

Assignment Two: Variables and Constraints Solutions

In the first three questions you will look at backtracking, arc consistency, and hill climbing for solving the same CSP problem. These problems will be easier if you use the CIspace applets at <http://www.cs.ubc.ca/labs/lci/CIspace/>

Question One

Consider a scheduling problem, where there are six variables A, B, C, D, E and F , each with domain $\{1, 2, 3, 4\}$. Suppose the constraints are: $A < B, |A - C| = 1, B - C$ is even, $B \neq D, D > A, D \neq C, E \neq C, E < D - 1, E \neq B - 2, A \neq F, B \neq F, C \neq F, D \neq F, |E - F|$ is odd.

A CIspace representation for this graph is at:

<http://www.cs.ubc.ca/spider/poole/cs502/2006/cspexample.xml>

- (a) Show how backtracking can be used to solve this problem, using the variable ordering A, B, C, D, E . To do this you should draw the search tree generated to find all answers. Indicate clearly the satisfying assignments.

To indicate the search tree, write it in text form with each branch on one line. For example, suppose we had variables X, Y and Z with domains t, f , and constraints $X \neq Y, Y \neq Z$. The corresponding search, with the order X, Y, Z , tree can be written as:

```
X=t Y=t failure
    Y=f Z=t solution
        Z=f failure
X=f Y=t Z=t failure
        Z=f solution
    Y=f failure
```

Hint: the easiest way to solve this problem may be to write a program to generate the tree (using whatever programming language you like).

- (b) Is there a smaller tree? Give a node selection heuristic that results in as small a tree as you can find. Show the tree, and say how many leaves there are. Explain why you expect the tree resulting from this heuristic to be good. (A good explanation as to why your ordering is expected to be good is more important than the right answer.)

Solution: Here is the tree generated by the Java or Prolog programs available from the web site.

```
A=1 B=1 failure
    B=2 C=1 failure
        C=2 D=1 failure
            D=2 failure
                D=3 E=1 F=1 failure
                    F=2 failure
                        F=3 failure
```

```

        F=4 success
        E=2 failure
        E=3 failure
        E=4 failure
    D=4 E=1 F=1 failure
        F=2 failure
        F=3 failure
        F=4 failure
        E=2 failure
        E=3 failure
        E=4 failure
    C=3 failure
    C=4 failure
B=3 C=1 failure
    C=2 failure
    C=3 failure
    C=4 failure
B=4 C=1 failure
    C=2 D=1 failure
        D=2 failure
        D=3 E=1 F=1 failure
            F=2 failure
            F=3 failure
            F=4 failure
            E=2 failure
            E=3 failure
            E=4 failure
        D=4 failure
    C=3 failure
    C=4 failure
A=2 B=1 failure
    B=2 failure
    B=3 C=1 D=1 failure
        D=2 failure
        D=3 failure
        D=4 E=1 failure
            E=2 F=1 failure
                F=2 failure
                F=3 failure
                F=4 failure
            E=3 failure
            E=4 failure
    C=2 failure
    C=3 D=1 failure
        D=2 failure

```

```

        D=3 failure
        D=4 E=1 failure
            E=2 F=1 success
                F=2 failure
                F=3 failure
                F=4 failure
            E=3 failure
            E=4 failure
    C=4 failure
B=4 C=1 failure
    C=2 failure
    C=3 failure
    C=4 failure
A=3 B=1 failure
    B=2 failure
    B=3 failure
    B=4 C=1 failure
        C=2 D=1 failure
            D=2 failure
            D=3 failure
            D=4 failure
        C=3 failure
        C=4 D=1 failure
            D=2 failure
            D=3 failure
            D=4 failure
A=4 B=1 failure
    B=2 failure
    B=3 failure
    B=4 failure

```

To find a smaller tree, a good heuristic is to always choose the most constrained variable.

You get a bonus mark for having a non-uniform tree. That is, where which variable chosen depends on the context.

Question Two

Show how arc consistency can be used to solve this problem. To do this you need to

- Draw the constraint graph,
- For the first 5 instances of arc consistency, show which elements of a domain are deleted at each step, and which arc is responsible for removing the element.
- Show explicitly the constraint graph after arc consistency has stopped.
- Show how splitting domains can be used to solve this problem. Draw the tree of splits and show the solutions.

- (e) Based on this experience, discuss how much arc consistency saves over the backtracking for this problem (even for the best tree that you found).

Solution: Parts (a)-(d) can be done with the applet. You were to write it out just to make sure you understood what was going on.

Question Three

Show how hill climbing can be used for the problem of question 1. (You can use the CIspace hill climbing applet for this).

- (a) For one particular run, where you select any variable that is involved in an unsatisfied constraint, and select a value that results in the minimum number of unsatisfied arcs, explain which element is changed at each step and what was the resulting number of unsatisfied arcs (You only need to do this for 5 steps).
- (b) Compare and explain the result of the following settings:
- i) select a variable with the maximum number of unsatisfied constraints, and the best value
 - ii) select any variable which is involved in unsatisfied constraints, and the best value
 - iii) select a variable at random, and the best value.
- (c) How important is to choose the value that results in the fewest unsatisfied arcs as opposed to choosing a value at random? Explain.
- (d) Based on this experience suggest what good settings for the parameters are. Explain your choice.

Solution:

- (a) This is what the show trace gives you. Just make sure you understand it.
- (b) You need to refer to the runtime distribution to get full marks. (a) works better than (b) for the first 60 steps, but can only solve about 70% of the runs. (b) can solve all of the runs. (Set “terminate after” to be about 1000 to see this). (c) can solve all of the runs, but is dominated by (b).
- (c) For each of these, choosing a random value works much worse than choosing the best value.
- (d) In terms of steps, choosing the best node and value, with a random restart after about 6 steps works really well. Choosing the best node-value pair works well (which for some reason doesn’t seem get stuck in a local minima in this example). These are not so good in term of time though. In terms of run time choosing the best variable say 60% of the time and a random “red” node 40% of the time, with the best value works well. Perhaps you did better with some other settings.

Question Four

In the SLS applet, it is difficult to compare simulated annealing with the other algorithms. Explain why. What would you recommend to do to make the comparison fair?

Solution: The problem is that simulated annealing doesn't have a well defined notion of a "step". Simulated annealing suggests a step, then either accepts or rejects it. You could either count suggested steps, or accepted steps. The first isn't fair to simulated annealing because it can check many suggestions in the time another algorithm does a single step. The second is problematic when the temperature is low and the algorithm is at a local minima; in this case it can run for a long time without making a step. This is even problematic when you stop after, say, 100 steps and simulated annealing doesn't do any steps in any reasonable time (so then it is even difficult to compare run times).