations.

To find out the best solution (i.e. the solution with minimum amount of moves) of an configuration using computer, breadth first search (BFS) algorithm is a feasiable method [6]. For an configuration, the best solution usually has less than 200 steps, and there are usually 2 or 3 choices in each step. Considering that mang choices may lead to duplicate states, if the BFS algorithm is adopt to the Hua Rong Dao puzzle, there will not be combinational explosion problem. Based on the above consideration, many BFS based algorithms have been presented. Their difference is on the state representation and pruning strategy. However, these algorithms are usually designed only for the basic version of the Hua Rong Dao puzzle. According to our knowledgements, there is not an algorithm that can solve all the configurations of the basic version and even all the variation versions.

In this paper, a HuaRongDao algorithm is proposed, which can solve all the known configurations of the basic Hua Rong Dao puzzle as well as the variations. The algorithm is based on the BFS, but applies a new state representation that can represent all the configurations and the variations. With a proper pruning method, the efficiency of the algorithm is guaranteed. The best solutions of all the configurations have been found by the algorithm, and the results are applied to a mobile game of the Hua Rong Dao puzzle.

II. RELATED WORKS

Several researchers have studied computational methods for the Hua Rong Dao puzzle.

LI Ruimin analysises the number of configurations for the basic version of Hua Rong Dao puzzle and presents an improved Depth First Search (DFS) algorithm that can find a solution quickly [3], but the solution found is not guarantted to be the best solution.

LI Yanhui and CAO Zhiguang adopts BFS algorithm to find best solution for Hua Rong Dao puzzle and uses Hash function in duplicate state detection [4] [1]. State pattern equivalence is considered in their algorithm, and CAO's method goes further in that symmetrical states can be detected and heuristic information is introduced in searching. LIN Xuanzhi studies several strategies attached to different state representations when using BFS in finding best solution for Hua Rong Dao puzzle [5].

Most of the above methods are focused on the basic version of Hua Rong Dao puzzle. These methods represent a

state with the positions of every block pieces, therefore they can not handle the variation version, such as the configurations shown in Figure 2. LIN mentions that a state can be represented by the contents of the 4×5 grid, but the duplicate state detection method is not provided.

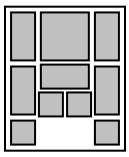
To represent both the basic version and variation version of Hua Rong Dao puzzle, this paper represents a state with the contents of the 4×5 grid. Based on this representation, three conditions are used to detect duplicate states. With the representation and the duplicate state detection method, both basic version and variation version can be solved using BFS.

III. THE HUARONGDAO ALGORITHM

A. State Representation

To represent a state, the box is viewed as a 4×5 grid in which every lattice is filled with a digit. See Figure 3(a). 0 represents nothing in the lattice. 1 represents a 1×1 block piece in the lattice. 2 represents the 2×2 block piece poccessing the lattice. 3~8 represents a 2×1 block piece poccessing the lattice. Therefore, a state is represented by a number with 20 digits.

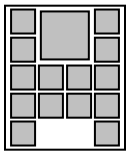
The advantage of this representation is that it can represents the states of variation versions of the puzzle. See Figure 3(b)(c).



3	2	2	4
3	2	2	4
5	6	6	7
5	1	1	7
1	0	0	1

3224322456675117 1001

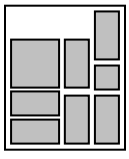
(a) Representation of a Basic Game



1	2	2	1
1	2	2	1
1	1	1	1
1	1	1	1
1	0	0	1

1221122111111111001

(b) Representation of a Variation 1



0	0	0	3
2	2	4	3
2	2	4	1
5	5	7	8
6	6	7	8

00032243224155786678

(c) Representation of a Variation 2

Figure 3. State Representation

B. Procedure

Like the general BFS algorithm, the HuaRongDao algorithm uses two FIFO queues. The states that have not been checked are placed in the OPEN queue and a state is moved to the CLOSE queue once checked. The check operation generates all posible new states of a state by one move and put them into the OPEN queue. All new states are achieved by trying to move every block piece to next position in four possible directions. Before putting a new state into the OPEN queue, pruning should be adopt to ensure the efficiency of the algorithm, which will be discussed in the next section. The working procedure of the HuaRongDao algorithm is shown in Figure 4.

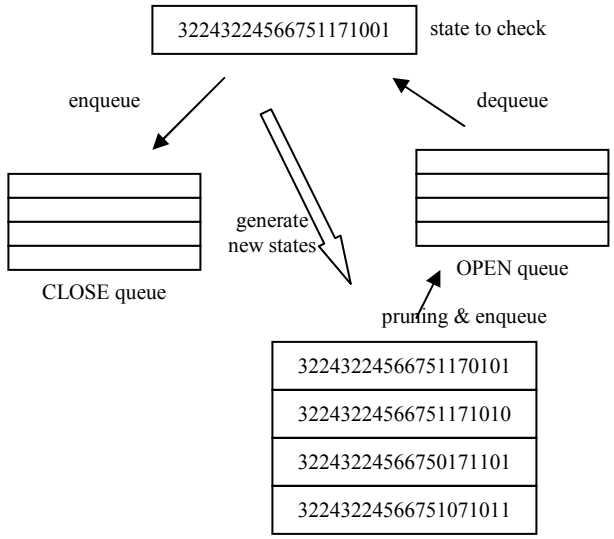


Figure 4. Working Process

At the beginning, only the initial state is placed in the OPEN queue and the CLOSE queue is empty. Then the states in the OPEN queue are checked one by one and placed in the CLOSE queue. This process will continue until an exit state is found, i.e. the block in 2×2 size is moved to the middle bottom position. If the OPEN queue becomes empty before an exit state is found, it can be proved that the initial state has no solution. Figure 5 gives the pseudocode.

```
procedure HuaRongDao:
  create OPEN queue, CLOSE queue;
  put initial configuration into OPEN queue;
  while OPEN queue is not empty:
    dequeue a state from OPEN into s;
    generate new states of s into V;
    for each state t in V:
      if t is an exit state, return SUCCEED;
      if t is not pruned, put t into OPEN;
    put s into CLOSE;
  return FAILED;
```

Figure 5. Pseudocode

C. Pruning

Before puting a new state into the OPEN queue, it should be checked if the OPEN queue or the CLOSE queue already contains a state with the same patten. If a state pattern is found in the CLOSE queue, its further branches have already been generated. If a state pattern is found in the OPEN queue, it has been reached by another branch. Under these two circumstances, the state needn't be put into the OPEN queue.

This check for pruning can avoid infinite loop and improve the efficiency of the algorithm.

To determine whether two states are in the same pattern, the following conditions should be tested:

1) the block in 2×2 size is located at the same position.

2) the blocks in 1×1 size are located at the same position correspondingly.

3) the blocks in 2×1 size can be changed to the same position by a series of position swapping.

Condition 1) and 2) can be tested directly, but condition 3) needs digits swapping first. For the example in Figure 6, state 1, 2, 3 are in the same pattern. The number representing state 2 will be as same as state 1 if swapping digit 3 and 4. State 3 needs a much complicated swapping, i.e. 4→3, 5→4, 6→5, 7→6, 3→7.

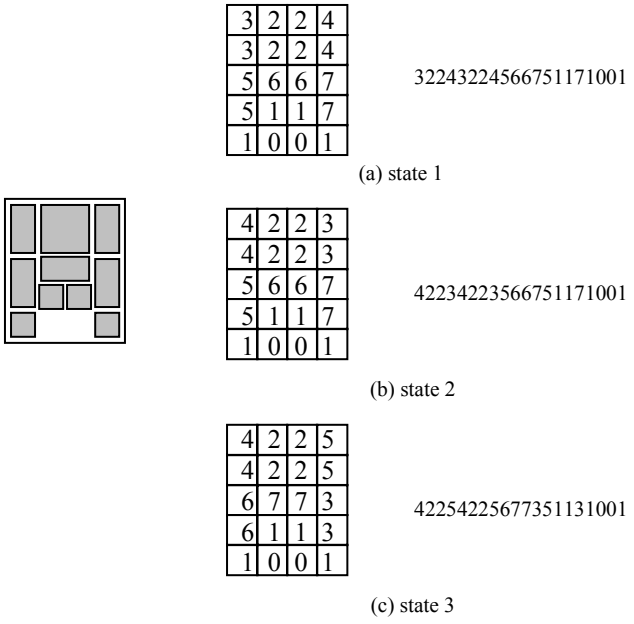


Figure 6. State pattern

To ensure the first found solution is the best one, the states in the OPEN queue should be ordered by the number of moves needed from the initial state. Usually, the number of moves needed to reach a state is larger than its previous state by 1, but there is exception. If a state and its previous state are reached by moving the same block, the number of moves needed to reach these two states are equal. For the example shown in Figure 7, state 1 is the initial state, thereby its number of moves is 0. State 2 is achieved from state 1 by moving the black block to the right, thereby its number of moves is 1. State 3 is achieved from state 2 by moving the black block to the right, but its number of moves is still 1 because state 3 and state 2 are achieved by moving the same block. State 4 is achieved from state 2 by moving another block, so its number of moves is 2. Both state 3 and 4 are generated from state 2, but state 3 should be closer to the header of the queue when putting them into the OPEN queue because its number of moves is smaller. The position of state 3 in the queue can guarrantee it checked before state 4, thereby the best solution will be found first.

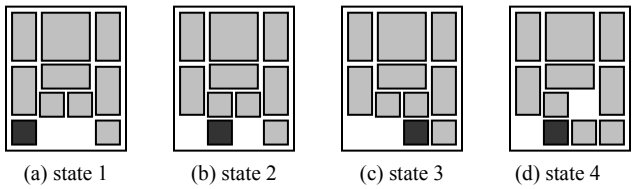


Figure 7. Counting number of moves

Therefore, to count the number of moves for every state, the block moved and to which direction should also be recorded. This information will also be used to compose the detail solution.

D. Solution composing

After an exit state is found, a backward traverse from the exit state to the initial state should be performed to compose the detail solution. Therefore, every state should have a pointer to its previous state.

As a summary, in the OPEN queue or CLOSE queue, a state structure comprises of five parts: state number, number of moves, block moved, direction and pointer to previous state. See Figure 8. The block moved is represented by its position.

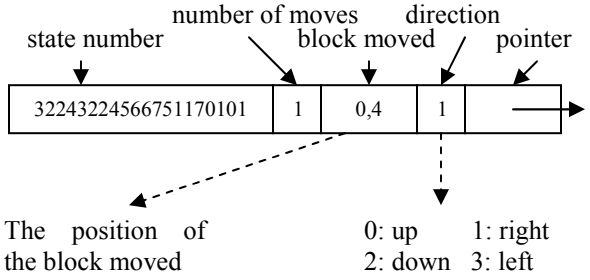


Figure 8. State structure

In the traverse process, the information of every move is inserted into a list from back to front. The list of moves is the detail steps of the best solution. See Figure 9.

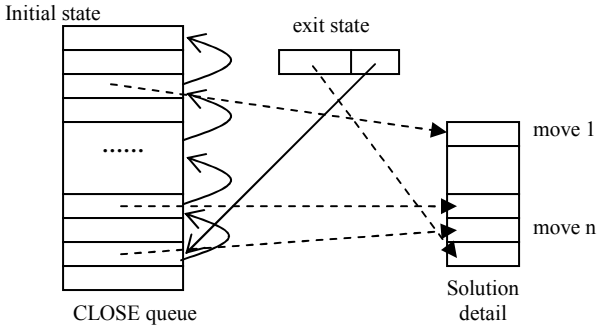


Figure 9. Result composing

IV. IMPLEMENTATION AND APPLICATION

A. Implementation

The algorithm has been implemented on the Java SE platform. A window shows the searching process of the algorithm. See Figure 10. The initial state is entered in the textbox in the bar at the top, and then the searching process will be illustrated step by step. The left list shows the states in the CLOSE queue and the right list shows the OPEN queue. In a working circle, the first state in the OPEN queue is checked and moved to the CLOSE queue, and new generated states are put into the OPEN queue. Every state can be shown as a picture.

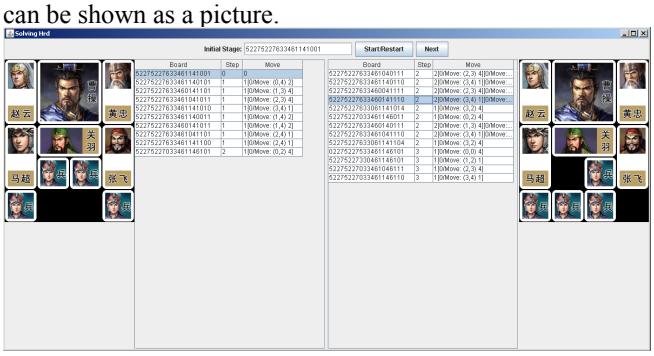


Figure 10. Searching process window

In the searching process for a state, the size of the CLOSE queue and the OPEN queue are recorded after every layer is finished in the searching tree. Layer n contains all the states that need n moves from the initial state. The size of the CLOSE queue increases continuely, but the size of the OPEN queue will begin to decrease after reaching a certain size. Table I lists the size of the two queues after every 10 layers.

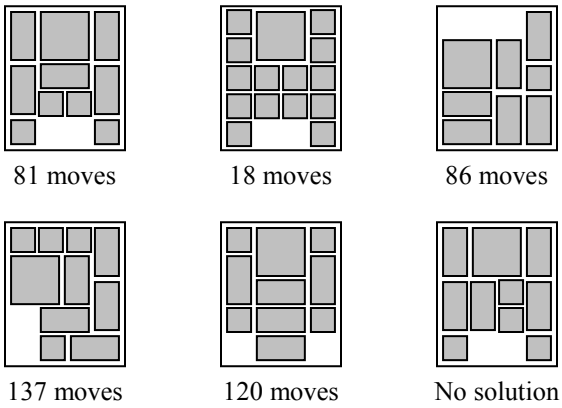


Figure 11. Results

B. Results

374 initial configurations have been searched using the algorithm. Parts of the results are shown in Figure 11. The basic configuration needs 81 moves at least. This result can be verified by the study of many other researchers. The easist configuration needs 18 moves, while the most difficult one needs 137 moves. In addition, there is an configuration having no solution.

TABLE I. SIZE CHANGING OF THE CLOSE/OPEN QUEUES

Number of moves	Size of CLOSE queue	Size of OPEN queue
1	2	3
11	256	41
21	820	43
31	2292	314
41	9190	959
51	16288	229
61	17998	141
71	20922	332

V. CONCLUSION AND FUTURE WORKS

A Hua Rong Dao algorithm is proposed in this paper, which can find best solutions for both basic version and variation version of the puzzle. A state is represented with the contents of the 4×5 grid, so that states of variation version can be also represented. Three conditions are used to detect duplicate states in the BFS. The algorithm has been implemented on Java SE platform and applied to find the best solutions of 374 configurations. The results verified the correctness and the effectiveness of the algorithm.

The algorithm can be improved in several aspects. First, symmetrical state patterns should be detected in the pruning. Second, heuristic information should be used when selecting a state to check from the OPEN queue. Third, checked state patterns should be reused among all configurations. The improvements in these aspects can bring further advance on the effectiveness of the algorithm.

ACKNOWLEDGMENT

The Project Sponsored by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry and Scientific Research Foundation for Doctors, Shandong University of Finance.

REFERENCES

[1] CAO Zhiguang, Breadth-First Search of Hua Rong Dao:Application of Hash Search and Heuristic Search, Journal of Electric Power, 2005, 20(1), pp13-16, (in Chinese).

[2] Elwyn R. Berlekamp, John H. Conway, Richard K. Guy, Winning Ways for Your Mathematical Plays, 2nd edition, A K Peters/CRC Press, ISBN-13 978-1568811307, 2001.

[3] LI Ruimin, JIANG Changjun, Research and implementation of Chinese Game(Hua Rongdao). Computer Engineering and Applications, 2007, 43(13): 108-110, (in Chinese).

[4] LI Yanhui, LI Aijun, Method for Hua Rongdao based on Breadth-First Search, Computer Systems & Applications, 2010, 19(11), pp222-225, (in Chinese).

[5] LIN Xuanzhi, Search Strategy for Optimal Solution of HuaRong Road, Journal of Zhangzhou Teachers College(Natural Science), 2003, 16(4), pp36-41, (in Chinese).

[6] Russell S., Norvig P. Artificial Intelligence–a modern approach, 2nd Edition, Prentice Hall, ISBN 978-0137903955, 2003.

[7] Solving HuaRongDao website, http://fayaa.com/youxi/hrd/.