Solving Fillomino with Efficient Algorithm

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, Intersection, CSP.

1. 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][6]; 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][3], 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 chapter, we will introduce others to the methods to solve puzzle game and introduce rules of fillomino in details in third chapter. The major method used in this research in the fourth chapter, including intersection method and logic rules. Experimental results will be presented in the fifth chapter, and the sixth chapter is conclusions.

2. Previous works

Yu and Jing [2][3] they used some logical rules are deduced to paint some cells. Then, they used the chronological backtracking algorithm to solve those undetermined cells and logical rules to improve the search efficiently. Experimental results show that they can solve nonogram puzzles successfully

Yen[4][5] 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.

3. Game rules

3.1 Clues i said, suggesting that the number of i, the number of cell connected like figure 8.

3.2 In the game have N * M digits to fill all N * M cell like figure 8.

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

3.4 The same number of blocks can not be connected.

4. 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.

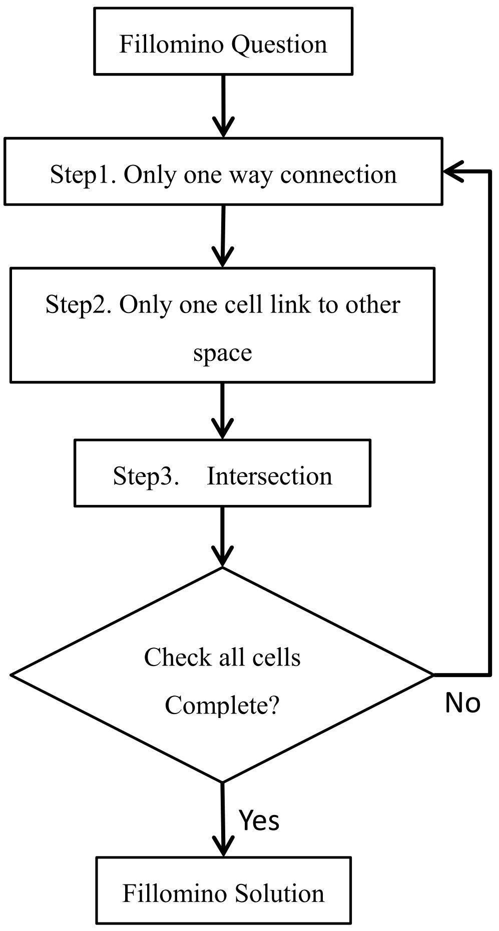

Fillomino Question
Step1. Only one way connection
Step2. Only one cell link to other space
Step3. Intersection
Check all cells Complete?
No
Yes
Fillomino Solution

Figure 1. Flow chart of Fillomino.

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」.

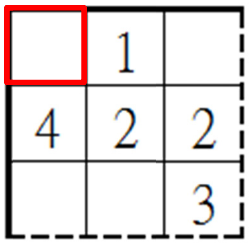


Figure 2. A simple step1.

Step2. Only one cell link to other 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」

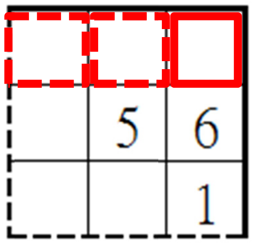


Figure 3. A simple step2.

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」.

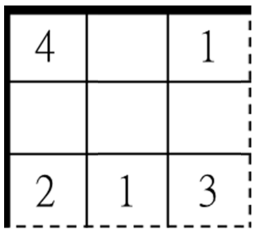


Figure 4. A simple.

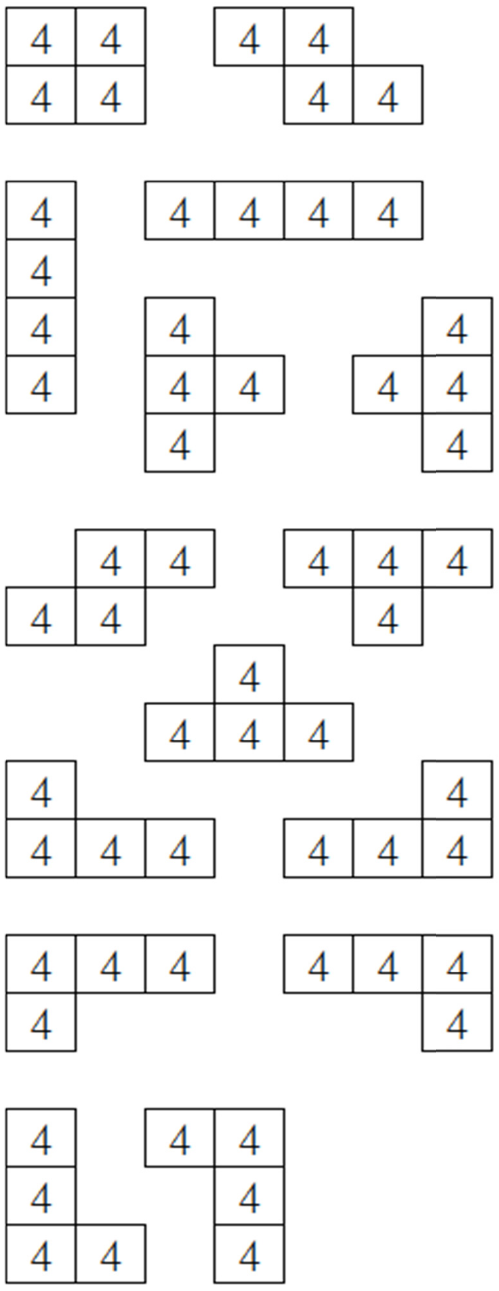


Figure 5. All of numeral possibilities pattern.

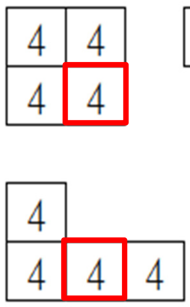
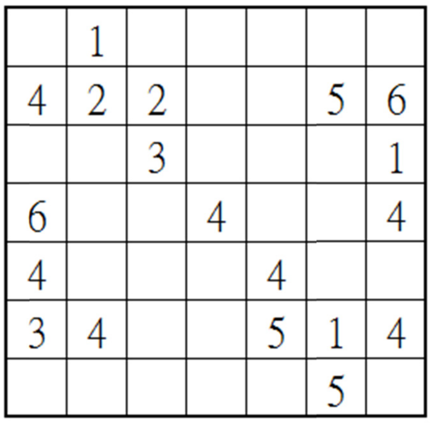


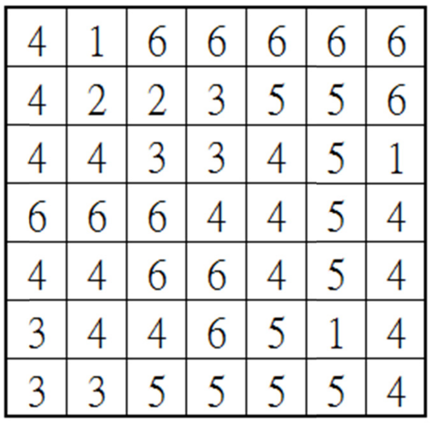
Figure 6. The pattern Intersect of cell.

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.



	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

5. 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 1 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.

Table 1. Experimental Result.

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

6. Conclusions

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.

Reference

[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 , 13 November, 2009. pp. 1-14.

[3] M. Q. Jing, C. H. Yu, H. L. Lee and L. H. Chen, "Solving Japanese Puzzles with Logical Rules and Depth First Search Algorithm," International conference on Machine Learning and Cybernetics 2009, Baoding China, 12-15 July 2009.

[4] S. J. Yen, T. D. Chuang, S. Y. Chiu and J. C. Chen, "Nurikabe, Heuristic Search, Puzzle," the 14th Game Programming Workshop (GPW-09), November 13-15, 2009, Hakone Seminar House, Kanagawa, Japan: Proceeding pp. 83-86.

[5] S. J. Yen, S. Y. Chiu ,"A Simple and Rapid Lights-up Solver," the 14th Game Programming Workshop (GPW-09), November 13-15, 2009, Hakone Seminar House, Kanagawa, Japan: Proceeding pp. 67-70.

[6] B. P. McPhail, “Light Up is NP-Complete,” Feb. 2005.URL: http://www.reed.edu/~mcphailb/lightup.pdf.