

An efficient algorithm for solving Fillomino

Shi-Jim Yen

Department of Computer Science &
Information Engineering,
National Dong Hwa University,
Hualien, Taiwan, R.O.C.
sjyen@mail.ndhu.edu.tw

Tsan-Cheng Su

Department of Computer Science &
Information Engineering,
National Dong Hwa University,
Hualien, Taiwan, R.O.C.
d9821009@ems.ndhu.edu.tw

Shun-Chin Hsu

Department of Information
Management,
Chang Jung Christian University,
Tainan, Taiwan, R.O.C.
schsu@mail.cjcu.edu.tw

Abstract—Fillomino is a logical puzzle game invented by Nikoli Company in Japan. Fillomino is played on a rectangular cell with no standard size; the internal cell lines are often dotted. Some cells of the cell start containing numbers, referred to as "givens". In this paper, we propose a puzzle solving algorithm to treat these problems. Based on the fact the Fillomino are compact and contiguous, some logical rules are deduced to paint some cells. Experimental results show that our algorithm can solve Fillominos successfully and efficient, and the processing speed is significantly faster than that of depth first search algorithm.

Keywords- Fillomino, Puzzle Game, CSP.

I. INTRODUCTION

Recently, many puzzle games popular in world. These puzzle games of Rules are usually simple, but solving them needs excellent logic concept. With contemplation, it is possible to finish these games, such as Sudoku, Nurikabe, Number Link, and some puzzle games.

Because of the popularity of puzzle games, many related studies become very popular. There are some directions for these studies. Some put focus on the generation of questions, trying to use the least implied numbers to generate a question with only one answer. Some try to demonstrate some puzzle games are NP-Complete problems [2][8]; besides, studying how to use programs to rapidly solve these puzzle games is an important direction. Previous researchers tried to use pattern match and DFS to solve Nonograms puzzle games [2][5], and had breakthrough in the speed of solving them. However, DFS may step into a wrong track, stuck in it, and cause great deal of resource waste.

In the second section, we will introduce others to the methods to solve puzzle game and introduce rules of fillomino in details in third section. The major method used in this research in the fourth section, including intersection method and logic rules. Experimental results will be presented in the fifth section, and the sixth section is conclusions.

II. PREVIOUS WORKS

Yen[6][7] they used Logical Reasoning and Heuristic to solve Nurikabe and Lights-up, and they can quite efficiency to solve puzzle. Their experimental results show that they can solve successfully and high-speed.

Wiggers [10] proposed a genetic algorithm (GA) and a depth first search (DFS) algorithm to solve Japanese puzzles. In 2004, he compared the performance of these two algorithms. For a small puzzle, the DFS algorithm is faster than the GA; otherwise, GA is faster. However, both methods are slow. In this study, some logical rules will be applied to determine the state filled of as many cells in a Japanese puzzle, then the DFS algorithm with "branch and bound" is used to determine the state of the remaining cells.

Jing and Yu [2][5] proposed a method based on logical rules (LR) and chronological backtracking with an LR filter to solve Japanese puzzles. Their rules exploited with a new concept of "the range of a black run." These rules have three main parts. The first part is used to determine which cells should be colored or left empty; the second part is used to refine the ranges of black runs, and that is used to determine which cells should be colored or left empty and to refine the ranges of black runs. One thing should be mentioned at first, we use row information to build a DFS tree, and the column information is used immediately to verify when a row is used to create a layer. It means that every layer of the DFS tree is composed of row information, and all nodes in each layer are possible solutions (PS) for the corresponding row.

Batenburg [3] described an evolutionary algorithm for Discrete Tomography (DT) then Batenburg and Kesters [4] provided a method to solve Nonogram puzzle. By modifying the fitness function in [3], the evolutionary algorithm can be used to solve Japanese puzzle. Since the evolutionary algorithm in [3] will converge to a local optimum, the obtained solution maybe incorrect.

III. GAME RULES

This Puzzle Game rule below:

A. *Clues i said, suggesting that the number of i, the number of cell connected.*

The rules need to be shown to satisfy the number of questions on a puzzle, if the clues is "i", that suggest you must have the same number i of connected cell. Like figure 8.

B. *In the game have $N * M$ digits to fill all $N * M$ cells.*

This rule needs to $N * M$ of digits, all cells filled with digits in the $N * M$ of Cells. Like figure 8.

C. All of the cells must meet to be number of block.

This rule is necessary to have filled digits all the cells and the number of connection cells must meet the clues number.

D. The same number of blocks can not be connected.

This prompted rule is the same clues number of cells block can not be connected. However, the different clues number of cells block can be connected.

IV. PROPOSED METHOD

The algorithm of our solver contains three phases. The first phase and second phase are logic rule. The third phase is Intersection. The first phase checked point which has only one way to connection with the other cell. If we determine the first one is true, we can link it with the cell which one way to connection. The next phase inspection one cell which only one cell link to other cell, that we must link it. The last phase intersection all the possible answer matches of every cell. If sure one cell belong which one cell, We immediately fill this cell that should belong to the number of cell. Figure 1 that is our algorithm of flow chart.

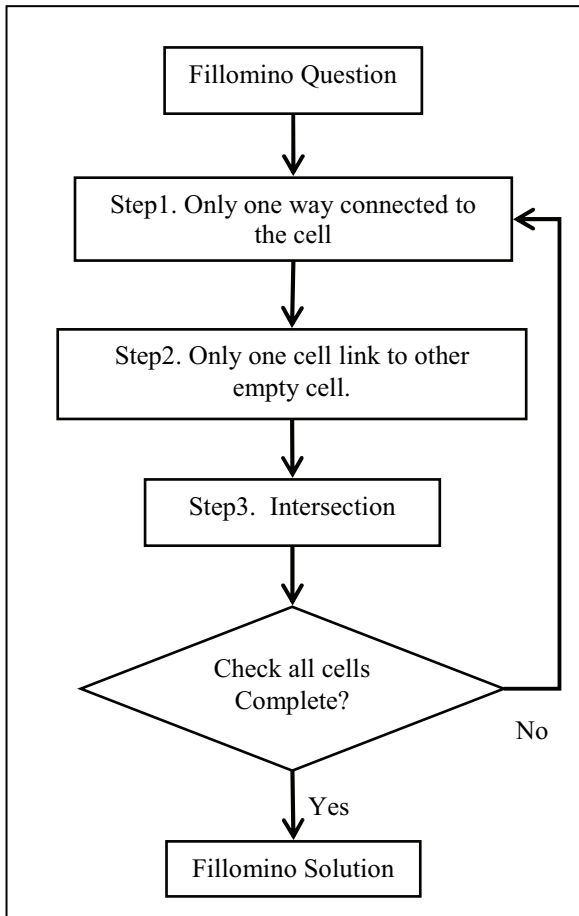


Figure 1. Flow chart of Fillomino.

A. Step1: Only one way connection

One cell that only has one way to connection other cell, that cell must be fill the only other cell numeral. Like figure 2, we marked the cell to fill 「4」.

	1	
4	2	2
		3

Figure 2. A simple step1.

B. Step2: Only one cell link to other empty cell

We used solid lines mark the cell, which is only one cell link to other cell. If we don't link it we will violation of the rules, and the cell no any numeral to fill it. Like figure 3, we used solid line mark the cell and dotted line mark the cells, these cells must to fill 「6」.

	5	6
		1

Figure 3. A simple step2.

C. Step3: Intersection

We used Intersection in one cycle of last. Figure 4 is a simple in corner. We do first and second phase, we create all of numeral possibilities pattern for the intersection. Like figure 5.

We intersection again to find intersect of cell. We find Intersect of cell to fill the numeral. Like figure 6. The cell marked that we can fill 「4」.

4		1
2	1	3

Figure 4. A simple puzzle Part of the corner.

We used three phases repeat continuously until the end of this puzzle. We can efficiently and successfully to solve the puzzle game. Figure 7 and Figure 8 are a simple question and solution.

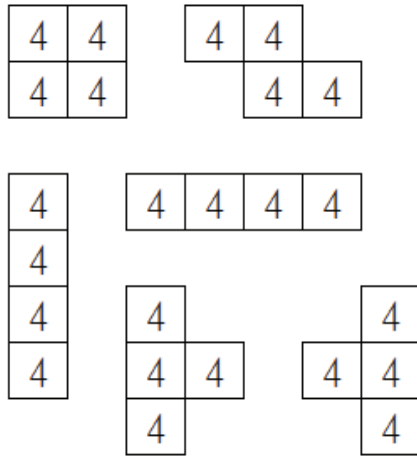


Figure 5. All of numeral possibilities pattern of Fig 4.

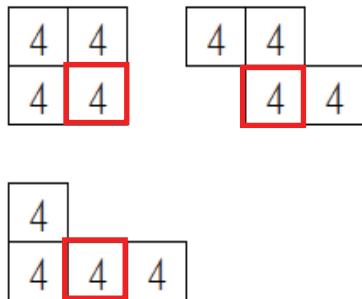


Figure 6. The pattern Intersect of cell.

	1					
4	2	2			5	6
		3				1
6			4			4
4				4		
3	4			5	1	4
					5	

Figure 7. Simple question of Fillomino.

4	1	6	6	6	6	6
4	2	2	3	5	5	6
4	4	3	3	4	5	1
6	6	6	4	4	5	4
4	4	6	6	4	5	4
3	4	4	6	5	1	4
3	3	5	5	5	5	4

Figure 8. Simple solution of Fillomino

```

Input puzzle of question Q
Do
{
Function1 (Only one way connected to the cell(Q));
{ Output Q1; }
Function2(Only one cell link to other empty cell(Q2));
{Output Q3}
Do Intersection(Q3);
{Output Q4; }
Function3(Check All call complete(Q4));
{
Return True or False;
}
}while(all cells Complete=True)
If ( Function3=True )
{Output the puzzle answer=Q4; }

```

Figure 9. our algorithm by pseudo code.

Figure 9: We show our algorithm by pseudo code. Our method is quick to find the answer and our execution time spends very little.

V. EXPERIMENTAL RESULTS

We found 100 of 4 type sizes Fillomino puzzles on [1] to verify the validity of our method. Our Algorithm by Microsoft Visual Studio C# 2008 of program on the computer with CPU of Intel Core 2 Quad Q9550 by 1 thread, 8 GB main memory and Windows Vista 64-bit OS. Table I shows the comparison of our proposed algorithm, Genetic Algorithms (GA) and CSP. Our proposed algorithm takes 2 seconds per question in average. Compared with GA and CSP, our solver is quite efficiency.

	1					
4	2	2			5	6
		3				1
6			4			4
4				4		
3	4			5	1	4
					5	

(a)

4	1	6	6	6	6	6
4	2	2	3	5	5	6
4	4	3	3	4	5	1
6	6	6	4	4	5	4
4	4	6	6	4	5	4
3	4	4	6	5	1	4
3	3	5	5	5	5	4

(b)

				5	5		5	5
4	5		5	3			1	
	3				2	4		4
	1			2	1			
							6	4
6		2						2
				6	4		1	4
	3		4	2			1	
					4	3		3
6	6			2	3		2	

(c)

4	4	4	1	5	5	5	1	5	5
4	5	5	5	3	2	5	5	1	5
3	3	5	3	3	2	4	2	4	5
3	1	5	6	2	1	4	2	4	5
6	6	6	6	2	4	4	6	4	4
6	3	2	2	6	6	6	6	2	2
4	3	6	1	6	4	4	1	4	4
4	3	6	4	2	2	4	4	1	4
4	4	6	4	4	4	3	2	3	4
6	6	6	2	2	3	3	2	3	3

(d)

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

(e)

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

(f)

Figure 10. Testing Puzzle of Question and Answer.

Figure 10: In a comparative experiment many puzzles were solved quickly and completely using this paper of Method. The average execution time was less than about 2 second. Fig. 10 shows the experimental results concerned the testing puzzle of Question and Answer. Fig.10 (a), Fig.10 (c) and Fig.10 (e) are puzzle of Questions. Fig.10 (b), Fig.10 (d) and Fig.10 (f) are solved Fig.10 (a), Fig(c) and Fig.10 (e) by order.

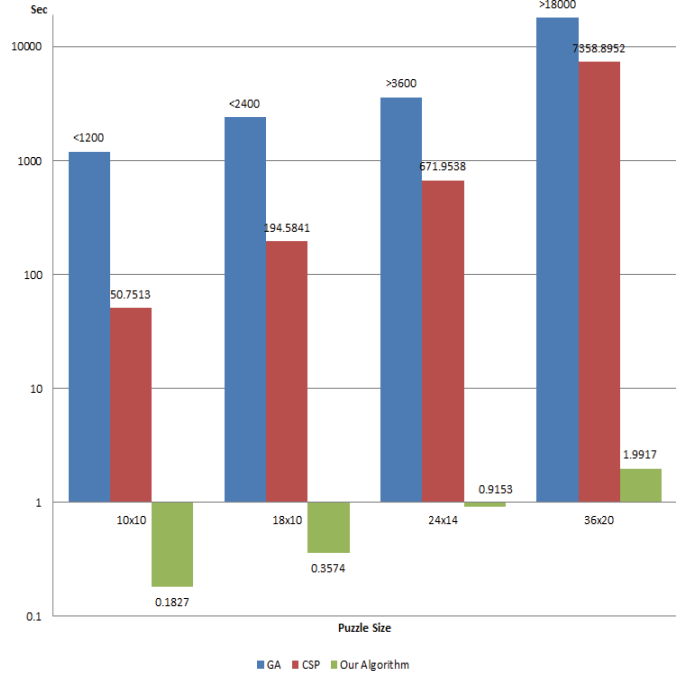


Figure 11. The comparison of the experimental results.

Figure 11: This paper comparison with GA and CSP to solve the Fillomino puzzle. This Figure show this paper method used execution time spend very little.

VI. EXPERIMENTAL

This research studies how to quickly solve Fillomino. This research tries to use intersection method to solve this question, and proves that this method can quickly and effectively solve Fillomino. According to the experiment results, we proved our algorithm than the method using the GA or the CSP more quickly and efficiently. Our algorithm we use is still a drawback to be improved. In Step3 to create all possible patterns, requires a lot of memory resources. This needs to be improved. At the same time, because the three phases do not need much domain knowledge basically, we can also try to apply them to other puzzle games and expect there will be great results.

ACKNOWLEDGMENT

This work is supported by the National Science Council of Taiwan under the grant NSC 98-2221-E-259-021-MY3 and NSC 99-2221-E-259-009-MY3.

TABLE I. TABLE TYPE STYLES

Puzzle Size : Piece	Average Execution time		
	GA	CSP	Our Algorithm
10x10:15	<20 min	50.7513 sec	0.1827 sec
18x10:30	<40 min	194.5841 sec	0.3574 sec
24x14:35	>1 hr	671.9538 sec	0.9153 sec
36x20:20	>5 hr	7358.8952sec	1.9917 sec

REFERENCES

- [1] Database of Japanese puzzles : Nicoli.2008. Fillomino 4thed,Japan: Nikoli Co., Ltd..
- [2] C. H. Yu, H. L. Lee, and L. H. Chen , "An efficient algorithm for solving nonograms," Applied Intelligence , pp. 1-14, 13 Nov. 2009..
- [3] K. J. Batenburg, "An Evolutionary Algorithm for Discrete Tomography," M.Sc. thesis in Computer Science, University of Leiden, The Netherlands, 2003.
- [4] K. J. Batenburg and W. A. Kosters, "A Discrete Tomography Approach to Japanese Puzzles," Proceedings of BNAIC, pp. 243-250, 2004.
- [5] M. Q. Jing, C. H. Yu, H. L. Lee and L. H. Chen, "Solving Japanese Puzzles with Logical Rules and Depth First Search Algorithm," Proceedings of the Eighth International Conference on Machine Learning and Cybernetics, pp. 2962-2967, Jul. 2009.
- [6] S. J. Yen, T. D. Chuang, S. Y. Chiu and J. C. Chen, "Nurikabe, Heuristic Search, Puzzle," the 14th Game Programming Workshop (GPW-09), pp. 83-86, Nov. 13-15, 2009.
- [7] S. J. Yen, S. Y. Chiu , "A Simple and Rapid Lights-up Solver," the 14th Game Programming Workshop (GPW-09), pp. 67-70, Nov. 13-15, 2009.
- [8] B. P. McPhail, "Light Up is NP-Complete," Feb. 2005.URL: <http://www.reed.edu/~mcphailb/lightup.pdf>.
- [9] Ueda N, Nagao T (1996) NP-completeness results for NONOGRAM via parsimonious reductions. Technical report TR96-0008, Department of Computer Science, Tokyo Institute of echnology, May 1996
- [10] Gary McGuire, "Sudoku checker and the minimum number of clues problem," In <http://www.math.ie/checker.html>.