제약 충족 문제 기반 네모 로직 풀기

Solving Nonogram as Constraint Satisfaction Problem

ABSTRACT

Nonogram is a logical puzzle on a two-dimensional grid, that is solved based on the numerical constraints on each row and column. This paper specifically uses Generalized Arc Consistency (GAC) algorithm to solve the nonogram as Constraint Satisfaction Problem (CSP).

I. INTRODUCTION

1.1 Nonogram

Nonogram (also known as Picross, Griddlers, Hanjie, or Paint by Numbers) is a logical puzzle on a two-dimensional grid. Originally from Japan since 1980s, this logical puzzle is one of the most famous logical games in Japan or Netherlands [Figure 1]. Nonogram requires the solver to determine whether each cell of the two-dimensional grid is empty or filled, given the numerical constraints on each row and column [1]. The puzzle on the question "Is this puzzle solvable?" is a NP-complete problem [2].

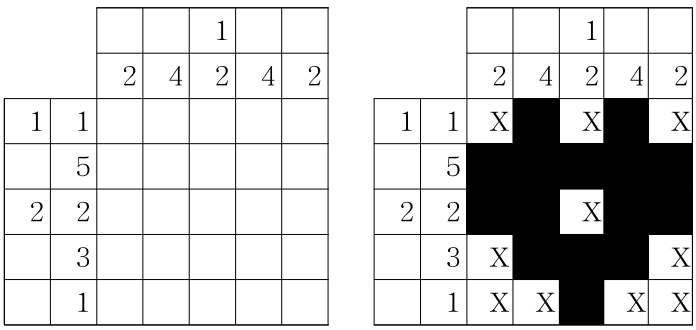

Figure 1. Nonogram example

1.2 Related Works

Wiggers [5] used genetic algorithm (GA) and depth-first search (DFS) algorithm to solve the nonogram. Genetic algorithm, however, might get stuck in local optima. Other studies proposed DFS and best-first search (BFS) algorithm [6, 7] to solve the puzzle.

1.3 Nonogram as Constraint Satisfaction Problem

In this paper, I implemented a generalized arc consistency (GAC) algorithm, approaching the nonogram as a constraint satisfaction problem (CSP). Next section will introduce a brief description of each method. Following will be the results and conclusions of the implemented program.

II. METHOD

2.1 Constraint Satisfaction Problem

"Constraint Satisfaction Problem (CSP) is a problem composed of a finite set of variables, each of which is associated with a finite domain, and a set of constraints that restricts the values the variables can simultaneously take. The task is to assign a value to each variable satisfying all the constraints [3]."

In the case of solving nonograms, variables correspond to each row and column (in this paper, noted as "R" and "C" followed by each row/column number). Constraints in nonogram are the ground-rules of nonogram:

1) Each cell must be either colored(black) or left empty.

2) Each number represents the length of one black run.

3) There should be at least one empty cell between consecutive black runs.

Therefore, the domain of each row and column contains all the possible permutations of cells based on the constraints.

2.2 Generalized Arc Consistency

"A variable of a constraint satisfaction problem is arc-consistent with another one if each of its admissible values is consistent with some admissible value of the second variable. If a variable is not arc consistent with another one, it can be made so by removing some values from its domain [4]."

By removing inconsistent arcs from the domain, generalized arc consistency (GAC) algorithm results in narrowing down the search space efficiently. However, the removal of domain values can cause inconsistency in other variables. Hence, it is important to check if consistency is violated by enforcing consistency in other variables. In addition, GAC can bring about three possible outcomes:

1) Each domain has a single value, a unique solution.

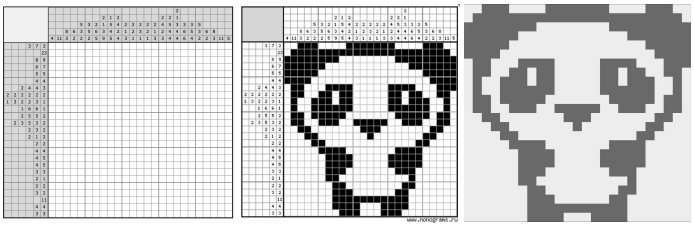

Figure 2. Panda Nonogram Example

2) At least one domain is empty, which means non-solvable.

3) Some domains have more than one value, requiring further search using simplified CSP.

Since nonogram has only one solution per puzzle, outcomes are limited to two in this paper: a unique solution or a non-solvable puzzle.

2.3 Input

The program takes text file as input. The file consists of three parts: size, rows, and columns. The size is a line consisting of row size and column size. Rows and columns are line by line numbers of each row/column constraints of the nonogram. All the input data to test the algorithm will be crawled through an online website, following the format.

III. FINDINGS

3.1 Input and Output

For input data, I crawled 200 nonograms on a free online nonogram website Nonograms.org (https://www.nonograms.org/). Three different sizes - 15×15, 20×20, 25×25 - with multiple difficulty levels were tested. The difficulty of the nonogram is subjective to the author of the nonogram.

Take the right side of Figure 2 as an example of the output format of the algorithm. The rest of Figure 2 shows the original nonogram puzzle and its solution.

3.2 Performance

The performance of the implemented algorithm was measured under this environment: 2.7GHz quad-core Intel Core i7 with

Table 1. Performance of algorithm in different sizes

Size	n	Average Difficulty	Average Runtime (s)	Runtime Stdev.
15 × 15	75	2.17 / 5.0	0.3532	0.3148
20 × 20	75	2.21 / 5.0	5.436	4.9847
25 × 25	50	2.81 / 5.0	56.9131	93.5619

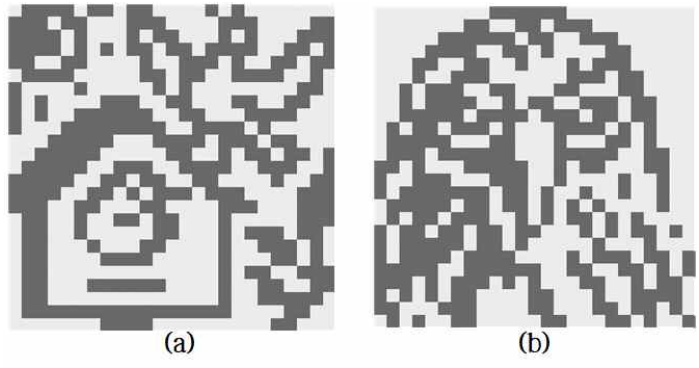

Figure 3. (a) Runtime of 465.16s. (b) Runtime of 232.8871s

16GB RAM and Mojave Mac OS. The algorithm was developed in Python.

As shown in Table 1, the average runtime for all sizes was relatively short compared to the previous studies using algorithms like GA and BFS (For comparison, read [6], [8]). One thing that stood out was the runtime standard deviation for 25×25 nonograms. By using IQR (Interquartile Range) method, I was able to detect three outliers that highly influenced on runtime standard deviation. Figure 3 shows the two outliers with runtime of 465.16 seconds and 232.8871 seconds. The result, however, is obvious due to my approach using CSP. The domains are found based on the permutation of all consecutive runs of the constraints. Therefore, smaller consecutive runs, either black or empty, results in larger domain possibilities. It is apparent that both 3(a) and 3(b) has small consecutive runs, hence, prompts longer runtime.

IV. CONCLUSION

A nonogram solving algorithm was implemented using generalized arc consistency (GAC) algorithm, based on constraint satisfaction problem (CSP) approach. The performance of the proposed algorithm shows that using GAC and CSP solves the nonogram in a relatively short amount of time. However, nonograms with small consecutive runs can produce distinctively longer runtime.

One of the limitation of this paper is that I excluded the multi-valued domain case of the GAC outcome. I would like to further work on it and apply other algorithms to solve the simplified CSP. Also, it would be possible to introduce intersection technique. Instead of using simple permutation to find the domain, intersection technique will efficiently help reducing the domain in the first place.

REFERENCES

1. Dandurand, Frédéric, Denis Cousineau, and Thomas R. Shultz. "Sovling nonogram puzzles by reinforcement learning." Proceedings of the Annual Meeting of the Cognitive Science Society. Vol. 34. No. 34. 2012.

2. Ueda, Nobuhisa, and Tadaaki Nagao. "NP-completeness results for NONOGRAM via parsimonious reductions." preprint (1996).

3. Tsang, Edward. Foundations of constraint satisfaction: the classic text. BoD–Books on Demand, 2014.

4. "Local consistency." Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 14 January 2019. Web. en.wikipedia.org/w/index.php?title=Plagiarism&oldid=5139350. 19 June, 2019.

5. W. A. Wiggers, "A Comparison of a Genetic Algorithm and a Depth First Search Algorithm Applied to Japanese Nonograms," *Twente Student Conference on IT*, Jun. 2004.

6. Jing, Min-Quan, et al. "Solving Japanese puzzles with logical rules and depth first search algorithm." 2009 International Conference on Machine Learning and Cybernetics. Vol. 5. IEEE, 2009.

7. D. Stefani, A. Aribowo, K. V. I. Saputra, S. Lukas, "Solving pixel puzzle using rule-based techniques and best first search", International Conference on Engineering and Technology Development, pp. 118-124, 2012.

8. Wang, Wen-Li, and Mei-Huei Tang. "Simulated annealing approach to solve nonogram puzzles with multiple solutions." Procedia Computer Science 36 (2014): 541-548.