Solving Sudoku Puzzles using CSPs

Zhang Dongjun, Xu Tunan, Calvin Chen, Derrick Teo

Abstract. Analysis and evaluation of variants of the backtracking search algorithm including: 1. variable ordering mechanisms/heuristics (MRV, Degree heuristic) 2. Value ordering heuristics (LCV, RV) 3. Inference Mechanisms (AC3)

1 Problem Definition

Sudoku is a logic-based, combinatorial number-placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids that compose the grid contains the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle, has a single solution.

1.1 Constraint Satisfaction Problem (CSP) model

The sudoku problem was modelled as a CSP with discrete variables and global constraints. The properties were defined for sudoku as such:

Variables:	$C_{0,0}$, …, $C_{8,8}$ (81 variables)
Domains:	$D_{i,j} = \{1, …, 9\}$
Constraints:	*AllDiff*($C_{i,j}$,…,$I_{i,j+2}$; $C_{i+1,j}$,…,$C_{i+1, j+2}$; $C_{i+2,j}$,…,$C_{i+2,j+2}$) for (i,j) ∈ {(0,0),(0,3),(0,6),(3,0),(3,3),(3,6),(6,0),(6,3),(6,6)} *AllDiff*($C_{i,0}$,…,$C_{i,8}$) for i=0,…,8 *AllDiff*($C_{0,j}$,…,$C_{8,j}$) for j=0,…,8

2 Algorithm Variants

To solve the sudoku CSP, we implemented different variants of the backtracking search algorithm. The backtracking search algorithm involves a Depth First Search (DFS) with single-variable assignments. The variants we attempted are a combination of the heuristics from each section explained below.

2.1 Variable ordering mechanisms/heuristics

Variable ordering heuristics chooses the variable to be assigned in each iteration of the backtracking search algorithm. The variants we used are shown below:
Minimum Remaining Values (MRV).
The minimum remaining-value heuristic chooses the variable with the fewest legal values.

Degree Heuristic (Most Constraining Variable).
The degree heuristic assigns a value to the variable that is involved in the largest number of constraints on other unassigned variables.

2.2 Value Ordering Heuristics

Value ordering heuristics determine the order in which each value is to be tried after choosing a variable. The variants we used are show below:
Least Constraining Value.
The least constraining value heuristic chooses a value that rules out the smallest number of values in variables connected to the current variable by constraints.

Random Value.
The random value heuristic chooses a value with uniform probability from the domain of the selected variable.

2.3 Inference Mechanisms

Inference mechanisms are used to infer domain reductions on unassigned variables. The variants we used are shown below:
Forward Checking.
Each time a variable is assigned, delete from domains of neighboring variables all values that conflict with current variable assignment. Search is terminated when a variable has no legal values to assign.

Arc Consistency (AC3).
For two cells C_i and C_j, C_i is arc-consistent with respect to C_j if and only if for every value $x \in D_i$ there exists some value $y \in D_j$ that x and y do not share the same value. The AC3 algorithm iterates through all arcs and executes domain reductions to maintain arc-consistencies for all arcs.

3 Experimental Setup

Terminology.
We use the following terminology when representing experimental results:

- AC3 (arc consistency)
- FC (forward checking)
- MRV (min remaining values)
- DG (degree heuristic)
- LCV (least constraining value)
- RV (random value)

Description.
The goal of our experiments was to investigate the effect of different methods of selecting variables and values, and for performing inference, on different difficulties to find the best performing variant. We measured the performance of each variant using its total execution time and number of nodes generated.

The variants tested include the following:

	MRV		**DG**	
	LCV	RV	LCV	RV
FC	√	√	√	√
AC3	√	√	√	√

In our experiment we tested all variants on 4 different difficulties of sudoku puzzles. The difficulties include 2 Easy, 1 Medium and 1 Hard puzzle. This was to ensure that the performance of each variant could be observed across all difficulties.

To measure the speed and nodes generated for each variant, we ran our program on Sunfire at each difficulty and took the average execution time and nodes generated across 10 executions. The results were recorded and tabulated below.

4 Results and Discussion

4.1 Experimental Results

Puzzle Difficulty/ Combination	**Easy(a)**	**Easy(b)**	**Medium**	**Hard**
FC+MRV+LCV (1)	0.00797s,78	0.00518s,51	0.0262s,252	0.925s,8354
FC+MRV+RV (2)	0.00581s, 53	0.00550s,51	0.00628s,56	0.128s, 1250
AC3+MRV+LCV (3)	0.398s, 52	0.372s, 51	0.433s, 56	33.4, 4321
AC3+MRV+RV (4)	0.420s, 55	0.373s, 51	0.959s, 123	5.13s, 675
FC+DG+LCV(5)	50.2s, 35618	16.7s,1125	TLE	TLE

FC+DG+RV(6)	TLE	49.0s,30841	TLE	TLE
AC3+DG+LCV (7)	0.452s,56	0.438s, 54	2.56s,388	TLE
AC3+DG+RV (8)	0.440s,53	0.443s,56	20.6s,3207	TLE

4.2 Analysis of experiment results

FC vs AC3
From row 1-4 in the table, we see that AC3 often generates less nodes as compared to forward checking. However, maintaining arc consistency incurs additional overhead which seems to result in more runtime penalty instead of efficiency in the backtracking search. Therefore, we decided to use forward checking as our inference algorithm instead.

MRV vs DG
The minimum remaining-value (MRV) heuristic chooses the variable with the fewest legal values while the degree heuristic (DG) chooses the variable which affects the most domains. The MRV seems more appropriate in this case since it assigns those variables with limited domain size first. This allows the algorithm to detect failure early, and hence solve the problem at a faster rate. This can also be seen from our experimental results from the table where row 1,2,3,4 consistently outperforms row 5,6,7,8. Therefore, we have decided to adopt the MRV heuristic.

LCV vs RV
The least constraining-value (LCV) heuristic chooses the value which has the least constraints with neighbouring domains while the random value (RV) heuristic simply chooses a random value from the current domain. From the experimental results, there seemed to be a mixed result since neither heuristic consistently outperforms the other. Nevertheless, we have decided to adopt the RV heuristic since it incurs less overhead as compared to the LCV heuristic.

Conclusion
Although Sudoku is a difficult CSP problem with a state space exponential in size of input, our combined heuristics were able to consistently solve the puzzle by generating less than 2000 states, exceeding 1000 states only when solving the “World’s hardest Sudoku”[1]. Therefore, we can conclude that our chosen heuristics indeed help to improve the performance of backtracking search in solving the Sudoku puzzle.

[1] https://www.telegraph.co.uk/news/science/science-news/9359579/Worlds-hardest-sudoku-can-you-crack-it.html