CMPSCI 683 Fall 2010

Prof. Victor Lesser October 18, 2010

General Comments: I will grade the exam based on 91 points total rather than 100 points. So many people were unprepared to answer question 1F. The grade on the answer to this question can be thought of as extra credit.

MID-TERM EXAM

The answers to these questions should be specific and to the point; we are not looking for essays! There are two types of questions: 6 short essay like questions (each worth 9 points for a total of 54 points) and 2 long questions (one worth 22 points and the other worth 24 point for a total of 46 points). **Please be careful about timing**!

1. Short questions (54 points)

A. (9 points) Suppose you had two admissible A* heuristics (h1 and h2) for a specific problem application and there was respectively cost (c1 and c2) every time you applied the heuristic in a search. How would you go about deciding which heuristic to use for the entire class of problems?

*Run experiments on a number of comparison cases using each of the heuristics to get the average time for each search with different heuristics. The heuristic whose average search time over the set of examples that is the lowest would be the one chosen. Another way of thinking about it would be E of h1 (average number of nodes expanded) *c1; and similarly for h2.. Another way to do it, is get the average number of nodes expanded in each search. Then it would be E of h1 (average number of nodes expanded) *c1; and similarly for h2. The formula that gave the lowest value would determine what heuristic to choose. Obviously, if h2 dominates h1 and the cost of applying h1 (c2) has lower cost than c2, you would choose heuristic h1 for all problems and no experimentation is necessary. In my answer, I did not think of trying experimentally the case of max (h1,h2) which incurs the cost of applying both heuristics to each search node expanded. However, in some cases that could be the best choice but I suspect it is very rare. Further in considering which heuristic to choose for a class of problems, I was only considering which would lead to the smallest expected search cost for solving a problem. Another criterion could be to minimize the number of nodes expanded, this would lead to a slightly different reasoning about which heuristic to choose.*

B. (9 points) What are the similarities and differences between Anytime A* and RTA*?

Both are doing an approximate search given a fixed amount of time that can be used. They both exploit an admissible and monotonically increasing h heuristic. However, their search strategies are very different. RTA* use a limited depth-first searches to get a better approximation of a node's f value to make a decision what operator to apply. It then applies after each search the chosen operator in the real world and then repeats the procedure to choose the next operator after the move is completed. Anytime A* in contrast is doing a complete search trying to get an acceptable solution quickly and then over time improving the solution. When it is finally terminated either because of time limits or an optimal solution is found, the best (lowest cost) complete path/plan that has been encountered is chosen.*

C. (9 points) What are the similarities and differences between SMA* and RBFS?

Both exploit the fact that f is monotonically increasing and there is a remembrance of the f values of previously encountered partial solutions to focus what node should be next (re)expanded; they also both are trying to reduce the amount of memory necessary for the search, and for that reason both may generate repeatedly the same node. In the case of SMA, it deletes nodes due to fixed memory limitations while RBFS may delete nodes because it keeps a*

very restricted open list based on a depth-first type of search. SMA needs to have as much memory as the length of the optimal path otherwise it will be able to find this optimal path.*

D. (9 points) Explain the common reason/principle for the use of the techniques of beam search in Genetic Algorithms and random restart in GSAT. Could you apply beam search to GSAT?

Both search techniques are trying to avoid getting stuck in local minima. The beam search has the potential advantage over random restart since it is able to constantly readjust what solutions are in the beam according to the quality and potentially the diversity of these solutions, and to be able to take parts of one solution and combine with parts of another solution in the beam to create a new solution. Maybe, the beam search could be applied to GSAT, it is interesting question of how often in GSAT do you need to do random restarts versus paying the overhead of concurrently processing multiple solutions. To really exploit the beam search idea in GSAT, you would in some sense need to alter the basic search strategy of GSAT so that there was more than one next solution generated at each iteration. In this way, at each stage, the beam could be narrowed back to k width based on "fitness" of the current solutions in the beam.

E. (9 points) What would be your explanation for why GSAT does

not exploit a specialized procedure to generate a “good” initial assignment for the truth values of the literals?

One possible explanation is that the cost of getting a good initial solution is quite expensive and it is better just searching based on a random initial solution and if that is not progressing well just try another random initial solution. It also may be that there are no general heuristics for a getting a good initial solution for an arbitrary problem though there may be good heuristics for a specific class of problems.

F. (9 points) The HEARSAY-II speech understanding system as described in class is not based on the A* search because of the difficult of constructing an admissible and effective heuristic. However, it uses a termination procedure resembling Anytime A*. When Hearsay-II search found a complete solution that was above a certain rating, it could prune partial solutions (nodes) on the blackboard based on calculating a measure using all the words that had been constructed either through bottom-up or top-down processing at the point that a complete solution was generated. Explain the basis for the pruning and also why this approach could potentially lead to incorrectly pruning a correct partial solution though we never saw an example of this.

Based on an analysis of the word lattice, a measure can be constructed for the highest ranking word in each segment of the speech signal. This rating can be used to construct the "highest" possible score that a partial solution could get when it is completed. This is not totally accurate because in expanding a partial solution, it is possible that new higher rated words could be generated as a result the top-down word verification process. For this reason, the heuristic is not admissible and thus could lead to pruning of a partial solution that could have created a higher score than the current best solution.

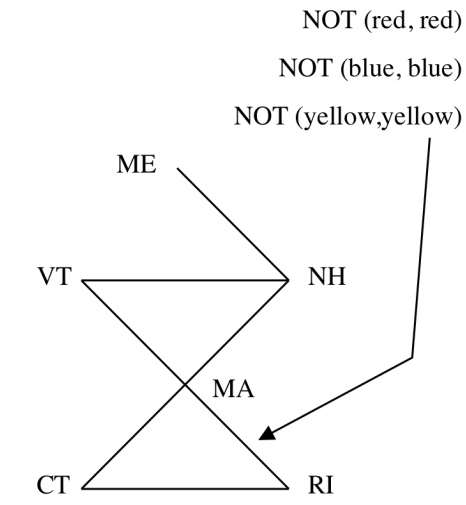
2. Long Questions (46 points)

A. (22 points) Sketch out an algorithm for bi-directional A*. As part of the sketch you should discuss why your algorithm will always find the minimal cost solution.

In order to do this problem, I would need to have both a heuristic admissible function that worked for both directions and obviously a well defined goal and start state (for example in route finding problem the city I am starting at the and city that I am going to) and appropriate operators for going in both directions. Obviously if you were doing the route finding you could do the search in

both directions using the same operators and heuristic function. Additionally the cost g between two directly connected nodes should be the same no matter what direction you are coming from. *There are two issues that must be resolved. First is how do I make a decision about which direction to next proceed. I would have two open lists one for each direction. I would choose for the node to next expand which has the smallest f value on either list. This way if I expand a node on backward search which is the smallest f and it is the initial state I have found the lowest cost solution and vice versa. I also have to understand how to handle the situation where in expanding a node one or more of its successors is on the other direction's open list. In that case you can combine the two paths and generate a new node on the open list of the node that was a complete path with appropriate cost. Like A* generating a complete solution does not mean you can immediately terminate the search, you need to wait until this solution is taken off the open list to make the decision that this is the minimal cost path. However, if the node was on the other agent's closed list then you could immediately stop.*

B. (24 points) Consider the following graph-coloring example.



3-color map coloring New England

It involved six nodes (CT, MA ,ME, NH, RI, VT) with the following adjacency links ME-NH, VT-(NH,MA), (VT,MA), MA-(VT,NH,CT,RI), CT-(MA,RI) and RI-(CT,MA). Each node can take on one of three colors (red, blue, yellow) and no nodes that are adjacent can be assigned the same color.

B.1 (8 points) Sketch out very briefly how this problem can be translated into an N-SAT problem in order to perform a stochastic search. You do not need to do the full translation!

For each node (state) in the graph there would be three literals. For example CT-red, CT-blue and CT-yellow. You would then have clauses indicating the one and only one of those literals is true. Similar to the mapping of the n-queens problem. You would then have clauses indicating the constraints among nodes. For instance there would be clauses indicating that if CT-red is true

then MA-red needs to be false and RI red needs to be false; this would require multiple 2-literal clauses to express this ((not CT-red) OR (not MA-red)) AND ((not CT-red) OR (not RI-red))

B.2 (8 points) How would you formulate it as a systematic constraint satisfaction search? Give representative examples of the different types of constraints.

I would have a variable associated with each node (e.g, CT) in the map whose domain of values include red, blue and yellow. I would then have a set of pairwise constraints (such as CT not equal to MA) for each node in the map that is directly connected with another node. I would use min-conflict heuristic search paradigm.

B.3 (8 points) If you had a larger graph, coloring problem, let us say the entire map of the US which has 50 states, which search approach (systematic or stochastic) would you use. Briefly explain your reasoning!

I don't think there is an obvious answer since using the mini-conflict heuristic search at least for the N-queens problems is in the same ballpark as a stochastic search. I would see first whether I could find a good and cheap way to generate a heuristic starting solution. Probably, if that was the case, I would go with the systematic search otherwise stochastic search.