Homework #3 Constraint Satisfaction Problem Solver Experiment Results

Experiment Introduction:

The CSP solver program was run on two primary types of constraint satisfaction problems (CSP). The first is the tri-coloring problem for the states and territories in Australia.[1] The second type of problem is Sudoku, which is substantially more complex problem than map tri-color. Below is a summary of the problem scopes:

Problem		Number of Variables	Domain Size	Number of Binary Constraints	Search Tree Size[2]
Australia Tri-Coloring		7	3	9	2187
Sudoku	4x4	16	4	56	$4.295 * 10^{9}$
	9x9	27	9	810	$5.815 * 10^{25}$
	16x16	256	16	4992	$1.798 * 10^{308}$

Seven Sudoku puzzles were used. The first puzzle is one of the hardest 4x4 puzzles that has a unique solution (the puzzle only has 4 cells fixed). Five puzzles were 9x9 and were categorized depending on the puzzle's level of difficulty. The five categories are: Easiest, Easy, Medium, Hard, and Hardest.3 The seventh puzzle was 16x16 and was rated as Easy. Appendix A contains the Sudoku puzzles used. Enclosed in my submission is an Excel workbook that covert Sudoku puzzles of varying size into files that are compatible with the standard specified in the homework description.

Experiment Metrics:

Three primary metrics were used to assess the time complexity of the CSP solver on a given problem. They are:

1. **Backtrack Function Call Count** – This quantifies the number of times the function "Backtrack" is called.

2. **Variable Value Setting Count** – This quantifies the number of times any variable in the CSP was assigned a value; this count is incremented irrespective of whether the value setting is consistent or inconsistent. Using the pseudocode on page 215 of Russell and Norvig, this metric corresponds to the number of times the *for* loop is run.

3. **Algorithm Runtime** – This quantifies in seconds how long the CSP solver took to find a solution. This metric is subject to variation due to computer system variation and is only a rough estimate of algorithm performance. This is measured in seconds.

If the backtracking algorithm executes on a CSP with optimal efficiency, the "Backtrack" function call count and variable value setting count would both equal the number of variables in the CSP. A greater difference between these metrics and the number of variables indicates the algorithm needed to do more searching.

Experiment Results:

[1] See file "australia.txt" for the Australia state and province tri-coloring problem.
[2] Search tree size for Sudoku does not account for already assigned values.
[3] The easiest puzzle came from the website: http://www.sudokukingdom.com/very-easy-sudoku.php. The remaining for puzzles were generated by the website: http://www.websudoku.com/. The difficulty classification for all puzzles was provided by the source website.

Metric	Type	Australia	Hardest 4x4 Sudoku	Easiest 9x9 Sudoku	Easy 9x9 Sudoku	Medium 9x9 Sudoku	Hard 9x9 Sudoku	Hardest 9x9 Sudoku	Easy 16x16 Sudoku
Backtrack Function Call Count	With FC	7	16	81	136	112	83	168	15926
	No FC	7	16	81	N/A	N/A	N/A	N/A	N/A
Variable Value Setting Count	With FC	7	16	81	143	121	84	176	17798
	No FC	7	16	81	N/A	N/A	N/A	N/A	N/A
Algorithm Runtime (s)	With FC	0.000	0.000	0.062	0.110	0.110	0.078	0.140	46.432
	No FC	0.000	0.000	0.140	N/A	N/A	N/A	N/A	N/A

Table 1 - CSP Solver Experiment Results With and Without Forward Checking

Table 1 contains the experimental results comparing the efficiency of the CSP solver with and without forward checking. For trivial problems such as the tri-coloring of Australia, there is not a meaningful performance difference between the two approaches. This even held true for a 4x4 Sudoku, where the two approaches showed comparable performance. However, at the easiest 9x9 Sudoku, the performance of the algorithm with and without forward checking begins to diverge. While the number of Backtrack function calls and variable value setting counts are equivalent, the algorithm runtime without forward checking was more than double (62ms versus 140ms). While computation time on a CPU is subject to many variables (e.g. operating system scheduling, memory contention, etc.), the time does provide inside into the efficiency of the algorithms. The difference in run times is due to the larger domain sizes when forward checking is not used. The additional domain values necessitate more computational checks to reach the same solution.

Beyond the simple puzzles that can be that can be solved with little to no searching, Backtrack search without forward checking becomes prohibitive. For example, for even the Easy 9x9 Sudoku puzzle, backtrack search without forward checking could not find solution despite running for a few hours. There are two primary reasons it struggles without forward checking on these boards:

1. Arc consistency was not enforced before the algorithm started. As such, the domain size for all unassigned variables is 9 for all unassigned cells in a 9x9 puzzle when forward checking is disabled. As such, the minimum remaining value (MRV) heuristic has limited value since it does not have a way to distinguish between those cells that can easily be solved and those that cannot.

2. Degree Heuristic is used to break the tie in MRV. In the implementation described by Russell and Norvig, degree heuristic specifies that the node to be selected for expansion is the one with the most unassigned neighbors. While this approach works well for problems like the tri-coloring of the Australian states and provinces, it can do a poor job in Sudoku where it is often preferable to examine those nodes with the least unassigned neighbors.

Table 2 shows the performance of the CSP solver on the same 9x9 Sudoku puzzles when degree heuristic is changed from maximum degree to minimum degree. The performance of the algorithm with forward checking stayed about the same or slightly improved. However, there was substantial speed for the algorithm without forward checking. What previously was unsolvable in hours was solved in a matter of seconds (or less) when the degree heuristic was switched to minimum.

When transitioning to larger boards (e.g. 16x16), performance degradation was observed when switching to the minimum degree heuristic when forward checking was enabled.

Metric	Type	Hardest 4x4 Sudoku	Easiest 9x9 Sudoku	Easy 9x9 Sudoku	Medium 9x9 Sudoku	Hard 9x9 Sudoku	Hardest 9x9 Sudoku
Backtrack Function Call Count	With FC	16	81	81	140	94	84
	No FC	16	81	301	2170	94	218
Variable Value Setting Count	With FC	16	81	81	145	95	85
	No FC	16	81	2063	18885	199	1318
Algorithm Runtime (s)	With FC	0.000	0.076	0.077	0.125	0.079	0.078
	No FC	0.000	0.125	1.500	13.781	0.390	1.109

Table 2 - CSP Solver Sudoku Performance With and Without Forward Checking Using the Modified Minimum Degree Heuristic

Appendix A – Suokdu Puzzles Used in CSP Solver Experiments

	1	2	3	4
A	4			
B				2
C	3			
D				1

Figure 1 – Hardest 4x4 Sudoku Puzzle

	1	2	3	4	5	6	7	8	9
A	9	2		6		8		3	
B	3		7				1	6	8
C	6			7	4	3		9	
D	2		4	5	3				6
E	8			1	7			5	9
F		1	9			2	4	7	
G		3	8		5	1	6		7
H		5	2				9		1
I		9		4	2	7	3		

Figure 2 - Easiest Difficulty 9x9 Sudoku Puzzle

	1	2	3	4	5	6	7	8	9
A				3	2	7		5	1
B					8				7
C				5		9	4		
D			3					9	6
E	7		2	1	9	3	5		4
F	8	4					1		
G			7	8		6			
H	5				3				
I	6	1		9	5	2			

Figure 3 – Easy Difficulty 9x9 Sudoku Puzzle

	1	2	3	4	5	6	7	8	9
A		3	9						6
B					7				
C			1	5		9		7	
D			3			7	1	6	
E			8	3	5	6	2		
F		6	5	4			3		
G		9		7		8	6		
H					4				
I	2						9	5	

Figure 4 – Medium Difficulty 9x9 Sudoku Puzzle

	1	2	3	4	5	6	7	8	9
A	8				4				9
B		1	4		6	8			
C	3	6		7					
D							9		
E	2	9	7				8	3	5
F			5						
G						3		5	7
H				5	8		6	9	
I	5				9				4

Figure 5 – Hard Difficulty 9x9 Sudoku Puzzle

	1	2	3	4	5	6	7	8	9
A	4	9					5	6	
B				9		2			
C		6							8
D	3			2	8			7	1
E									
F	9	8			3	7			4
G	7							4	
H				3		8			
I		1	9					8	2

Figure 6 – Hardest Difficulty 9x9 Sudoku Puzzle

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A			15			13		3		9	12			7		
B	3	10						6			7					
C		9			12	15		4					5			10
D				1	14	7		2	13	10	6					
E	7	4			8	6			1	3			15		12	
F				11	13					12						14
G	2	8	1				14	12			10					
H	14	3		6						11			9			
I			10	4					5		13	8	1			
J			7	15	6	4		9	14					12		
K						3					2	4		8		
L		2			7			8		1				13	15	
M				13	5			15		14	11	12				9
N		6				10	11	14		13	15					
O	15	14			4	9				2		3	10	6	11	
P		11			3				6			7				

Figure 7 – Easy Difficulty 16x16 Sudoku Puzzle