Yu Hu

March 22, 2017

Sudoku Solver report

This report uses three methods to solve sudoku puzzles. Sudoku is a 9*9 grid within which contains 9 3*3 sub-grids. Numbers from 1 to 9 can be put into each cell. However, each number can only appear once in a row, a column, and the sub-grid that it belongs to.

By the rule of sudoku, when a value is assigned to a cell, a constraint is set to its related vertical, horizontal and sub-grid cells. In other word, these cells cannot be assigned the same value. Therefore, the sudoku problem is defined as constraint satisfaction problem.

To solve the sudoku, four methods are used in this paper: backtracking, forward checking(FC) and FC with minimum remaining value (MRV) heuristic and FC with MRV and least-constraining-value (LCV) heuristic.

Backtracking method uses depth first search that assigns values to the cell row by row. When no value from 1 to 9 can be added to a cell without violating the rule, it backtracks and update the value of the previous cell.

Forward checking (FC) method also uses depth first search that assigns value to the cell row by row. However, when a value is assigned to a cell, the value is removed from the domain of its related cells (vertical, horizontal, and sub-grid). If a domain of its related cell becomes empty due to the action, it backtracks. Hence, the invalid value can be detected earlier.

Minimum remaining value forward checking (MRV FC) uses depth first search. However, instead of assigning value row by row, it assigned cell based on the size of the cell's domain. MRV FC selects the cell with the least remaining value, i.e. smallest domain, to assign value. Similar to FC, when a value is assigned to a cell, constraint is set to its related cell. If a domain of its related cell becomes empty due to the action, the search backtrack. Because the cell with

the smallest domain is selected fist, the search can start backtracking earlier and avoid going too deep into a wrong path.

Least-remaining-value heuristic assign the value that set the least constraint to its related cell first. By assigning order with LRV heuristic, the search keeps the options for paths at maximum.

Five puzzles with different depth of the goal is used to test the three methods. The depth of the goal (d) is defined as the number of empty cells in a sudoku puzzle. Effective branching factor (b*) = expanded nodes ^ (1/d).

The effective branching factors of the three methods are consistent with the expectation. b* of FC with MRV LCV is the smallest and b* of BT is the largest. The basic BT method will backtrack when there's no legal value that can be assigned to the variable. BT can be improved by forward checking methods that assigned constraint to the related cells. Failure can be detected earlier when the domain of a related cell becomes empty. Thus before that node is expanded, the search knows that this path will fail. However, FC still chooses expanded variable based on the row and column of the cell. Also, it assign value to a variable in the order of 1 to 9. Forward checking method can be more efficient by MRV and LCV heuristic. MRV heuristic decides the order of assigning variable. LCV heuristic decides the order of assigning values to a variable. Backtracking can be more efficient by strategically choosing the order of assigning variable. MRV FC strategy, for instance, assign the most constrained value first, so the search can fail faster when it goes down a wrong path. LCV heuristic assign value that set the least constraint to the adjacent cells so it leaves more flexibility to the following search.

Puzzle 1:
the depth of the goal (d) = 55

	BT	FC	MRV FC	LCV
total time:	10	27	5	8

	BT	FC	MRV FC	LCV
search time:	8	25	3	6
backtracking number:	8107	2724	0	0
expanded nodes number:	11873	2779	55	55
memory used:	1.95 MB	4.55 MB	1.30 MB	1.95 MB
effective branching factor	1.20	1.16	1.08	1.08

Puzzle2:
the depth of the goal (d) = 57

	BT	FC	MRV FC	LCV
total time:	10	65	5	19
search time:	9	63	3	17
backtracking number:	11502	8410	55	306
expanded nodes number:	17021	8467	112	363
memory used:	1.95 MB	11.05 MB	1.30	1.3
effective branching factor	1.19	1.18	1.09	1.11

Puzzle 3
the depth of the goal (d) = 58

	BT	FC	MRV FC	LCV
total time:	22	111	14	22
search time:	21	109	11	20
backtracking number:	51714	29380	266	453
expanded nodes number:	73626	29438	324	511
memory used:	1.95 MB	3.66 MB	1.30 MB	1.95 MB
effective branching factor	1.22	1.20	1.11	1.12

Puzzle 4
the depth of the goal (d) = 54

	BT	FC	MRV FC	LCV
total time:	14	18	6	8
search time:	13	16	4	4
backtracking number:	21227	674	32	0
expanded nodes number:	31057	729	87	55
memory used:	1.95 MB	1.95 MB	1.30 MB	1.30
effective branching factor	1.21	1.13	1.09	1.08

Puzzle 5
the depth of the goal (d) = 52

	BT	FC	MRV FC	LCV
total time:	2	8	5	6
search time:	1	6	3	3
backtracking number:	817	193	10	0
expanded nodes number:	1099	245	62	52
memory used:	1.95 MB	1.30 MB	1.30 MB	1.30 MB
effective branching factor	1.15	1.14	1.08	1.08

Effective branching factor of backtracking, forward checking, and MRV FC.

	Search Cost (nodes generated)				Effective Branching Factor			
d	**BT**	**FC**	**MRV FC**	**LCV**	**BT**	**FC**	**MRV FC**	**LCV**
52	1099	245	62	52	1.15	1.14	1.08	1.08
54	31057	729	87	55	1.21	1.13	1.09	1.08
55	11873	2779	55	55	1.20	1.16	1.08	1.08
57	17021	8467	112	363	1.19	1.18	1.09	1.11
58	73626	29438	324	511	1.22	1.20	1.11	1.12