CSCI 446 — Artificial Intelligence

Project #1
Search, Constraint Satisfaction, and Sudoku

The purpose of this assignment is to give you an introduction to artificial intelligence from the perspective of search. Being that this is a team project, several elements will need to be implemented.

Sudoku is a class of puzzles taking place on a 9x9 grid. In standard games, this is further subdivided into separate 3x3 sections. The start of the puzzle consists of a subset of numbers already filled into spaces on the grid. The goal of each puzzle is to finish filling the grid with numbers. However, there are three basic rules of number placement that constrains the puzzle. First, each row must contain all numbers, one through nine, with no repeats. Second, each column must also contain all numbers, one through nine, with no repeats. Finally, each 3x3 grid is also subject to the same constraints and must have all numbers, one through nine. An example puzzle and solution is shown in Figure 1.

For this project, you need to be able to read in a Sudoku puzzle and then solve it. More specifically, you will implement and compare different heuristics in combination with a backtracking search designed to solve Sudoku puzzles. Three separate iterations of the backtracking search must be implemented, with differing heuristics for each. You will then test the performance of these methods against sample puzzles.

The input files for this assignment utilize the same format. Each line of the sudoku puzzle is contained on a separate, single-spaced line. All items within the line are separated by spaces. For this assignment, a '?' indicates an unknown value and any number will indicate a known value. An example input file is shown in Figure 2. Any created puzzles utilized in this assignment for experimentation should be non-trivial. Thus, a nearly solved puzzle with only a few missing items should not be created as an example.

Here are the specific steps that need to be followed in completing this assignment:

- Implement a puzzle importer. This will use a standard format (described below) for your puzzles.
- Twenty different puzzles will be provided to you consisting of five easy, five medium, five hard, and five extremely hard. These should be read and processed separately based on the specified file name.
- Implement a constraint solver to solve each of the Sudoku puzzles, implementing the following variations:
  - Simple backtracking (only run this on easy and medium puzzles)
  - Backtracking with forward checking
  - Backtracking with arc consistency
  - Local search using simulated annealing with the minimum conflict heuristic
  - Local search using a genetic algorithm with a penalty function and tournament selection
- Write a design document that outlines the architecture of your software (include a fully explained UML class diagram), design decisions made in implementing your algorithms, and the experimental design for testing the results.

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

→

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

Figure 1: Sudoku Example

puzzle1.txt:

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

Figure 2: Example File

- Run experiments, keeping track of the main decisions being made for each algorithm. The count of these decisions will be used to gauge run time since CPU time and wall clock time are not reliable estimators.
- Write a paper that incorporates the following elements, summarizing the results of your experiments. Make sure you explain the experimental setup, the tuning process, and the final parameters used for each algorithm.
  1. Title and author names
  2. A brief, one paragraph abstract summarizing the results of the experiments
  3. Problem statement, including hypothesis (how do you expect the different algorithms to do?)
  4. Description of algorithms implemented
  5. Description of your experimental approach
  6. Presentation of the results of your experiments
  7. A discussion of the behavior of your algorithms, combined with any conclusions you can draw
  8. Summary
  9. References (you should have at least one reference related to each of the algorithms implemented, a reference to the data sources, and any other references you consider to be relevant)
- Create a video that is no longer than 5 minutes long demonstrating the functioning of your code. This video should focus on behavior and not on walking through the code. You need to show input, data structure, and output. How you do this is entirely up to you, but be sure it will convince the grader that your program works.
- Submit your fully documented code for the data converter, the video demonstrating the proper functioning of each algorithm, your design document, and your paper (which will include the results of the experiments). Note that you are welcome to provide a link to your video using a streaming service such as YouTube or Vimeo of you so choose.

Due Dates:

- Design document – September 8, 2021
- Fully documented program code – September 20, 2021
- Video demonstrating code functionality – September 20, 2021
- Project report – September 20, 2021