School of Data and Computer Science, Sun Yat-sen University

P02 CSP and KRR

Suixin Ou Yangkai Lin

September 28, 2020

Contents

1 Futoshiki (GAC, C++/Python)

1.1 Description

Futoshiki is a board-based puzzle game, also known under the name Unequal. It is playable on a square board having a given fixed size (4×4 for example), please see Figure 1.

The purpose of the game is to discover the digits hidden inside the board's cells; each cell is filled with a digit between 1 and the board's size. On each row and column each digit appears exactly once; therefore, when revealed, the digits of the board form a so-called Latin square.

At the beginning of the game some digits might be revealed. The board might also contain some inequalities between the board cells; these inequalities must be respected and can be used as clues in order to discover the remaining hidden digits.

Each puzzle is guaranteed to have a solution and only one.

You can play this game online: `http://www.futoshiki.org/`.

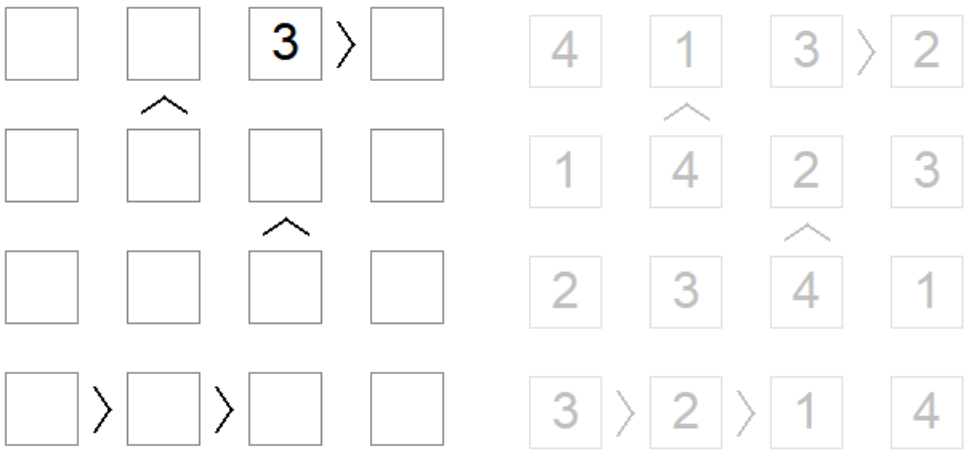

Figure 1: Futoshiki Puzzles

1.2 Tasks

1. Describe with sentences the main ideas of the GAC algorithm and the main differences between the GAC and the forward checking (FC) algorithm. (10 points)

2. The GAC_Enforce procedure from class acts as follows: when removing d from CurDom[V], push all constraints C' such that $V \in scope(C')$ and $C' \notin$ GACQueue onto GACQueue. What's the reason behind this operation? Can it be improved and how? (20 points)

3. Use the GAC algorithm to implement a Futoshiki solver by **C++** or **Python**. (20 points)

4. Explain any ideas you use to speed up the implementation. (10 points)

5. Run the following 5 test cases to verify your solver's **correctness**. We also provide test file "datai.txt" for every test case i. Refer to the "readme.txt" for more details. (20 points)

6. Run the FC algorithm you implemented in E04 and the GAC algorithm you implemented in Task 3 on the 5 test cases, and fill in the following table. In the table, "Total Time" means the total time the algorithm uses to solve the test case, "Number of Nodes Searched" means the total number of nodes traversed by the algorithm, and "Average Inference Time Per Node" means the average time for constraint propagation (inference) used in each node (note that this time is not equal to the total time divided by the number of nodes searched). Analyse the reasons behind the experimental results, and write them in your report. (20 points)

Test Case	Algorithm	Total Time	Number of Nodes Searched	Average Inference Time Per Node
1	FC			
	GAC			
2	FC			
	GAC			
3	FC			
	GAC			
4	FC			
	GAC			
5	FC			
	GAC			

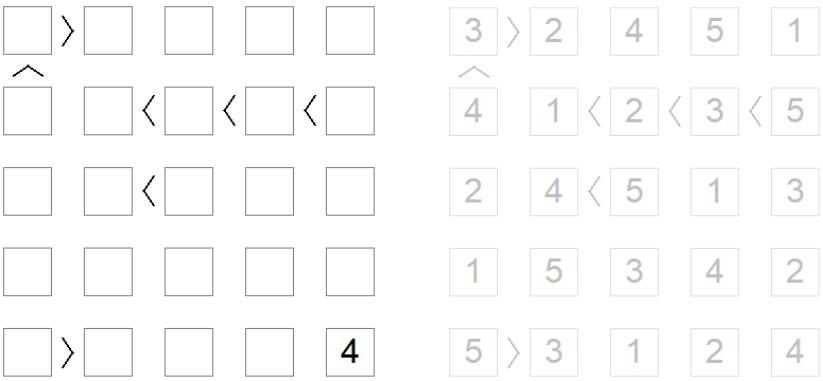

Figure 2: Futoshiki Test Case 1

Figure 3: Futoshiki Test Case 2

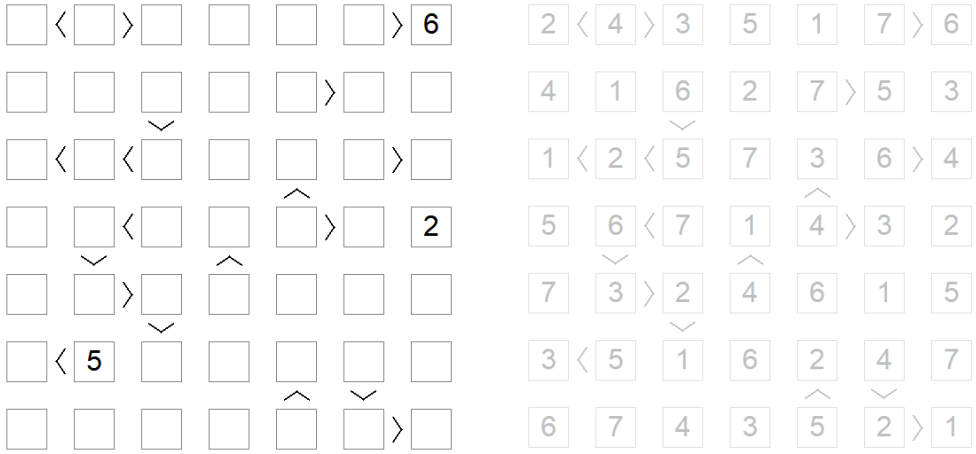

Figure 4: Futoshiki Test Case 3

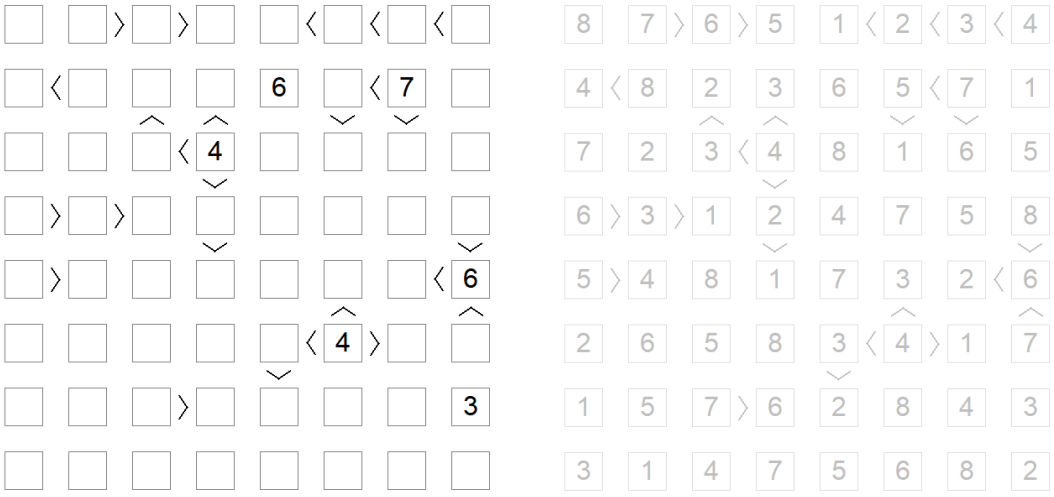

Figure 5: Futoshiki Test Case 4

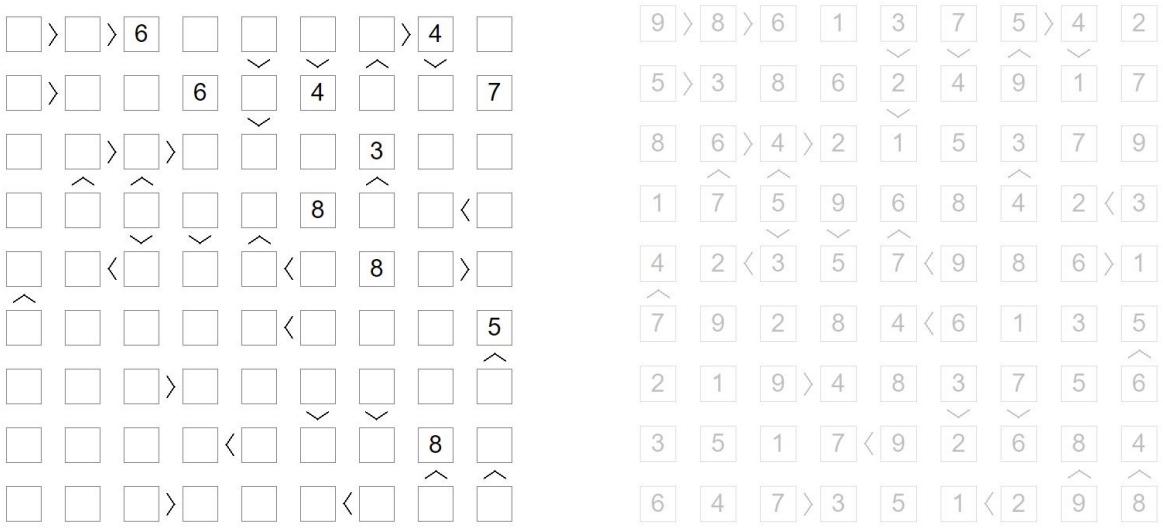

Figure 6: Futoshiki Test Case 5

2 Resolution

1. Implement the MGU algorithm. (10 points)

2. Using the MGU algorithm, implement a system to decide via resolution if a set of first-order clauses is satisfiable. The input of your system is a file containing a set of first-order clauses. In case of unsatisfiability, the output of your system is a derivation of the empty clause where each line is in the form of "R[8a,12c]clause". Only include those clauses that are useful in the derivation. (10 points)

3. Explain any ideas you use to improve the search efficiency. (5 points)

4. Run your system on the examples of hardworker(sue), 3-blocks, Alpine Club. Include your input and output files in your report. (15 points)

5. What do you think are the main problems for using resolution to check for satisfiability for a set of first-order clauses? Explain. (10 points)