CS440

Assignment 3 (due Tue 10/7 at 5pm)

Solving the MAX-SAT problem with local search algorithms [80 pts]

Let's start with a few definitions:

Our focus in this question is on *Boolean formulas*. A Boolean formula is constructed using variables that take on the values `True` or `False` and the operators AND (conjunction, denoted by $\wedge$), OR (disjunction, denoted by $\vee$), and negation (denoted by $\neg$). We need a few more definitions: A *literal* is either a variable (a positive literal), or the negation of a variable (a negative literal). A *clause* is a disjunction of literals or a single literal. A formula is in *conjunctive normal form* (CNF) if it is a conjunction of clauses or a single clause. For example, the formula $(x_1 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee \neg x_4)$ is in CNF. The formulas we will use in this assignment all have 3 literals per clause (3-CNF). The satisfiability problem is the problem of finding whether there is an assignment of truth values to the variables which satisfies a formula in CNF, i.e. makes the formula evaluate to True. This problem is known to be NP-complete. We'll consider a more general problem, which is MAX-SAT: finding an assignment that maximizes the number of satisfied clauses (a clause is satisfied if there is an assignment of truth values for which it evaluates to True. This problem is also NP-complete.

Your task is to solve the MAX-SAT problem with local search strategies. To do that, write a class called `MAXSAT` which is a sub-class of `search.Problem`. The constructor of your `MAXSAT` class should receive a file name as a parameter. This file will encode the instance of `MAXSAT` in a format described here: `http://www.satcompetition.org/2009/format-benchmarks2009.html`. On the assignment page we provide a set of 20 instances of MAXSAT that is a small subset of the instances at `http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html`.

Your task is to compare the following solution strategies:

- Steepest ascent.
- Steepest ascent with random restarts (use 20 restarts).
- Genetic algorithms.
- Simulated annealing.

Use the same method for choosing the initial state and neighborhood of a state for all strategies (for GAs you will need crossover and mutation operators). To compare the search strategies run each of them on the instances of MAXSAT we provide (link on the assignment page) and compute the average value of the solutions and running time. Describe in detail your choice of initial state and neighborhood and the details of your GA crossover and

mutation operators. Compute the average quality of the solution and the running time for each strategy, treating instances of 50 variables and 150 variables separately. Explain the difference you observe in performance (how good is the state returned by the algorithm, and how long did each strategy take to run).

In your code provide a method called `run_maxsat()` that reproduces your results and outputs the results to a text file called `maxsat_results.txt`. On Piazza I have posted an example of how to solve the Eight Queens problem with local search, which you can use as a reference point. For timing python code you can go as simple as:

```
import time
t0 = time.time()
code_block
total = time.time() - t0
```

or you can use the `timeit` module.

Heuristics for the Huarong Pass Problem [30 pts]

We have seen that BFS has great difficulty in finding solutions to the Huarong Pass problem, and DFS finds solutions that are often sub-optimal in terms of the number of moves. In this question we will consider the use of A* with an appropriate heuristic to solve the problem. Design at least two admissible heuristics for solving the problem and run A* to see how well they are doing. In your writeup you need to:

- Describe each heuristic, explain why it is likely to be useful, and prove that it is admissible.
- Compare the running time and number of nodes expanded during the search process.
- Explain the results. If one heuristic worked better, explain why you think that happened.

In your code provide a method called `run_huarong_pass()` that reproduces your results and outputs the results to a text file called `huarong_pass_results.txt`.

Submission Prepare a module called `p3.py` and submit it via checkin as P3. At the top of the module have a comment in triple quotes that identifies you and the assignment. Your writeup describing your experiments and results should be in pdf format in a file called `a3.pdf` and submitted via checkin as A3. It should include your name and eid. You will be graded primarily on the basis of the writeup; the code is for us to refer to if we would like to see more details of your implementation.