CSEP 573 Applications of Artificial Intelligence Winter 2011 Assignment 2

Due: Wednesday February 2, 6:30PM

Q 1: [9 points] Your boss woke up in the middle of the night with the bright idea of using local search algorithms for sorting N distinct integers, where N is a fixed number. You've been asked to test this idea.

The state representation you have decided to use is simply a sequence of N integers; a successor of a state is obtained by swapping any two adjacent elements in the sequence. For example, the successors of (13, 12, 19, 2, 5) are (12, 13, 19, 2, 5), (13, 19, 12, 2, 5), (13, 12, 2, 19, 5) and (13, 12, 19, 5, 2).

q1.a: **[3 points]** Formulate a linearly computable objective function for this problem to use with a local search algorithm.

q1.b: **[3 points]** Does your objective function have any local maxima or just a global maximum? Explain your answer.

q1.c: **[3 points]** Does your objective function have any shoulders? Explain your answer.

Q 2: [12 points] You are given two boxes 1 and 2, that can hold W1 and W2 lbs each. There are n items with weights w(i) lbs respectively. Additionally, each item has a value v(i) dollars. You need to figure out which item should be added to which box so that the total value is maximized. Note that the weight constraints of the boxes should not be violated. Formulate the problem using genetic algorithms.

q2.a: **[3 points]** Design a string representation of a state

q2.b: **[3 points]** Design a fitness function

q2.c: **[3 points]** Design a crossing over operator

q2.d: **[3 points]** Design a mutation operator

Q 3: [15 points] Take the minimax search tree linked here (svg - png - dot) and for each node show the final value (if the node will be evaluated) or put a cross (if the node will be pruned using alpha-beta search).

Q 4: [4 points] Look at the Slide 71 of the slide-deck for adversarial search. It shows a partial expecti-minimax tree for an adversarial game with chance nodes. Fill the values in each node (as much as you can given the information provided).

Q 5: [10 points] Several kinds of symmetries are exploited in constraint satisfaction problems to reduce the size of the search space. This happens in two steps.

First we recognize that two or more states represent the same structure in the problem, and hence will behave similarly (i.e., will either both have a solution or both won't have a solution). As an example, consider an 8-queens representation of the problem where Xi represents the location of queen in the i^{th} row. The board configuration B=(X1,...,X8) is symmetric to configuration Bh=(X8,X7,...,X1) because Bh is a mirror image of B1 along the horizontal axis.

Second, to break this symmetry we need to add constraint(s) so that only one of the symmetric configurations will be explored by the algorithm. For example, in the previous example we can add an additional constraint X1<=X8. Verify that this constraint allows it to break configurations symmetric along the horizontal axis.

In this question consider a similar symmetry along the vertical axis of the board.

Q 1: [3 points] give an example of two configurations (B and Bv) from 8-queens that are symmetric along the vertical axis.

Q 2: [3 points] give an algebraic representation of Bv if B is (X1,...,X8)

Q 3: [4 points] add constraint(s) so that only one of B and Bv are explored by the CSP algorithm.

Q 6: [Extra credit points]

q6.a: Prove that Borda Count protocol satisfies the monotonicity criterion.

q6.b: Prove that Copeland protocol satisfies the pareto criterion.

Q 7: [50 points] Programming Project. We will solve Graph Subset Mapping problem using SAT solvers (miniSAT).

Description of the Programming Project

The problem can be defined as follows. There are two directed graphs G and G'. You need to check whether G is present inside G'. In other words can you create a one-one mapping M between all nodes of G to some nodes of G' such that there is an edge from v1 to v2 in G if and only if there is an edge from M(v1) to M(v2) in G'. Sample cases are shown here.

We will use miniSAT, a complete SAT solver for this problem. Your code will read two graphs in the given input format. You will then convert the mapping problem into a CNF SAT formula. Your SAT formula will be the input to miniSAT, which will come back to you with a variable assignment that satisfies the formula (or an answer "no", signifying that the problem is unsatisfiable). You will then take the SAT assignment and convert it into a mapping from nodes of G to nodes of G'. You will output this mapping in the given output format.

Generate problems of increasing hardness. You can vary sizes of G and G' (you are being provided a generator for the problem). Study the scaling of miniSAT with problems of increasing size. Also study phase transitions using this problem. Vary #clauses/#variables. See if you see a steep jump at the critical value near 4.3.

Formats and Code

Your code should compile, if necessary, and run on attu.cs.washington.edu. Please supply a `compile.sh` script if compiling is necessary. You should supply a shell script called `run.sh` that takes one input file and generates 3 output files. The input file should be in the format produced by the provided random graph generator. Nodes are represented by positive integers. Each line represents an edge from the first node to the second. Both graphs are presented in the single file, the larger first. The line with "0 0" is the boundary between the two. An example is shown to the right. This example could represent the last example shown in the sample diagrams.

```
1 2
1 3
1 4
2 4
3 4
0 0
1 2
3 2
```

If a call is made to `./run.sh test1`, then your code should take the input file `test1.graphs` and produce:

`test1.satinput` - the input file for minisat.

`test1.satoutput` - the output file from minisat.

`test1.mapping` - the mapping from small to large graphs

The .mapping file should be in the format shown on the right. The first numbers on each line represent a node as numbered in the smaller graph, and the second number represents the node of the larger graph to which it is mapped.

```
1 2
2 4
3 3
```

The formats of the other two files are determined by the minisat program.

The Minisat Page

MiniSAT User Guide

Random Graph generator

Samples diagrams for graph subsets.

Grading for the Programming Project

[15 points] accurate coding as verified by testing your turned in code on some test cases. The tests will use scripts, so exact naming conventions are important.

[15 points] a writeup on your conversion of the graph subset mapping into SAT, and backconversion of SAT output to the mapping.

[20 points] Your evaluation/explanation of scalability and phase transition results.

[Extra credit points] Take a local search solver (such as WalkSAT or any other). Evaluate the differences between WalkSAT and miniSAT with respect to scalability and phase transitions.

[Extra credit points] if your code can solve problems that others cannot.

What to Turn In

For questions 3 and 4, turn in a single sheet of paper showing the search trees with values (for both) and pruning (for 3) marked. Make sure your name is on the paper.

For the other questions, use the Class Dropbox to turn in a single file: "a2submit.zip". This should contain the a2submit directory with a single .pdf file called assignment2Report.pdf and all of your code for the graph subset problem.

Your code should include at minimum the `run.sh` file, and the `compile.sh` file if necessary. Do not include any pre-compiled files. Your code should compile and run in the attu.cs.washington.edu environment. (Note that attu uses Python version 2.6.4 - newer features might not work)