CSC242: Intro to AI
Project 2: Constraint Satisfaction

February 18, 2025

1 Overview

This project is designed to increase your understanding of constraint satisfaction (CSP) algorithms, by asking you to implement a Sudoku solver.

Constraint satisfaction is a deep, interesting area of computer science originating from the early days of AI research. A CSP is specified as a set of variables, domains, and constraints, and the objective is to find an assignment of values to each of the variables which simultaneously satisfies all the constraints. There are many applications, ranging from scheduling jobs on the Hubble telescope, to validating very large scale integrated circuits, to finding solutions to puzzles such as Sudoku.

The general problem of constraint satisfaction is NP-complete, however, with careful programming, many interesting cases can be solved efficiently on even very modest hardware. Like the first project, this assignment has three primary components:

1. Design and development of your Sudoku solver.
2. Experimentation and evaluation of solution.
3. A written report.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

(a) Puzzle

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

(b) Solution

Figure 1: Example Sudoku puzzle and solution. (Standard 9x9 game.)

2 Program Requirements

2.1 Sudoku

Background: Sudoku is a beloved puzzle which involves carefully assigning integers to locations on a grid-based board in such a way that each column and each row contain each integer from 1 to 9 exactly once. Furthermore, the overall grid is arranged as a 3x3 grid of 3x3 sub-grids, where each sub-grid also maintains the pattern that each integer appears exactly once. The general case involves kxk subgrids with a (k^2xk^2) overall board. To simplify notation, let $n = k^2$, so for standard Sudoku, n is 9. The general case is an interesting example of growth in computational complexity, as the total number of states grows as $O(n^{(}n^2))$; however, a candidate solution can be validated in $O(n^2)$.

For more information, and the source of this specific example, see https://en.wikipedia.org/wiki/Sudoku. For a deep selection of Sudoku strategies and puzzles, see https://www.sudokuwiki.org/Introduction.

2.2 Input and Output Format

We will use very simple text-based IO for this assignment, via `stdin` and `stdout`. Your program should read one puzzle, print the solution, then ter-

minate.

To streamline the format, we will only consider standard 9x9 sudoku. The input format will be a series of 9 lines, each line consisting of exactly 9 tokens, where each token will be a digit 0-9, with the digit 0 representing a missing value. Each token will be separated by a single space. The overall input will be terminated by a linebreak.

The output from your solver should be similarly formatted, outputting 9 lines of 9 tokens each, this time giving only digits 1-9 and forming a complete solution. If there are multiple solutions, you may print any solution. If there are no solutions, you should print "No solution."

```
INPUT:
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
```

```
OUTPUT:
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
```

2.3 Program Evaluation

Your program will be evaluated on a set of test cases which incrementally increase in difficulty, with more difficult puzzles providing fewer (and less

helpful) clues than easier ones. Your program will be limited to 1 seconds per test case on a VM with 4.0 CPU and 6GB RAM. The environment will be Ubuntu 22.04.

3 Report Requirements

The main challenge for this assignment is to choose an effective representation for your Sudoku boards. The representation will need to support extremely efficient enumeration of states. You are required to implement your solution from the perspective of constraint satisfaction. You must find a way to apply the AC3 algorithm, and to leverage variable and value selection heuristics. Note: the included test cases do not come with solutions, so you must also implement a reliable validator.

Your report should address the following topics:

1. Explain your design choices (e.g., data structures/objects) for representing sudoku boards and the correpsonding constraints; identify the computational complexity (time and space) of your overall implementation, as well as key methods.

2. How effective is an approach based on pure backtracking? (I.e., without AC3.) Can you solve all of the easy cases? What about the hard ones?

3. How effective is your approach based on the full solution (i.e., including AC3, variable selection, and value selection.) Can you solve more problems now? How would you characterize the change in runtime?

4. (Group submissions.) How did you work together as a team, and who worked on which components?

4 Submission Instructions

You may use Java, Python, C, or C++ for this assignment. In any case, you should write your solution as a single file named according to your chosen language: `Main.java`, `main.py`, `main.c`, or `main.cpp`. No other languages are supported for this assignment. Do not use any package declarations or modules.

Upload the source code of your solution and a PDF export of your report directly to gradescope. Ensure your name and UR email are clearly indicated at the top of your source file (e.g., as a comment), and as the top of your report. Group submissions should be submitted by one student who is responsible for tagging all group members in the gradescope submission. Group submissions should include the names and emails of ALL members in both the program source code and at the begining of the written report. The report must also contain a brief section indicating the contributions of each team member and a description of your collaboration style. Maximum of 4 students per group.

Submissions will be accepted until 11:59PM, Wednesday, March 5th.

4.1 Grading

- 70% Program Behavior (solving sudoku puzzles, test cases TBD.)
- 30% Report (addresses all questions listed above, clearly organized and formatted).

5 Academic Honesty

Please note that because this assignment is relatively straightforward, there are certainly many solutions online. Please refrain from searching online for Sudoku solvers, and similarly, do not prompt any AI system to generate a complete solution to this problem. The purpose of this assignment is for you - as a human - to creatively build and implement your solution, and to overcome any challenges which may arise along the way.

Submission of AI generated solutions, or solutions obtained from the internet, will be considered a violation of the academic honesty policy for this assignment.