CPE/CSC 480-F05 Artificial Intelligence Lab Exercise 5

Name:	
Status	Final
Points:	10
Deadline:	Tuesday, Oct. 25, end of lab

This lab exercise will use another tool in the Computational Intelligence Lab at UBC in Vancouver (http://www.cs.ubc.ca/labs/lci/CIspace). As with the previous exercise, you can use the online version, or download the code, and run it locally on your machine as a Java application.

The topic of this exercise is local search, and we will use the *n-queens problem* to investigate some local search methods.

Constraint Satisfaction

Instructions

Invoke the *Constraint Satisfaction* applet, and load the sample "Five Queens Problem". Look at the graph displayed, and determine the relationship between the conventional representation of the queens problem on a chess board, and this one. Then use the tool to solve the problem. You can do this either in "Auto-Solve" mode, or manually: Select values for a variable, and then let the tool calculate compatible values for the other variables. This is done with the "Auto Arc Consistency" button. If you made a choice that does not have any possible solution, you can backtrack, and select different values. You can speed up the process by selecting "Very Fast" for the Arc-Consistency Speed in the CSP Options.

Once you're comfortable with the tool, try your luck on the eight queens problem. This will take longer, but even on my relatively slow Mac iBook it was tolerable, and I could generate 30 solutions in a few minutes.

Questions

- How is the queens problem represented in the constraint satisfaction (CS) graph? What is the meaning of the components that make up the CS graph:
    - nodes
    - links
    - labels on the links
- How is it guaranteed that the queens do not threaten each other
    - horizontally,
    - vertically, and
    - diagonally?
    -
- Is this representation easy for humans to follow?
- Write down at least two of the solutions you found for the five queens and two for the eight queens problem.
- How many solutions did you find for the five queens problem?
- How many for eight queens (you don't need to examine all of them)?
- Does this method use an incremental or a complete-state approach? Explain your answer!
- Is this method complete? Is it optimal? If so, under which circumstances? If not, what causes it to be incomplete or non-optimal? What does optimal mean here?
- Is this a good method to determine how many solutions there are for the n-queens problem?

Hill Climbing

Instructions

Invoke the *Stochastic Local Search* applet (`hill.jar`), and load the sample "Five Queens Problem". Select the "Greedy Descent" method, initialize the graph, and press the "Autosolve" button. You can view additional information about the progress on the "Plot" and "Trace" panes. The "Batch" button will initiate multiple runs of the sample; it is initially set to 100 runs.

After experimenting with the regular greedy descent method, try out some of the modifications to see the effects they have. The most relevant for this type of problems is probably the "min-conflicts" heuristic.

Finally, try out the "Simulated Annealing" method. For this one, I recommend to observe the plot, and do some batch runs. You can also try to increase the number of steps.

Questions

- Do these method use an incremental or a complete-state approach?

- Write down at least two of the solutions you found for the five queens and two for the eight queens problem.
- Is this a good way to determine how many solutions there are for the n-queens problem?
- Are these methods complete and optimal? If so, under which circumstances? If not, what causes them to be incomplete or non-optimal? What does optimal mean here?
- Did you notice any significant differences between the various versions of the greedy descent method?
- How did the simulated annealing method perform?