CSL452– Artificial Intelligence
Lab 3

Due on 11/3/2016 11.55pm

Instructions: Upload to your moodle account one zip file containing the following. Please do not submit hardcopy of your solutions. In case moodle is not accessible email the zip file to the instructor at ckn@iitrpr.ac.in. Late submission is not allowed without prior approval of the instructor. You are expected to follow the honor code of the course while doing this homework.

1. **You are allowed to work in teams of size at most 2 for this programming lab**.

2. A neatly formatted PDF document with your answers for each of the questions in the homework. You can use latex, MS word or any other software to create the PDF.

3. Include a separate folder named as 'code' containing the scripts for the homework along with the necessary data files.

4. Include a README file explaining how to execute the scripts.

5. Name the ZIP file using the following convention rollnumber1rollnumber2hwnumber.zip

---

1. Sudoku Solver using CSP

The goal for this assignment is to implement a Sudoku solver that treats the puzzle as a constraint satisfaction problem. Sudoku is a logic puzzle involving placement of digits in squares satisfying certain constraints. Learn more about the puzzle from the Wikipedia article - https://en.wikipedia.org/wiki/Sudoku

Your first goal is to formulate the puzzle as a constraint satisfaction problem. Identify the variables, domains and constraints that define the problem. Automatically convert the n-ary constraints into binary constraints.

The next step is to write a generic backtracking search without any heursitics. Use this backtracking search to find a solution to the Sudoku CSP. Let us call this search BS.

Now infuse intelligence into the backtracking search, by integrating the minimum remaining value heuristic. Let us call this search as BS-I. Implement and integrate the least constraining value heuristic into the BS-I search. Let us call this BS-II

Implement forward checking and integrate it into the BS-II and call it as BS-FC.

Finally, implement and integrate the MAC algorithm into BS-II. Call this algorithm BS-MAC.

Our goal is to study the performance of these algorithms on sample Sudoku puzzles. Define performance metrics (such as search time, memory utilization, number of backtracks) that will highlight the contrasting aspects of these algorithms. Include a PDF document in your submission that reports the performance of the different algorithms and your detailed analysis. Sample puzzles are provided in the p folder of the zip file.

Input
The input to your code will be the name of the text file containing all Sudoku puzzles. Each line in the text file will correspond to a single puzzle. The puzzles are all rasterized row-wise. Empty squares in the puzzle are represented as '.'

Output
Your code must output to a text file the solved puzzles that are rasterized row-wise. The solutions to the puzzles must appear in the same order as the puzzle in the input file.

2. Sudoku Solver using MiniSAT

In this part of the lab, formulate Sudoku as a Boolean satisfiability problem. The format of the inputs and outputs are defined as before.

You will use a standard SAT solver – MiniSAT to obtain the solution to the puzzle (if it exists). Learn more about the MiniSAT solver from here

http://minisat.se/MiniSat.html

MiniSAT requires input Boolean sentence to be in Conjunctive Normal Form (CNF). The input to the solver follows the DIMACS format defined as

$p\ cnf\ nVars\ nClauses$
$Clause 1$
$Clause\ 2$
$\vdots$
$Clause\ n$

The variables are to be numbered from 1 to $nVars$. The i^{th} variable is represented by the positive integer i; the negation of this variable is coded by the negative integer $-i$. A clause is

represented by listing the literals in the clause, separated by spaces and followed by 0. For example, consider the following sentence in CNF

$$x_2 \wedge (x_1 \vee x_4) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_4 \vee x_4)$$

This will be written in DIMACS format as

$p\ cnf\ 4\ 4$
$2\ 0$
$1\ 4\ 0$
$-2\ -3\ \ 0$
$-1\ -2\ 3\ 4\ 0$

More information about DIMACS can be obtained from
http://www.dwheeler.com/essays/minisat-user-guide.html

After writing the problem in the DIMACS format into a text file, you can run MiniSAT solver by the following command

/minisatdirectory/minisat InputFile OutputFile

If the input formula is unsatisfiable, then MiniSAT writes 'UNSAT' to the output file. If the formula is satisfiable, then it writes 'SAT' and a satisfying assignment.

The goal of your program is 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. The overall input and output format for your code is defined as before in part 1.

Include in the pdf document your approach to formulate the puzzle as a Boolean SAT problem. Specifically describe the process of converting constraints to Boolean sentence in CNF.

Evaluation

We will compare the output of your program on sample puzzles. Please follow the output formatting instructions as an automated checker will be used for comparison.