Assignment 2

Introduction to AI - CS 487, Fall 2021

Assignment 2

<u>**Deadline:**</u> Wednesday, 01/12/2021 on e-learn (`https://elearn.uoc.gr/`)

Deliverables: Submit a zip file containing a report in PDF with the answers AND all code files (included the .ipynb) written by you in the scope of the assignment. **Your deliverable should include your student ID in the filename**

Note that this assignment is to be done **individually**. Cheating in any way, including giving your work to someone else, will result in failing this assignment (a mark of zero will be given).

Theoretical Exercises

Exercise 1 (10 points)

How many solutions are there for the map-coloring problem in Figure 6.1 (using 3 colors)? How many solutions if four colors are allowed? Two colors? Explain the solution in detail.

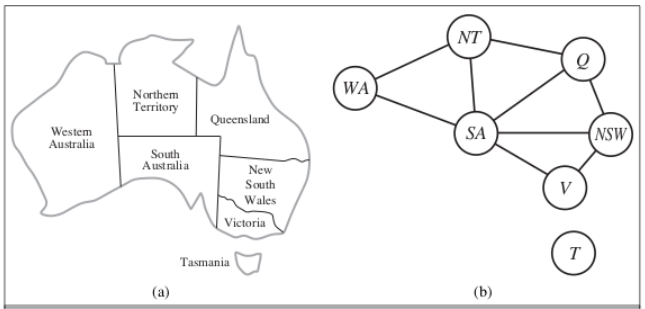

Figure 6.1 (a) The principal states and territories of Australia. Coloring this map can be viewed as a constraint satisfaction problem (CSP). The goal is to assign colors to each region so that no neighboring regions have the same color. (b) The map-coloring problem represented as a constraint graph.

Exercise 2 (30 points)

You are in charge of scheduling the meetings for the oral examination of CS487 project. There are 5 students that delivered the project and 3 teaching assistants who will examine them. You are constrained by the fact that each teaching assistant can only examine one student at a time.
The students along with their appointments are:

- Student 1: 8:00-9:00am
- Student 2: 8:30-9:30am
- Student 3: 9:00-10:00am
- Student 4: 9:00-10:00am
- Student 5: 9:30-10:30am

Note that teaching assistants can only examine students that implemented a project related to their areas of expertise.

- TA A is available to examine Students 3 and 4
- TA B is available to examine Students 2, 3, 4 and 5
- TA C is available to examine Students 1, 2, 3, 4 and 5

1. Formulate this problem as a CSP problem in which there is one variable per student, stating the domains, and constraints. Constraints should be specified formally and precisely, but may be implicit rather than explicit.
2. Draw the constraint graph associated with your CSP.
3. Show the domains of the variables after running arc-consistency on this initial graph (after having already enforced any unary constraints).
4. Give one solution to this CSP.
5. Your CSP should look nearly tree-structured. Briefly explain (one sentence or less) why we might prefer to solve tree-structures CSPs.
6. Name (or briefly describe) a standard technique for turning these kinds of nearly tree-structured problems into tree-structured ones.

Programming Exercises

Exercise 3 (60 points)

In the csp.py file of the book code there is an implementation and modeling of the well-known mathematical puzzle Sudoku 9x9 as a problem of constraint satisfaction. In this exercise you will run appropriate algorithms for its solution. You can use the code of the book given to you or implement your own. Reminder: If you use third-party code beyond what is given, it is necessary to clarify the sources! (You need to put the files given to you to the aima-search folder, given to you in the previous assignment)

Select 50 random sudoku problems from the given (sudoku_puzzles.py). For each problem run at least four (4) different combinations of algorithms and heuristics at least ten (10) times each. Measure the average execution time and

solution percentage (evaluate with a penalty of 10 for each empty box and a penalty of 20 for each collision of restrictions). Note: Because some algorithms may take too long to solve a problem, you are free to limit the solution (even by modifying the algorithms accordingly) by clearly stating the termination criteria in your report. It does not matter if the algorithm fails to find a solution (it has reached the maximum limit of the situations it can visit, it has reached the maximum limit of the memory it can use or for some other reason), consider the final sub-solution as final by assigning the aforementioned penalty. **What conclusions do you draw from the results based on what you know about the algorithms and heuristics you used? Which algorithm is more suitable and why? Explain your results thoroughly using the theory** (Tip: Use plots to visualize your results).
Choose one of the algorithms you used and solve the 16x16 problems described in (Assignment3.ipynb).

(Bonus) Solve the 100x100 Sudoku puzzle. Also, extra bonus will be given to the students that achieve the top 5 scores.

(Bonus 2) In this assignment we were able to solve some sudoku puzzles. Is there any way to produce Sudoku puzzles with a single solution and parameterized size?

Important Note: Due to limited memory usage from the jupyter notebook do not limit yourself to implementing .ipynb files but also try normal .py files and run them through eg pycharm or terminal window