Solving Nonogram Puzzle Using Backtracking Algorithm

Christine Hutabarat - 13520005
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail : 13520005@std.stei.itb.ac.id

***Abstract*—Nonogram is a logic drawing puzzle in which the player needs to fill cells in a grid with black or white according to the description given to the corresponding columns and rows. Backtracking algorithm is a problem-solving technique that considers only the choices leading to the solution, pruning all other invalid states. This paper discusses how backtracking algorithm can be used to solve nonogram puzzles by systematically turning white grids to black then immediately going back to the previous state and choosing a different step as the latest row/column state violates the rule given to it. The problem is modeled using both mathematical expressions and regular expressions. The backtracking method has proven to be efficient for solving nonogram puzzles by significantly reducing the search space.**

Keywords—nonogram; puzzle; backtracking algorithm; regular expression

I. INTRODUCTION

Nonogram or Japanese puzzle is a logic drawing puzzle consisting of a pixel grid in which each of the pixels should be colored according to the rule given to its corresponding column and row. The nonogram grid has a rectangle shape, but the size of the grid varies as there is no restriction to it. Outside of the grid, next to each column or row border, there is a set of integers which describes how many and how long separated lines should be drawn in that column or row. The final state of the colored grid usually creates a pixel image.

Fig. 1 and Fig. 2 give an example of a nonogram puzzle with the size of 25x25. Fig. 1 shows the unsolved state of the puzzle, and Fig. 2 shows the solved state, or the solution of the puzzle. The numbers above and next to the grid are the puzzle rules. For example, the number 1 above the leftmost column indicates that there should be one black line of the same length as the length of one cell in that column. The set of numbers 7 and 3 next to the second uppermost row tells us that there should be two black lines separated by one or more white cells, in which the first or the leftmost line has a length of 7 and the second line has a length of 3. By following the rules, a hidden pixel image of a dolphin is revealed.

Most nonogram puzzles can be solved even by relying on the rules given, with the help of logic or common sense. Fujiwara in [4] categorized and explained some of these common senses. Yu, Lee, and Chen in [3] expressed the logical rules in mathematical equations and also applied them to a computer program that is designed to solve nonogram puzzles.

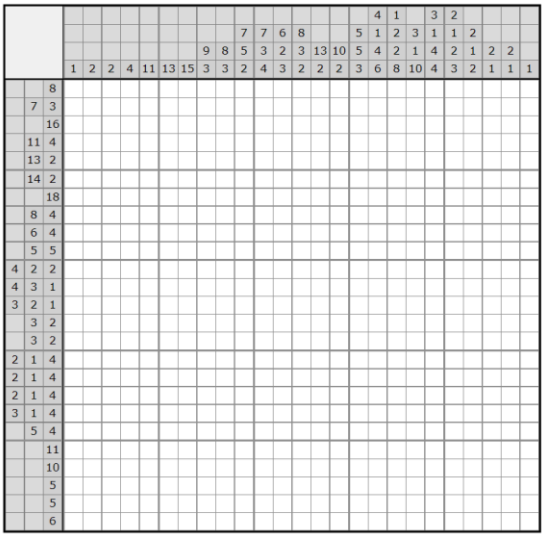

Fig. 1. An example of unsolved nonogram puzzle [6]

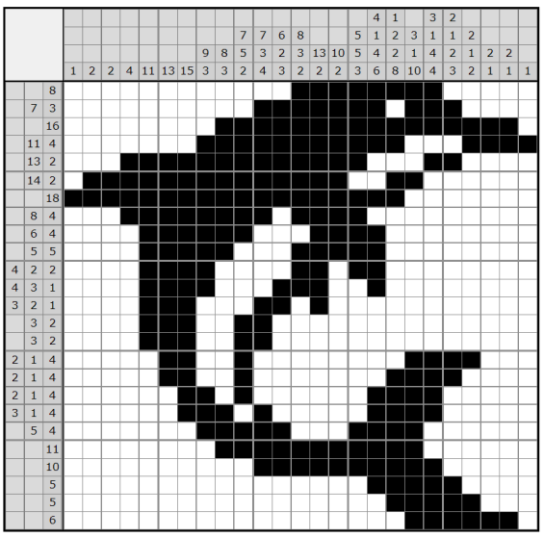

Fig. 2. An example of solved nonogram puzzle [6]

Although the mere use of logical rules yields correct results most of the time, some puzzles take a lot of time to solve with this method. To improve efficiency, logical rules are used along with backtracking algorithm.

The backtracking algorithm is a problem-solving technique similar to the exhaustive search algorithm that can be represented as a tree. However, this technique is more efficient than exhaustive search since the key point of this algorithm is to prune branches that are not leading to the solution, thus reducing the search space. Therefore, the backtracking algorithm is considered the improved version of exhaustive search.

In the backtracking algorithm, the rules or constraints for the final state are given in the bounding function. Starting from the root node, in every step taken, if the current state does not satisfy the constraints, then go back one step and choose another path, generating a new node in the tree. Go back to the previous state if there is no other option available. These steps are repeated recursively until the solution state is found or all possible solutions have been checked. Having all the states generated in the tree would be the worst-case scenario of the backtracking algorithm. The best-case scenario for this algorithm is when the starting state is also the final state.

Since the choices made in the backtracking algorithm depend heavily on the rules for the solution, it becomes easy to deduce that backtracking is suitable for solving the nonogram puzzle. Yu, Lee, and Chen proposed an algorithm which makes use of backtracking and eleven logical rules to solve nonogram puzzles in [3]. A similar idea will be analyzed in this paper. However, the rules used are not specific as they are simplified in many ways. The following section explains how these rules are modeled and generated.

II. Modeling the Problem

To solve the nonogram puzzle with backtracking algorithm, the bounding function needs to be determined. This can be done by converting the set of rules given in the puzzle into more specific functions. But prior to doing that, in order to simplify the method, let the initial state of the puzzle be a grid whose cells are all painted in white. Therefore, the method that is going to be discussed will change the state by turning the white grid to black. This way, backtracking can also be seen as painting the cells in a column or row that violate the rules back to white.

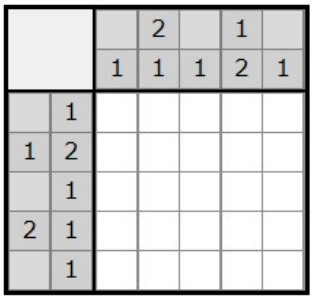

Fig. 3. An example of a 5x5 nonogram puzzle

To define the problem formally, the parameters described in Table 1 will be used for further steps of modeling and calculation.

TABLE I. List of Parameters

Parameter	Description
m	The length or the number of columns in the grid
n	The width or the number of rows in the grid
c_i	The i-th column (index of column increasing from left to right, starting from 1)
r_i	The i-th row (index of row increasing from top to bottom, starting from 1)
w_x	The amount of white cells present in column/row x
b_x	The amount of black cells present in column/row x
br_x	The amount of required black cells in column/row x
wr_x	The amount of required white cells in column/row x

From Fig. 3, it is known that for each row or column, $b_x = 0$ and $w_x = 5$. It can also be seen that m = 5 and n = 5. This means that for each row and each column,

$$br_x + wr_x = 5 \quad (1)$$

Now pay attention to the rules for the first column. The sum of the numbers in the set is the value of b, so we have $br_{c1} = 1$. By assigning this value to Eq. (1), we have $wr_{c1} = 4$. This will be the first constraint for the first column. As another example, look at the second row. The sum of numbers is 3, so $br_{r2} = 3$. By assigning $br_{r2} = 3$ to Eq. (1), we know that $wr_{r2} = 2$. Apply this rule to get the first constraint for each column and row.

TABLE II. Cell Amount Rule for Rows

Index of row (i)	br_{ri}	wr_{ri}
1	1	4
2	3	2
3	1	4
4	3	2
5	1	4

TABLE III. Cell Amount Rule for Columns

Index of column (i)	br_{ci}	wr_{ci}
1	1	4
2	3	2
3	1	4
4	3	2
5	1	4

A regular expression will be used to define the pattern of black and white cells in every column or row. For example, look at the rules given to the first row. There should be only one cell-long black line. This line can be drawn at the leftmost side of the row, at the rightmost side of the row, or in between. Now let B represent a black cell and W represent a white cell. Using regular expression, the pattern of cells in the first row can be written as

W*BW*

The pattern would mean: zero or more white cells, followed by one black cell, followed by zero or more white cells.

Let us now try to formally define the pattern for the second column. The pattern would be expressed as

W*B{2}W+BW*

Explained in words, the pattern would mean: zero or more white cells, followed by exactly two black cells, followed by one or more white cells, followed by one black cell, followed by zero or more white cells. This expression will be the second constraint for the third column. Apply this to every column and row to get the cell pattern.

Using the new derived constraints stated in Table II, Table III, Table IV, and Table V, the puzzle can be solved with backtracking algorithm with these rules as the bounding functions. The constraints are grouped according to their corresponding columns or rows in Table VI.

TABLE IV. PATTERN RULE FOR ROWS

Index of row	Pattern
1	W*BW*
2	W*BW+B{2}W*
3	W*BW*
4	W*B{2}W+BW*
5	W*BW*

TABLE V. PATTERN RULE FOR COLUMNS

Index of column	Pattern
1	W*BW*
2	W*B{2}W+BW*
3	W*BW*
4	W*BW+B{2}W*
5	W*BW*

TABLE VI. CONSTRAINTS FOR COLUMNS AND ROWS

Column/Row (x)	br_x	wr_x	Pattern
r_1	1	4	W*BW*
r_2	3	2	W*BW+B{2}W*
r_3	1	4	W*BW*
r_4	3	2	W*B{2}W+BW*
r_5	1	4	W*BW*
c_1	1	4	W*BW*
c_2	3	2	W*B{2}W+BW*
c_3	1	4	W*BW*
c_4	3	2	W*BW+B{2}W*
c_5	1	4	W*BW*

To simplify the algorithm, every step taken would fill the current row with the valid pattern if the method is done row-wise. This means that in every iteration the columns might be in three possible states: solved, violated, or not violated but not solved, and the pattern rules for the rows are strictly enforced. The solved state means that the configuration of black and white cells in the corresponding column satisfies all of its rules. Since we initially painted all cells white and we will paint some of them black through the algorithm, we can consider a column not violated but not solved if it does not satisfy the pattern rule, but the number of black cells painted already in the column is no more than the black cells required. The expression can be written mathematically as:

$$b_{ci} \leq br_{ci} \quad (2)$$

With this being said, a column will be considered violated if its cell configuration does not satisfy the pattern rule, as well as the rule expressed in Eq. 2.

The same thing goes for the puzzle solved column-wise. If it is done column-wise, then in each step the current column will be filled with one of its possible patterns, and each row will have three possible states: solved, violated, not violated but not solved, as explained in the previous paragraph. The extra rule for determining whether a row is not violated but not solved will be

$$b_{ri} \leq br_{ri} \quad (3)$$

In this approach, the pattern rules for columns will be strictly enforced in each step, and a row will be considered as violated if the cell configuration does not satisfy the pattern rule and Eq. 3.

III. SOLVING THE PROBLEM

The initial state of the puzzle is a 5x5 grid, as shown in Fig. 3. The final state of the puzzle will be the state when every column and row rule is in the solved state. In this sample case, a path to the final state is guaranteed to exist. We will be using the column-wise approach to solve this puzzle.

Let us look at the first steps taken in solving this puzzle. The first column requires only one black cell, and it can be anywhere in the column. The first possible cell configuration for this rule is shown in Fig. 4.

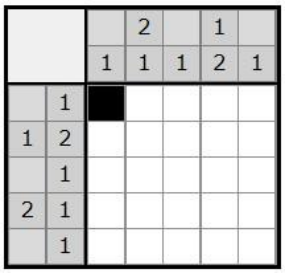

Fig. 4. A new state generated from the initial state

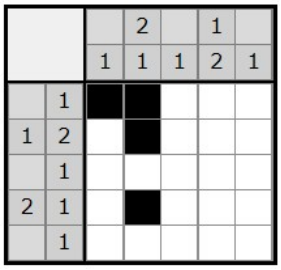

Fig. 5. A new state generated from the state shown in Fig. 4

Notice that the pattern of cells in the first row is not violated, and the first row can even be considered solved. Since there is no rule violated so far, we continue iterating to the second column. The second column requires three black cells and two white cells, creating the pattern W*B{2}W+BW*. The first possible configuration satisfying these rules is shown in Fig. 5.

Pay attention to the first row. It now has two black cells and three white cells. Since $2 > br_1$, we can say that the first-row rule is violated. This means that we need to go back one step and check on another possible pattern for the second column. Remember that our previous state is shown in Fig. 4.

The method used to generate the combination of cells in a column or row is not discussed in this paper. However, we will use one of the common senses to solve nonogram puzzle suggested by Fujiwara in [4]. This is done just so we can search for the next pattern systematically. We can see that in the state shown in Fig. 5, the second column has black cells on its edge. By applying the black-edge common sense, the following sequence of black cells can be extended towards the bottom edge. Thus, revealing the next possible pattern of this puzzle. The state mentioned is shown in Fig. 6.

The latest state generated also violates the first row rule from Eq. 3, so backtracking needs to be done again. In the current state, black-edge common sense cannot be used. However, notice that there is another possible pattern for the second column that can be found by moving the first black line one cell towards the bottom edge. The result can be seen in Fig. 7.

Fig. 7. displays the next pattern for the second column. Notice that the first row is satisfied already, and the other rows are not violated and not solved. The pattern rule for rows 2-5 might not be satisfied, but the rule from Eq. 3 is not violated. Since there is no violation, continue the algorithm by generating a child from the current state. From this point, the algorithm will continue to generate and backtrack until it finds the final state. All of the states generated with this algorithm are shown in Fig. 8 as a tree.

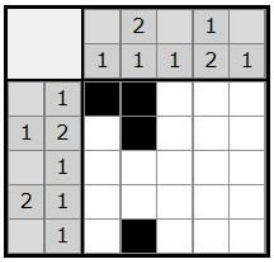

Fig. 6. A new state generated from the state shown in Fig. 4

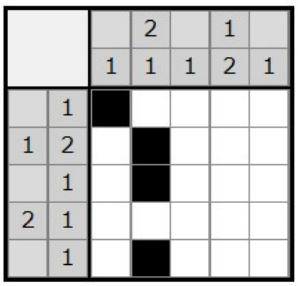

Fig. 7. A new state generated from the state shown in Fig. 4

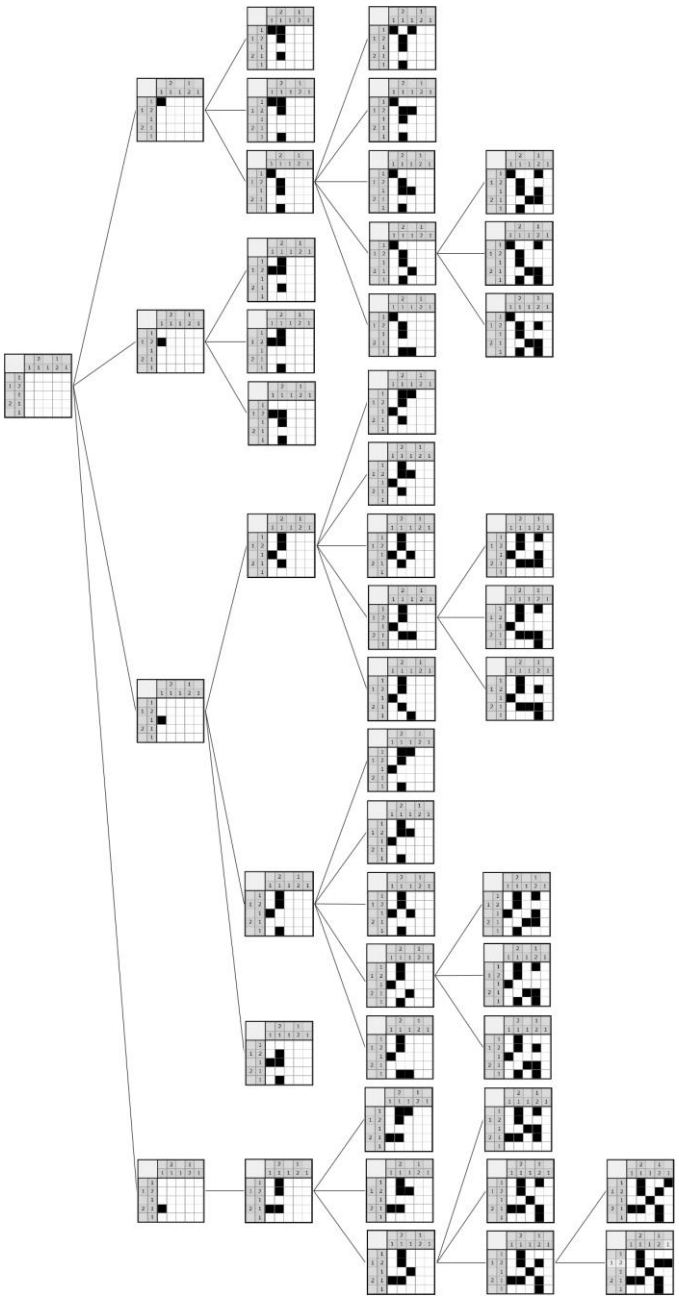

Fig. 8. The tree generated from solving nonogram shown in Fig. 3 with backtracking algorithm

IV. Analysis on Efficiency

In the previous section, a 5x5 nonogram puzzle was solved, generating 46 nodes through the process. If exhaustive search was used instead, every combination of the valid pattern for each column would be generated and checked. Table VII contains the actual possible pattern for each column in the puzzle.

Using the information in Table VII, the number of combinations can be calculated. Hence, by multiplying all the total number of possible patterns, we know that 1125 combinations will be generated by the exhaustive search method. These are the combinations of all five columns with their valid patterns.

Meanwhile, the backtracking algorithm only generates 46 nodes, with two of them being actually subsets of the column combinations. These two states are in the sixth layer of the tree in Fig. 8. This shows us that in the sample case, the backtracking algorithm eliminates 1123 possible combinations of columns as well as the steps leading to them.

TABLE VII. Pattern Rule for Columns

Index of column	Valid Pattern	Total
1	BW{4}	5
	WBW{3}	
	W{2}BW{2}	
	W{3}BW	
	W{4}B	
2	B{2}WBW	3
	B{2}W{2}B	
	WB{2}WB	
3	BW{4}	5
	WBW{3}	
	W{2}BW{2}	
	W{3}BW	
	W{4}B	
4	BWB{2}W	3
	BW{2}B{2}	
	WBWB{2}	
5	W*BW*	5
	BW{4}	
	WBW{3}	
	W{2}BW{2}	
	W{3}BW	

V. Choosing the Right Approach

Using the column-wise approach, we know that for a nonogram puzzle with the size of m x n, the maximum number of possible patterns for a column is n, that is, if the column only requires one black cell. It means that the worst-case scenario will generate n^m combinations if the puzzle is done using exhaustive search. For the row-wise approach, each row will have fewer than m different patterns. The worst-case scenario for this approach will generate m^n combinations. Since there is no restriction on a nonogram puzzle size, then it can be intuitively concluded that the row-wise approach should be taken if $m^n < n^m$, and vice versa. Thus, if m^n turns out to be equal to n^n, we might want to consider another factor.

Now look at the the nonogram in Fig. 9. Solving this puzzle using the backtracking method with column-wise approach will require 25 steps. On the other hand, if row-wise approach is used, we will require only 5 steps to get to the final state from the initial state. This shows us that the cell pattern of the final state also plays an important role in choosing the right approach.

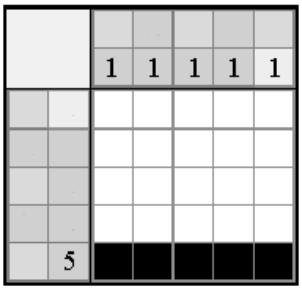

Fig. 9. An example of a 5x5 nonogram puzzle

The cells in Fig. 9 tend to gather in a row, forming a single horizontal line. With this information, we can guess that the row-wise approach will give better results compared to the column-wise approach. The information can also be extracted merely from the puzzle rules, so it is not necessary to have a picture of the final state. This statement, however, is not discussed in this paper and still needs further analysis.

Yu, Lee, and Chen stated in [3] that the backtracking method yields the correct solution with over 98% confidence. The method also solves the puzzle fast, especially for those with compact black cells. The backtracking algorithm helps with the puzzle with the scattered black cells by pruning the tree, which makes the search space smaller.

VI. Conclusions

In this paper, we have seen how the backtracking algorithm can be used to solve nonogram puzzles. The method is demonstrated using a 5x5 nonogram puzzle. There are two types of approaches for the algorithm: the row-wise approach, and the column-wise approach. The row-wise approach fills the next empty row in the current step with its valid pattern. In other words, the row-wise approach finds the possible combination of rows that are in a solved state and eliminates those that violate the columns' rules. On the other hand, the column-wise approach fills the next empty column in the current step with its valid pattern. This approach can also be seen as finding the combinations of solved columns that do not violate the rows' rules.

The puzzle is modeled with mathematical equations to express the rule on the number of black cells and white cells for each column or row. And as for pattern, the rule is expressed by regular expression, using W to represent a white cell and B to represent a black cell. The pattern is used to find the set of valid cell configurations of a row and check the cell configurations of a column if the problem is approached row-wise. If the problem is approached column-wise, the pattern then is used to find the set of valid cell configurations of a column and check the cell configurations of a row. In the

sample case, the nonogram puzzle is solved with column-wise approach.

The tree yielded by the puzzle-solving process contains 47 nodes, in which one of them is the root node representing the initial state, and only two of them are the states reaching the last column. By using backtracking, over 99% of the combinations are eliminated in the process without being checked. Thus, it can be concluded that backtracking reduces a significant number of states generated. This also supported by Yu, Lee, and Chen in [3].

The size and expected cell pattern of a nonogram puzzle about to be solved need to be considered for choosing the most efficient approach. This step has to be discussed further to prove its validity.

Aside from how strong the correct approach is related to the amount of time required to solve a nonogram puzzle using a backtracking algorithm, the algorithm has been shown to produce correct results 98% of the time. Given that it also provides quick solutions, the backtracking algorithm is efficient for sloving nonogram puzzles.

REFERENCES

[1] K. J. Batenburg, S. Henstra, W. A. Kosters and W. J. Palenstijn, "Constructing Simple Nonograms of Varying Difficulty," *Pure Math. Appl.*, vol. 20, pp. 1-15, 2009.

[2] I.-C. Wu, D.-J. Sun, L.-P. Chen, K.-Y. Chen, C.-H. Kuo, H.-H. Kang and H.-H. Lin, "An Efficient Approach to Solving Nonograms," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 5, no. 3, pp. 251-264, 2013.

[3] C.-H. Yu, H.-L. Lee and ·. L.-H. Chen, "An efficient algorithm for solving nonograms," *Applied Intelligence*, vol. 35, pp. 18-31, 20011.

[4] H. Fujiwara, "Nonogram Solution Class: Basic Common Sense," 27 November 2000. [Online]. Available: http://www.pro.or.jp/~fuji/java/puzzle/nonogram/knowhow.elem-4.html. [Accessed 20 May 2022].

[5] Gambiter, "Nonogram," [Online]. Available: https://gambiter.com/puzzle/Nonogram.html. [Accessed 16 May 2022].

[6] nonogram.org, "Japanese crossword «Dolphin»," 29 April 2020. [Online]. Available: https://www.nonograms.org/nonograms/i/32344. [Accessed 18 May 2022].

[7] nonograms.org, "Japanese crossword «Fan»," 10 May 2022. [Online]. Available: https://www.nonograms.org/nonograms/i/56638. [Accessed 20 May 2022].

[8] R. Munir, 'Algoritma Runut-balik (Backtracking) (Bagian 1)', Sekolah Teknik Elektro dan Informatika ITB, 2021.

[9] M. L. Khodra, 'String Matching dengan Regular Expression', Sekolah Teknik Eletro dan Informatika ITB, 2022.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Mei 2022

Ttd
Christine Hutabarat
13520005