P02 CSP (GAC)

Suixin Ou

School of Computer Science
Sun Yat-sen University

October 19, 2021

Task

Problem

- Futoshiki is a board-based puzzle game. It is playable on a square board having a given fixed size.
- 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/.

Task

Input-output

- Input: a n x n matrix of initial state, and a list of inequal constraints.

Figure 1: An Futoshiki Puzzle

- Output: the n x n matrix of the terminate state that satisfys all constraints (including inequal constraints, row and column constraints).

Task

Grading

- 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)
- 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)
- Use the GAC algorithm to implement a Futoshiki solver by **C++** or **Python**. (20 points)
- Explain any ideas you use to speed up the implementation. (10 points)
- Run the following 5 test cases to verify your solver's **correctness**. (20 points)

Task

Grading

- Run the FC algorithm you implemented in E04 and the GAC algorithm you implemented in Task 3 on the 5 test cases, **fill in the following table and analyse the reasons behind the experimental results**. 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). (20 points)

Task

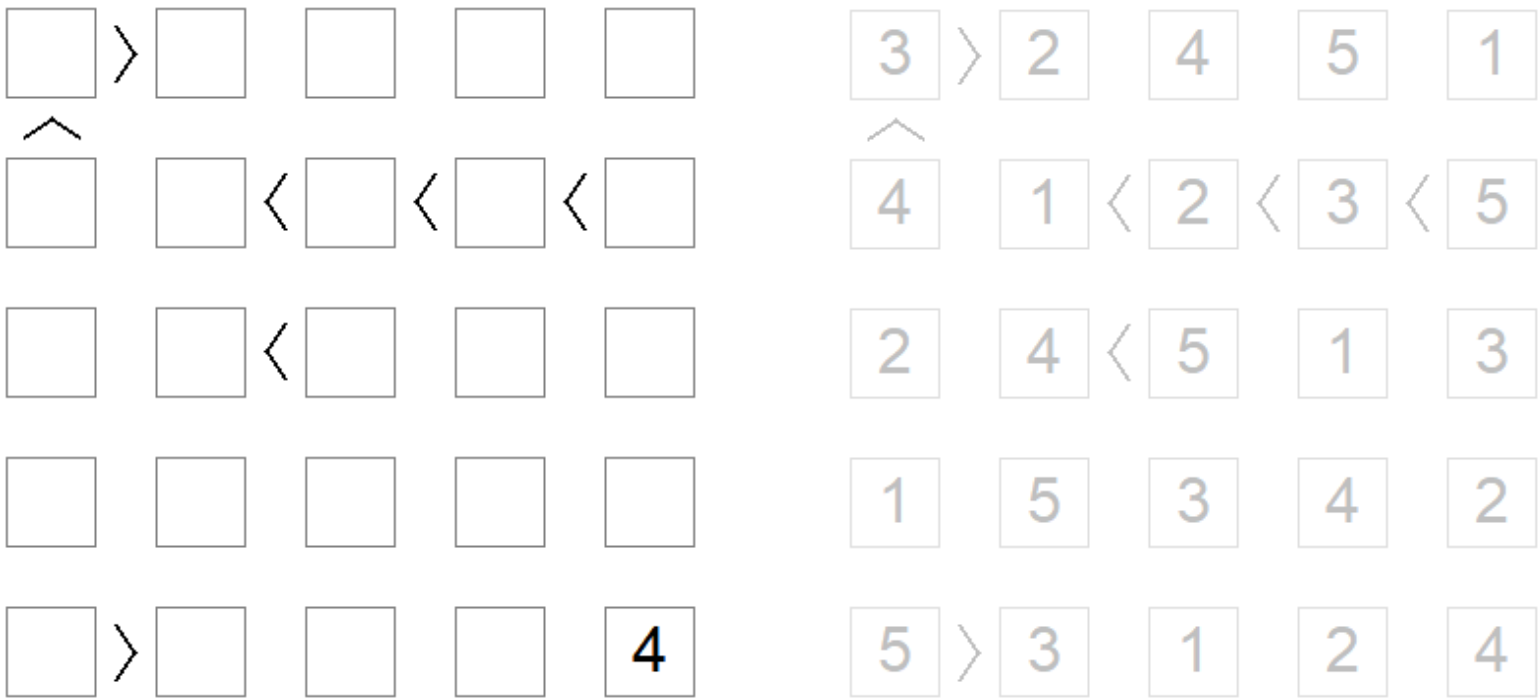

Figure 1: Futoshiki Test Case 1

Task

Figure 2: Futoshiki Test Case 2

Task

Figure 3: Futoshiki Test Case 3

Task

Figure 4: Futoshiki Test Case 4

Task

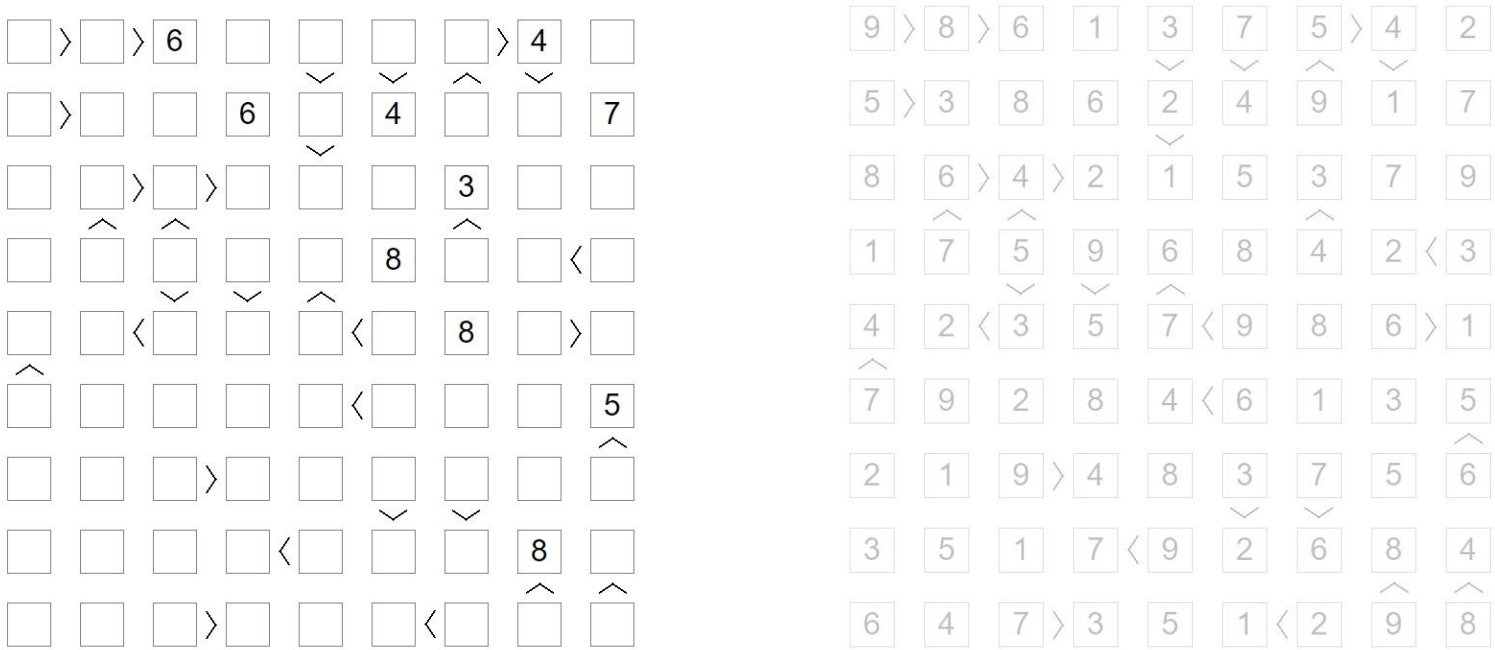

Figure 5: Futoshiki Test Case 5

Task

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			

Task

Submission

pack your report `P02_YourNumber.pdf` and source code into zip file `P02_YourNumber.zip`, then send it to `ai_course2021@163.com`.

Solution

Algorithm procedure

```
GAC(Level) /*Maintain GAC Algorithm */
   If all variables are assigned
      PRINT Value of each Variable
      RETURN or EXIT (RETURN for more solutions)
                     (EXIT for only one solution)
   V := PickAnUnassignedVariable()
   Assigned[V] := TRUE
   for d := each member of CurDom(V)
      Value[V] := d
      Prune all values of V ≠ d from CurDom[V]
      for each constraint C whose scope contains V
        Put C on GACQueue
      if(GAC_Enforce() != DWO)
        GAC(Level+1) /*all constraints were ok*/
      RestoreAllValuesPrunedFromCurDoms()
   Assigned[V] := FALSE
   return;
```

Solution

Algorithm procedure

```
GAC_Enforce()
   // GAC-Queue contains all constraints one of whose variables has
   // had its domain reduced. At the root of the search tree
   // first we run GAC_Enforce with all constraints on GAC-Queue
   while GACQueue not empty
      C = GACQueue.extract()
      for V := each member of scope(C)
          for d := CurDom[V]
                Find an assignment A for all other
                variables in scope(C) such that
                C(A ∪ V=d) = True
                if A not found
                   CurDom[V] = CurDom[V] - d
                   if CurDom[V] = ∅
                        empty GACQueue
                        return DWO //return immediately
                   else
                        push all constraints C' such that
                        V ∈ scope(C') and C' ∉ GACQueue
                        on to GACQueue
   return TRUE //while loop exited without DWO
```

Solution

Read input

```
20      void initial() {
21          //初始地图
22          maps = {{0, 0, 0, 0, 0},
23                  {0, 0, 0, 0, 0},
24                  {0, 0, 0, 0, 0},
25                  {0, 0, 0, 0, 0},
26                  {0, 0, 0, 0, 4}};
27          nRow = maps.size();
28          nColumn = maps[0].size();
29
30          //初始化行列约束
31          for (int i = 0; i < 5; i++) {
32              for (int j = 0; j < 5; j++) {
33                  if (maps[i][j] != 0) {
34                      Count_RowNumbers[i][maps[i][j]]++;
35                      Count_ColumnNumbers[j][maps[i][j]]++;
36                  }
37              }
38          }
39          //添加比较符号约束
40          addConstraints(0, 1, 0, 0);
41          addConstraints(0, 0, 1, 0);
42          addConstraints(1, 1, 1, 2);
43          addConstraints(1, 2, 1, 3);
44          addConstraints(1, 3, 1, 4);
45          addConstraints(2, 1, 2, 2);
46          addConstraints(4, 1, 4, 1);
47      }
```

```
37          //初始化行列约束
38          for (int i = 0; i < 9; i++) {
39              for (int j = 0; j < 9; j++) {
40                  if (maps[i][j] != 0) {
41                      Count_RowNumbers[i][maps[i][j]]++;
42                      Count_ColumnNumbers[j][maps[i][j]]++;
43                  }
44              }
45          }
```

Visualize output

```
124     //显示结果
125     void show() {
126         for (int i = 0; i < nRow; i++) {
127             for (int j = 0; j < nColumn; j++) {
128                 cout << maps[i][j] << " ";
129             }
130             cout << endl;
131         }
132         cout << "======================" << endl;
133     }
```

Solution

Check whether conditions are all satisfied. **You should finish a check function for the Generalized Arc Consistency(GAC) algorithm.**

```
54      //check函数检查当前位置是否可行，
55      //以下注释掉的内容是back tracking算法的check函数
56      //你们需要自行实现GAC算法的check部分
57      bool check(int x, int y) {
```

Search for correct solution.

```
134     //搜索流程，可以不用修改这部分
135 >   bool search(int x, int y) {···
183     }
```

The End