CPS331 Lecture: Heuristic Search last revised January 31, 2012

Objectives:

1. To introduce the use of heuristics in searches
2. To introduce some standard heuristic algorithms
3. To introduce criteria for evaluating heuristics

Materials:

1. Projectable of Figure 4-6 from Winston 2nd ed
2. 8 puzzle demo program with file hard.8p
3. Maze program demo with lost_freshman
4. Animation of A*

I. Introduction

A. Thus far, we have considered two basic search techniques (DFS and BFS) and some variants that are essentially "blind": they explore alternatives in some fixed order without regard to their likelihood of success. We call such search techniques UNINFORMED.

B. Though such search techniques will always ultimately find the solution to a problem if one exists, they can be terribly inefficient for even moderate size problems. In most cases, we need to make use of an INFORMED search - one that relies on heuristic knowledge about the specific problem to focus the search on the "right" region of the search space.

1. A heuristic is a "rule of thumb" that gives some sense of what alternative among those available is most likely to lead to success.

   Example: in trying to get to an unfamiliar destination, we often use the heuristic "drive in the general direction of the destination"

2. An important characteristic of a heuristic is that it is not guaranteed to always identify the best alternative; rather, a heuristic gives a good answer most of the time.

C. The key idea behind a heuristic technique is to make use of knowledge about the problem to estimate how close a given state is to the final solution, and then prefer steps that minimize this

1. For example, for the 8 puzzle there are quite a few heuristics one might use - some better than others.

   a) One heuristic measure of a state is the count of the number of tiles that are out of place. As we saw last time, using this heuristic makes it possible to solve puzzles that could not be solved (in practical time) using an uninformed search like BFS.

   b) A better one is the sum of distances out of place for the tiles.

      Example:
      2 1 6
      4 8
      7 5 3

      Count is 7 - the only tile in the correct place is 7
      Sum of distances is 1 + 1 + 3 + 2 + 2 + 0 + 1 + 2 = 12

      (This is the one we will use for our examples)

2. Note the distinction between the goodness of a SOLUTION and the goodness of a STATE.

   a) The goodness of a solution is related to the cost of actually carrying it out.

   b) The goodness of a state is related to the ESTIMATED cost of a solution from that state. (E.g., for the distance heuristic, any solution from a given state must each tile a number of times

equal to the distance it is out of place - but usually quite a few more!)

c) We will shortly see an example where preferring to go through a state that looks good (based on its heuristic estimate) may actually lead us to a much less than optimal solution.

D. We will consider three subtopics in this series of lectures.

1. Two heuristic search techniques for finding ANY solution to a given problem

2. A heuristic technique that minimizes both search effort and execution effort.

3. Criteria for measuring the "goodness" of a heuristic.

II. Some Heuristic Algorithms for Finding Any Solution

A. Hill-climbing: an informed variant of DFS

1. In pure DFS, when expanding a node the selection of a new current node from among the children of the new current node is arbitrary. We call such a selection UNINFORMED.

2. However, if it is possible to derive some sort of measure as to how close a given node appears to be to the goal, then it makes sense, when expanding a node to select the successor node that is (apparently) closest to the goal. We call this an INFORMED choice.

   Example: The 8 puzzle, using sum of distances out of place as a heuristic measure, starting from the following state

   1 2 3 Sum of distances = 0 + 0 + 0 + 0 + 1 + 1 + 0 + 0 = 2
   8 4 5
   7 6

The states resulting from expanding this node (assuming we try moves in the order move blank left, up, right, down) and their heuristic values, are:

left:

1 2 3 Sum of distances = 0 + 0 + 0 + 0 + 1 + 1 + 0 + 1 = 3
8 4 5
7 6

up:

1 2 3 Sum of distances = 0 + 0 + 0 + 0 + 1 + 0 + 0 + 0 = 1
8 4
7 6 5

right - not possible

down - not possible

Given the order of expansion used, pure dfs (with no heuristic) would try the "left" move first - which would clearly be wrong!

On the other hand, dfs with hill climbing would prefer "up", and would complete the solution on the next move.

3. This method is called hill-climbing because it is analogous to the way one might set out to climb a mountain in the absence of a map: always move in the direction of increasing altitude.

   a) This is especially useful if we don't know ahead of time what the final outcome will be, and want to find "the highest ground".

   b) Like real-world hill climbing, however, it suffers from the problems of false peaks: one can reach a non-goal node from which there is no way to go but down.

Example: Hill climbing does not work well with a classic "toy" problem" called the missionaries and cannibals problem, if we use the obvious heuristic.

(1) Statement of the problem: Three missionaries and three cannibals wish to cross a river, using a boat that can only carry two people at a time. If the number of cannibals on either shore ever exceeds the number of missionaries on that shore, the cannibals will eat the missionary/ies there, so the solution must avoid this.

(a) A state can be represented by three numbers: # of missionaries on the far shore, # of cannibals on the far shore; 1 if boat is on the far shore, 0 if it is not

i) Initial state: { 0, 0, 0 }
ii) Goal state: { 3, 3, 1 }

(b) Operations:

i) Two missionaries row from near to far shore (2MNF)
ii) One missionary and one cannibal row from near to far shore (1M1CNF)
iii) Two cannibals row from near to far shore (2CNF)
iv) One missionary rows from near to far shore (1MNF)
v) One cannibal rows from near to far shore (1CNF)
vi) Two missionaries row from far to near shore (2MFN)
vii) One missionary and one cannibal row from far to near shore (1M1CFN)
viii) Two cannibals row from far to near shore (2CFN)
ix) One missionary rows from far to near shore (1M1CFN)
x) One cannibal rows from far to near shore (2CFN)

(Of course, only a subset are applicable at any given time, both because of the placement of the people and the boat and because of the “avoid getting eaten” rule)

(c) An obvious heuristic measure of the goodness of a state is “number of people on the starting bank” - initially 6, goal 0.

(2) Clearly there are three possible initial moves that don't result in anyone being eaten: one cannibal rows across, or two cannibals row across, or one cannibal and one missionary row across.

The hill-climbing strategy rightly prefers either of the latter moves to the first possibility.

However, the hill climbing approach would reach a false peak after this first move. Someone has to bring the boat back to the starting point, so all possible moves increase the number of people on the starting bank by at least one. If we were not confident that there in fact existed a total solution to the problem, we might content ourselves with a hill climbing approach that stops whenever it can find no move that improves the situation.

(a) In such a case, we would stop after the first move. A program that did this naively might report "There is no way to get all six people across the river, but I did the best I could and got two across"

(b) Though this is laughable here, it is a real problem when looking to maximize some measure of success whose actual maximum possible value is not known ahead of time.)

(3) We also run into another problem later in the solution, because there comes a time when we must have TWO people row back from to the starting shore, which looks like a worse move than having just one person row that way. This happens when we have two missionaries and two cannibals on the far shore. If a missionary rows back alone,

the two cannibals will eat the other missionary; and if a cannibal rows back alone, there will be two cannibals and just one missionary on the starting shore, so he will be eaten!

c) Hill-climbing suffers from some other potential problems that have analogies to physical hill climbing.

(1) Flat terrain: if there are large regions where movement in any direction results in no visible change in the situation (the hills are very steep but small) then hill-climbing may wander aimlessly without "spotting" a hill.

Illustration: the Colorado Rockies are located in western Colorado, but the eastern part of the state is flat as a pancake. Movement in the direction of increasing height would never get you to the Rockies.

(2) Ridges: if the number of possible moves is restricted, it may be that any one move produces no improvement, but two moves in the right combination would.

Example: See Winston figure 4-6 page 94 for examples of false peaks, flat terrain, and the ridge problem. (PROJECT)

d) Finally, because hill climbing only sorts the children of the node being expanded, it can easily become committed to a bad early choice that leads to a bad solution.

EXAMPLE: Demo DFS-Hill on hard.8p (for which there is an 18 move solution - using “Distance” heuristic.

DFS-Hill finds a solution relatively quickly, but it's quite bad!

(1) Show solution - note that the first move is “move the blank left”.

(2) It turns out that in the 18 move solution the first move should be “move the blank right”.

(3) DFS-Hill, having made a wrong initial move, eventually finds a poor solution because it has no way of backing out of a poor initial choice

B. Best-first-search: an informed variant of BFS

In best-first search, we seek to address this weakness of hill-climbing by reorganizing the ENTIRE LIST after every expansion, and choosing to expand the best node of all the ones that are open.

1. This means that bad estimated cost nodes are unlikely to be expanded, except as a last resort; but we do not ever totally reject a possible solution - even if it doesn't look good - the way the other two techniques do.

2. Of course, the complete reorganization of the list after each expansion is more computationally-expensive than the other methods.

   DEMO on hard.8p hard problem, using distance heuristic.

   Note that the solution that is found now takes more work to find, but is much better, though still far from optimal).

III. A Heuristic Algorithm for Finding an Optimal Solution

A. In the version of Branch and Bound we developed in our discussion of uninformed methods, the only criterion for solution selection was actual accumulated cost so far. We could improve the search if we added an estimate of the remaining cost to obtain an estimated total cost (cost so far + estimate), using that as the basis for ordering.

B. A * is a standard search algorithm based on this idea. At each step we order the nodes based on total cost, not just cost so far. We keep going until a goal is at the front of the open list.

   Demo: Project Maze with lost-freshman. Assume we use the following estimates of remaining cost (essentially an "as the crow flies" heuristic) - considering, for simplicity, only the nodes that actually show up in our search

   Barrington 80
   Roosevelt 90
   Frost 240
   MacDonald 155
   Emery 110
   Jenks 40
   KOSC 220
   Chase 210

   Run through animation

C. Naturally, our estimate of remaining cost will not be exactly correct. To preserve the guarantee of an optimal solution, it is important that we use an under-estimate of the remaining cost. (Where by under-estimate we mean one that is certain to be <= the true cost.)

   1. This is necessary and sufficient to guarantee we will still find the optimal solution.

   2. To see why this must hold, consider our maze problem again:

      PROJECT maze with lost-freshman animation - second slide (just after expanding MacDonald)

      Suppose our estimate of remaining distance for Emery were 400 (which is obviously way too big). Then at some point, after expanding KOSC, Frost, and Phillips) we would have the following (among other) on the open list:

Chase (estimated cost (210+210) 420)
Emery (estimated cost (50+400) 450)

At this point, we would expand Chase and find a suboptimal path to Drew

3. To show that it is sufficient, suppose that A* finds a suboptimal solution. This would mean that the a goal node representing a suboptimal solution got to the front of the open list, which in turn would mean that its total cost is <= the estimated total cost of any other state on the open list. But since we are using an underestimate, the estimated cost of any state on the open list must be <= the true cost of any solution passing through that state, which violates our assumption that the solution we found is suboptimal (assuming, of course, as would reasonably be the case, that all costs are positive)

4. For a given problem, a heuristic that always yields an estimate of remaining cost that is <= the true cost is called an admissable heuristic. For our example problem, the "as the crow flies" distance heuristic is admissible, since it is always <= true path length.

IV. **Criteria for choosing a good heuristic**

A. Clearly, one key to successful use of many of the search algorithms we have discussed is finding a good, reliable heuristic for estimating remaining cost. In fact, apart from the use of a heuristic or another strategy we shall consider shortly (constraint propagation), search is seldom a practical way to find a solution to a problem.

B. As we have noted, if we are searching for the best solution to a problem, we want to be sure our heuristic is admissible - that is, that it is an underestimate. Otherwise, we lose the guarantee of finding the best plan. (However, this is not an issue if we simply want to find a solution .)

C. However, the less the heuristic underestimates the better.

1. Example: a heuristic that is guaranteed to be an underestimate is to always use an estimate of 0. This, of course, degenerates to a totally uninformed search.

2. The ideal heuristic would be one that gives the exact value of remaining cost. This would be a totally informed search - but is seldom possible.

3. Often, we have several candidate heuristics which are neither totally uninformed nor totally uninformed. In this case, we want to choose the most informed of the candidates.

   We say that a heuristic h1 is more informed than a heuristic h2 if h1 >= h2 for all nodes.

   Example: For the 8 puzzle, we one heuristic is the raw count of the # of tiles that are out of position. This is clearly admissible, since each of these tiles must be moved at least once.

   We used a somewhat better heuristic: the sum of the distances tiles are out of place (the Manhattan distance). This is still admissible, and is more informed since it is always >= our raw count heuristic.

   DEMO: hard 8 puzzle using A* with Count and Distance heuristics

   a) Both find an 18 move solution (since both are admissible).

   b) But using Count involves expanding a lot more states

   c) Actually, we can do even better using some other possibilities:

      - Try each - note that some are much better, but others are worse (including one worse than Count)

      (We will explore these on homework.)

D. Though only an admissible heuristic guarantees that our search will find an optimal solution, there are times when a non-admissible heuristic (i.e. one that sometimes overestimates) may yield close to optimal results while also helping to minimize search effort. The last heuristic we demonstrated (Seq x 3 + distance) is such a heuristic

1. The sequence score (which is listed by itself as well) is arrived at by going around the outside, counting a 2 for every tile not followed by its proper successor and a 0 otherwise. (Note: if a tile is followed by a blank and then its proper successor - e.g. if the top row is 1 _ 2 - then count this as a 0; otherwise count it as a 2. (In effect, we skip over the blank and look at the very next tile.) Finally, if there is a piece in the center, count it as 1.

   Example: The Sequence score for

   2 8 3
   1 6 4
   7 5

   is 2 followed by 8 - count as 2;
   8 followed by 3 - count as 2;
   3 followed by 4 - count as 0:
   4 followed by 5 - count as 0
   5 followed by 7 - count as 2
   7 followed by 1 - count as 2
   1 followed by 2 - count as 0
   tile in middle - count as 1:

   total = 9

2. The sum of distances out of place is something we are already familiar with. For example, for the puzzle we just looked at, the sum of distances is 1 + 2 + 0 + 1 + 1 + 0 + 0 + 0 = 5

3. So the total using sequence x 3 + distance is 9 x 3 + 5 = 32

4. However, this heuristic is not admissible. For example, the following puzzle has a 1 move solution (move 8 left one square):

```
1 2 3
  8 4
7 6 5
```

However, for this configuration the sequence score is 1 and the distance score is 1, yielding an estimate of 4, which is clearly an overestimate.

5.Nonetheless, the heuristic can produce good results, as we showed when using it with a hard puzzle.

V. **Conclusion**

A. The use of a good heuristic frequently makes the difference between a problem that is not practically-solvable and one that is. In practice, then, uninformed algorithms like BFS DFS and its variants, and Branch and Bound are not often useful by themselves. (In fact, we discussed them largely to set the stage for heuristic algorithms).

B. There are two heuristic algorithms that tend to be especially useful.

1. Hill-climbing

2. A* (It can be used with problems that don't have a separate cost measure by using number of moves as a measure of cost)

   Can you think of examples where you have used a process akin to one or the other of these in your own life (outside of this course)?

   ASK

C. However, it appears that the intelligence really resides more in identifying the heuristic - rather than in the actual search process that uses it!