CSL302 Artificial Intelligence
LAB 3

Abhishek Chowdhry
2015CSB1002

March 2017

SUDOKU SOLVER USING CSP

We had to formulate the this problem as a CSP(constraint satisfaction problem) by identifying the variables domains and constraints and then solve it using different search algorithms. Then we had to compare their performance based on the performance metrics(such as search time, memory utilization and backtracks).
The performance metrics and the observations for different algorithms are given below:

BS: Generic backtracking search

In this algorithm we just search through through all the possible outcomes by assigning a value to an unassigned variable at each step and backtracking whenever there is an inconsistency. Naturally this algorithm should take the minimum amount of memory because it doesn't store anything but maximum amount of time since it searches through all the possible outcomes. Also the number of backtracks is highest in this search algorithm because no domain specific knowledge is used to select the new unassigned variable or the value which is assigned to that variable.
Search time: 1 minute 12.371 seconds
No of backtracks: 227502770
Memory used: 100,469 bytes

BS-I: Backtracking search with MRV heuristic

This algorithm implements simple backtracking search with some intelligence. i.e. It used domain specific knowledge to select the next unassigned variable. It selects the next unassigned variable the one which has minimum remaining values. Since this algorithm uses some intelligence the search time should decrease because it will explore the most important branches first. Accordingly the number of backtracks should also decrease. But the memory used is the same as backtracking search because there is no need of storing the domains(the next variable to be selected can be selected by applying a simple loop).
Search time: 4.095 seconds
No of backtracks: 734417
Memory used: 100,469 bytes

BS-II: Backtracking search with MRV and LCV heuristic

This algorithm implements BS-I search with some more intelligence. i.e. It uses the domain specific knowledge to select the next value for the variable selected using MRV heuristic. It selects the value which will cause minimum number of restrictions in the domains of the neighbouring variables as the next value. Since this algorithm uses some more intelligence over the previous BS-I search algorithm, the time taken by this algorithm will be almost same(because computations increased but backtracks decreased). Also, the backtracks will be less than the previous algorithm because it will explore the best branch first. But the memory used in this algorithm will be more than the previous algorithm because we will have to store

the updated domains at each point of time after assigning a value to a variable.
Search time: 4.644 seconds
No of backtracks: 689692
Memory used: 22,404,228 bytes

MAC: Backtracking search with MRV and LCV heuristics while maintaining arc consistency

This algorithm implements the BS-II Algorithm while maintaining arc consistency in the at all times. After choosing a variable and the value to assign, it assigns the value to the variable and runs arc consistency on the graph. It does this after each assignment. Since the domains are becoming smaller due to arc consistency algorithm on each step, the backtracks will be less than before. But now since there is a lot of memory to be stored and altered in the form of domains, the memory used will increase. But the time taken will again not change drastically because although the backtracks are decreasing, the computation time is increasing(arc consistency on each step).
Search time: 13.481 seconds
No of backtracks: 295027
Memory used: 1,037,493,845 bytes

Note:- Time taken by the program to run and the memory used by the program have been calculated using the command line functions "time" and "valgrind" respectively.

Conclusion

In the 4 algorithms implemented the number of backtracks is the highest for generic backtracking search and decreases on adding MRV heuristic(BS-I). The number of backtracks further reduces on adding the LCV heuristic(BS-II) and decreases further more on applying arc consistency on each step(MAC). The time taken is the maximum for generic backtracking search and reduces on adding the MRV heuristic. But on adding the LCV heuristic and MAC, the time taken does not decreases further because the computation time of LCV and MAC is greater than the time reduced due to reduction of backtracks.

SUDOKU SOLVER UISNG MINISAT

In this question we had to formulate sudoku in the form of a boolean satisfiability problem and use the standard SAT solver to solve it.The goal was to read the input Sudoku file, encode the puzzle as a CNF formula in the DIMACS format, use MiniSAT to find a satisfying assignment to the CNF formula, read the output of the solver and translate it into the solution for the puzzle.

Approach to formulate Sudoku as a Boolean Satisfiability problem

To formulate sudoku as a boolean satisfiability problem we, used 729 variables.
x(i,j,k) goes to variable number (i*81+j*9+k)
The truth of x(i,j,k) means that the (i,j) coordinate in the sudoku contains value k.
We then took the following constraints:
1) For each co-ordinate (i,j) atleast one of x(i,j,1),x(i,j,2),...,x(i,j,9) is true.
i.e. Every variable contains atleast one entry.
2) In one row, no two variables should have same value.
i.e. For each value k, (-x(i,0,k) or -x(i,1,k)) and (-x(i,0,k) or -(i,2,k)) and..... should be true
3) In one column, no two variables should have same value.
i.e. For each value k, (-x(0,j,k) or -x(1,j,k)) and (-x(0,j,k) or -(2,j,k)) and..... should be true
4) In one Box(3X3), no two elemets should have same value same.
i.e. For each value k, (-x(i1,j1,k) or -x(i2,j2,k)) and (-x(i1,j1,k) or -x(i3,j3,k)) and ..... should be true, where (i1,j1), (i2,j2), (i3,j3), etc. all lie in the same box.
5) The constraints related to the values already present in the sudoku.