Assignment 3: Constraint Satisfaction

Worth: 7.5% of total grade
Due Date: Friday October 9th 2020 23:59pm
For assignment 3a submit upi.py and
for assignment 3b submit upi.pdf respectively
to the CANVAS dropbox.

Download the constraint satisfaction code that comes with the assignment. This zip contains a number of python files required to run the csp code, but you will only need to modify *instrumented_solvers.py*. The test.py file has tests you can run against your code to make sure they work correctly, and to generate the dataset you will analyse in Part B. For full marks in Part A your code will need to pass the provided tests, as well as continue to perform correctly on some additional problems. Make sure that you name your python code <your upi>.py when submitting it!

Part A

Complete the backtracking_search_instrumented and min_conflicts_instrumented functions found in instrumented_solvers.py, so that they output a dictionary containing the same result as the uninstrumented versions found in textbook/backtracking_search_solver.py and textbook/min_conflicts_solver.py, as well as some additional information. Consult the function skeletons provided, and test.examples for the exact output format required.

Task 1 [1 mark]

Instrument the backtracking search code so that it outputs, the number of backtracks and the number of assignments made during different csp settings. Look at the test cases to verify the output.

Task 2 [1 mark]

Instrument the min conflicts code so that it outputs the total number of variable assignments performed, and the number of assignments made during the repair section of the code. Look at the test cases to verify the output.

Part B

Task 2 [5.5 marks]

Part B should be a maximum of 2 pages including the table.

You will need to run a set of experiments with the following settings:

1. Backtracking Search, First Unassigned Variable, Unordered Domain Values, No Inference
2. Backtracking Search, First Unassigned Variable, Unordered Domain Values,

Forward Checking
3. Backtracking Search, First Unassigned Variable, Unordered Domain Values, Maintain Arc Consistency(AC3)
4. Backtracking Search, Minimum Remaining Value, Least Constrained Values, No Inference
5. Backtracking Search, Minimum Remaining Value, Least Constrained Values, Maintain Arc Consistency(AC3)
6. Minimum Conflicts Solver
The function get_solvers in test.py can be called to return a dictionary containing these 6 solvers.

You will need to run the following datasets: NQueens (10,30,50,70,90), map colouring for USA and Australia, and the zebra problem. The function get_assignment_problems in test.py can be called to return a dictionary containing these problems.

Each solver should be run 10 times, with max_steps set to 1000. The function get_assignment_results can be called to return a dictionary in the form {solver: {problem: results}}. After implementing your instrumented versions of the csp solvers, it is advised that you use this function to generate your data, as it will take care of setting a random seed so that the dataset generation is deterministic.

Complete the table that includes:
1. the average number of assignments made when solving the problem (for Minimum Conflicts you will need to record two numbers for this, ie: total number of assignments, repair assignments) plus the standard deviation
2. the average amount of time used when solving the problem plus the standard deviation
3. the probability that each of these will solve the problem and
4. the number of backtracks

The table should look as below:

# queens or Map for 4-colour	method	Average # of assignments +/- std dev	Average time to solve +/- std dev	Probability of solving	# of Backtracks
n queens / map name					
	1				
	2				
	3				
	4				
	5				
	6	total assignments, repair assignments			N/A

Australia Map					
	1				
	2				
	3				
	4				
	5				
	6	(both counting initial assignment and without)			N/A
USA Map					
	1				
	2				
	3				
	4				
	5				
	6	(both counting initial assignment and without)			N/A
Zebra					
	1				
	2				
	3				
	4				
	5				
	6	(both counting initial assignment and without)			N/A

In the report you should discuss
a) Which settings you think work best for the constraint satisfaction algorithm and whether they are specific to certain problems or are always better or worse. Why do you think they work best?
b) Talk about the growth of the number of assignments and time as the N-Queens problem and the Map-Colour Problem grow. Do the two domains act the same? If not why not?
c) When would you use Minimum Conflict instead of a Constraint Satisfaction Solver? Why?
d) How does Minimum Conflict (which is essentially a local search algorithm) Perform Compared to the Local Search Algorithms you used in Assignment 2? (You will need to run them on the same NQueens Size Problem to determine this)

e) In some problems there might be a large number of assignments, but no backtracking (or very little), why would this occur?

Report Rubric:
3 points for the table
2.5 points for the report