

## Materials:

1. Search Algorithms handout from last time
2. Projectable of Figure 4-6 from Winston 2nd ed
3. 8 puzzle demo program with files 2moves, Nilsson
4. Maze demo program with files lost\_freshman, lost\_freshman\_with\_140\_lewis

## I. Introduction

- -----

- A. Thus far, we have considered two basic search techniques (DFS and BFS) 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. Today, we consider issues related to search "quality".
  1. Search techniques that are applicable when we have some (possibly imperfect) way of measuring how close a given state is to the goal. The methods we will consider all seek to minimize SEARCH EFFORT by preferring to expand nodes that appear closer to the goal, rather than those that appear further away.
  2. Search techniques designed to minimize EXECUTION EFFORT for actually implementing the plan discovered by the search, by finding a solution whose cost is minimal by some measure.
  3. Note that these are two very different goals - indeed, in some cases they may be antithetical. (E.g. in some cases DFS may FIND a solution much more quickly than BFS; but BFS is guaranteed to find the solution requiring the fewest moves (which may or may not be the best if different moves have different costs.))
  4. The final search algorithm we will consider is one that seeks to minimize BOTH measures of effort (search effort and execution effort.)
- C. We will consider three subtopics in this series of lectures.
  1. Several heuristic search techniques for finding ANY solution to a given problem.
  2. Techniques for finding the BEST solution to a given problem.
    - a. A blind technique
    - b. A heuristic technique
  3. Criteria for measuring the "goodness" of a heuristic.
- D. The distinction between search techniques is related to the distinction between weak and strong methods of problem solving.
  1. A weak method uses blind search, or a heuristic that is broadly applicable to many kinds of problems - e.g. means-ends-analysis as used with GPS.
  2. A strong method uses a heuristic that incorporates significant knowledge about the specific problem - e.g. the examples we will look at in conjunction with using heuristic search with the 8 puzzle shortly.

## II. Heuristic Search Techniques for Finding ANY Solution to a Problem

-- -----

- A. All of these techniques depend on the existence of some kind of HEURISTIC MEASURE of the closeness of a given state to the goal. We call such searches INFORMED SEARCHES. Such measures are often reasonably easy to find for a given problem, though some care needs to be taken to ensure that the measurement chosen is a good one. (We consider criteria for "goodness" later.)

Example: For many of our examples, we will use the 8 puzzle, as in our earlier discussion of blind search.

1. The state space has  $9! = 362,880$  states (partitioned into two disjoint subsets.)
2. For non-trivial puzzles, a blind search technique can easily get caught up in trying so many possibilities that it becomes intractable.

DEMO: Nilsson's hard problem with DFS, BFS

For such problems, we will probably need to use a search method that prefers "good" states over "bad" states at each step along the way.

4. A possible measure of "goodness" of a state is the number of tiles that are out of their proper place - where a score of 8 is the worst case and 0 is the goal. (The initial state for the problem we just looked at scores 7 - only the 7 tile starts out in its proper place.) (Note that, as is often the case, a higher value of the heuristic means a worse state since we are using estimated COST of a solution.)
5. As we shall see, though, a better measure is helpful for really hard puzzles like this one. We will look at some more sophisticated measurements later.
6. 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. any solution from a given state must move at least as many tiles as the number that are 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 may actually lead us to a much less than optimal solution.

### B. 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, as just noted, often it is possible to derive some sort of measure as to how close a given node appears to be to the goal. In this case, when expanding a node it makes sense to select the successor node that is (apparently) closest to the goal. We call this an INFORMED choice.

Ex: The 8 puzzle, using tiles out of place as a heuristic measure, starting from the following state

```

      1 2 3           Tiles out of place = 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           Tiles out of place = 3
       8 4 5
       7  6
  
```

```

up     1 2 3           Tiles out of place = 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.

HANDOUT: Discuss algorithm

DEMO: The above puzzle with file 2moves and both search methods

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.

- i. 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.

The obvious heuristic measure of the goodness of a state is "number of people on the starting bank" - initially 6, goal 0.

- ii. 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.

- iii. 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.

- 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"

(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.)

- iv. 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.

One of the two optimal solutions to this problem looks like this - the other is similar. States are shown as (# cannibals on start shore, # missionaries on start shore, 1 if boat is at start, else 0). Moves are shown as #cannibals + #missionaries in boat - F = forward, B = back.

```
(3 3 1) -> (2 2 0) -> (2 3 1) -> (0 3 0) -> (1 3 1) -> (1 1 0) ->
      1+1F      0+1B      2+0F      1+0B      0+2F      1+1B
(2 2 1) -> (2 0 0) -> (3 0 1) -> (1 0 0) -> (2 0 1) -> (0 0 0)
      0+2F      1+0B      2+0F      1+0B      2+0F
```

Note the move that requires 1 of each to row the boat back - if this is not done, a single missionary ends up on a shore with two cannibals!

- 4. One approach that is sometimes used to handle false peaks - especially in a situation where we are not sure whether our goal is attainable - is called SIMULATED ANNEALING.
  - a. This technique derives its name from an analogy to what happens when metals are hardened by annealing. The metal is heated to a high temperature and gradually cooled. As a result, the atoms settle into a very uniform crystalline structure with no defects.

- b. In ordinary hill-climbing, we always choose the move that gives the best improvement in heuristic score. If we are unsure of what the best possible outcome is (e.g. in the missionaries and cannibals problem stopping with two people on the far bank may be the best we can do), then we never choose a move that makes the heuristic score worse.
  - c. To use simulated annealing in such a case, we create a "temperature" variable ( $T$ ) that is initialized to a fairly large value and then gradually decreased toward  $0$  as the search proceeds. We associate with each "bad" move a probability value  $p' = \exp(-\Delta E/T)$ , where  $\Delta E$  is the increase in heuristic cost score that would result from choosing the bad move. Based on the value of  $p'$  (and using a random number generator), we occasionally choose a bad move instead of a good one.
  - d. Note that:
    - i. Making a slightly "bad" move is more probable than making a really bad one. (Since  $p'$  decreases as  $\Delta E$  increases.)
    - ii. As temperature decreases, the probability of making any "bad" move drops toward  $0$ .
5. Hill-climbing suffers from some other potential problems that have analogies to physical hill climbing.
- a. 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.
  - b. 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.
 

Ex: See Winston figure 4-6 page 94 for examples of false peaks, flat terrain, and the ridge problem. (TRANPARENCY)
6. 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 Nilsson's hard problem (for which there is an 18 move solution - using sum of distances out of place heuristic.
- DFS-Hill finds a solution relatively quickly, but it's quite bad!

### C. Best-first-search: an informed variant of BFS

1. In best-first search, we seek to address this weakness of hill-climbing by reorganizing the ENTIRE LIST after every expansion by inserting the newly generated nodes into it by heuristic value. Thus, the first node in the list is always the one having the best heuristic value and the last node has the worst heuristic value. (Ties between states having equal heuristic values are broken FIFO).
  - a. 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.
  - b. Of course, the complete reorganization of the list after each expansion is more cumbersome than the other methods. If the open list is maintained as a linked list, this can be made efficient by first sorting the new nodes (a short list), and then merging the two lists.
2. HANDOUT - discuss algorithm
3. DEMO on Nilsson's hard problem, using count heuristic. (The solution that is found now takes more work to find, but is much better, though still far from optimal).

### III. Search Methods for Finding the Best Solution

--- -----

- A. So far, we have had as our goal finding ANY solution to the problem at hand, without regard to trying to find an optimal solution.
  1. Many times, search is used to develop a plan of action that will later be carried out to actually solve the problem. (E.g. if the missionaries and cannibals problem were a real problem then real people would actually have to row the boat back and forth across the river.)
  2. There is a tradeoff here between search cost and plan execution cost.
    - a. For example, if we are planning a cross-country trip, we will probably find it worthwhile to engage in some "search" by having the travel agent check various special fares etc., rather than just taking the first flight we find that goes where we want. That is, we trade off search effort now for reduced cost later.
    - b. On the other hand, if we are taking a cab to a destination two miles from home any cab that is available will probably do.
  3. When looking for the BEST solution to the problem, we have to consider two possible ways of measuring "best":
    - a. Smallest number of individual steps. This is the measure we would use for the 8 puzzle.
    - b. If different steps have different costs, then the smallest total cost, even if it involves more steps. (For example, one can often

save considerable money on air fare by taking two connecting flights instead of a single non-stop.)

4. The methods we have considered so far do not find optimal solutions (by either criterion) in general - though BFS will find the solution with the smallest number of steps. They are therefore primarily useful if the total expected effort in carrying out the search is comparable to the total expected effort of using the information once the search succeeds. However, if more effort is expected from using the information, or if we plan to use the information many times, a search such as we are about to describe may be better.
- B. One approach to finding optimal solutions is the (tongue-in-cheek named) "British Museum" procedure: generate all the solutions, then pick the best. Clearly, this is absurd for large problems.
- C. For this section of the lecture, we will be using a hypothetical example of a freshman trying to find his way from MacDonald to Drew. The following is a campus map (not 100% accurate) that the freshman is using. (Distances are in Smoots).

#### 1. PROJECT EXAMPLE

2. Observe: the shortest path, in terms of shortest total distance, is  
MacDonald Emery Jenks Drew (220 smoots)

there is a path with fewer moves (MacDonald Lewis Drew), but it's longer (300 smoots)

3. Demonstrate searches discussed so far:

DFS (note: tries south first)

#### BFS

DFS-Hill (approximation to an as the crow flies heuristic) - note that an initial move to Physical Plant brings the freshman closest to the goal, but does not lead to the shortest path. Thereafter, the fact that we choose the best among the children of the node currently being expanded leads to a fairly bad ultimate solution.)

Best-First (same heuristic) - still does not find optimal solution, but does better in this case. (Roosevelt is closer than Emery, even though the ultimate path through Emery ends up being better.)

- D. A non-heuristic method for finding optimal solutions is called branch and bound.

1. Its basic principle is this: for each open node, keep a record of the accumulated cost so far of getting to it from the start node and organize the nodes on the list on this basis. This means that, when a goal node gets to the front of the open list, it will have the smallest accumulated cost of any node on the open list, and hence of any goal that can be reached (assuming that all costs are non-negative).

2. HANDOUT - Discuss algorithm

### 3. DEMONSTRATE

4. Unfortunately, without addressing one additional issue, pure branch and bound can sometimes yield a wrong result.

DEMONSTRATE: Modified maze with distance from MacDonald to Lewis at 140, rather than 160.

Run using Branch and Bound. Why does this give a suboptimal solution?

ASK

The path through Lewis to Drew is discovered before the path through Jenks - even though the latter is better - because we order based on cost so far. (Cost to Lewis = 140; to Jenks = 150). We still expand Jenks, but we discard Drew because it duplicates a node already on the open list.

The additional refinement that is needed is this: when expanding a node, do not discard a child that is already on the open or closed list if the path we have just found to it is shorter. This applies not only to goal nodes, but also to intermediate nodes. We call this refinement DYNAMIC PROGRAMMING.

5. Note that branch and bound does NOT require any heuristic estimate of plan cost. Thus, it can require considerable search effort on problems where the path to the solution involves many steps, or can even fail to find a solution in reasonable time.

DEMO - Branch and bound on Nilsson 8-puzzle

6. The final method we will consider is basically a heuristic-based improvement to branch and bound, with some additional "twists".
- E. Branch and Bound (with dynamic programming) + underestimate of remaining length + one more refinement yields A\*.
1. In the version of Branch and Bound we just developed, the only criterion for solution selection was actual accumulated cost so far. Clearly, 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.
  2. 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  $\leq$  the true cost.)
    - a. This is necessary and sufficient to guarantee we will still find the optimal solution.

- b. To show that it is necessary, consider the maze problem we have been using for our examples. Suppose the estimate for Emery was too high - e.g. suppose it was 260. Then the path through Lewis would be discovered first, and would be taken as the optimal solution, since its actual total (300) is less than the estimated total through Emery ( $50 + 260 = 310$ ).
- c. 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  $\leq$  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  $\leq$  the true cost of any solution passing through that state, which violates our assumption that the solution we found is suboptimal.
- d. For a given problem, a heuristic that always yields an estimate of remaining cost that is  $\leq$  the true cost is called an admissible heuristic.

For our example problem, the "as the crow flies" distance heuristic is admissible, since it is always  $\leq$  true path length.

- 4. The algorithm we have developed above is a widely used AI search algorithm called A\*. To summarize, A\* involves:
  - a. Use total estimated cost (actual cost so far + estimate of remaining cost) as the basis for ordering open nodes.
  - b. Use dynamic programming: when a state is discovered that duplicates one already on the open or closed list, do not discard it if the new path is shorter than the one seen previously.

HANDOUT - discuss algorithm

(Note: Luger gives a slightly different version of Best-First search which is equivalent to A\* if used with an admissible heuristic)

- 5. Though both Branch and Bound (with dynamic programming) and A\* (with an admissible heuristic) can guarantee optimal solutions, A\* generally involves less search effort.
  - a. Re-run original maze with Branch and Bound and A\* - note that both find the optimal path, but A\* considers fewer nodes.
  - b. Recall that Branch and Bound failed to find a solution (in reasonable time) to the hard 8 puzzle.

DEMO A\* on this puzzle, using count heuristic

- F. IDA\* is a variant of A\* that uses a form of iterative deepening. Here, we prune the search tree by discarding nodes whose estimated cost is greater than some cutoff; and then gradually increase the cutoff until we succeed.

#### 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.
- 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.
- C. Another characteristic is MONOTONICITY. In brief, what this means is that for any pair of states on the same path to a solution, the estimated distance between them (heuristic estimate for earlier - estimate for later) is less than or equal to the true distance between them.
  1. Of course, monotonicity implies admissibility - but the reverse is not true.
  2. The benefit of using a monotonic heuristic is that the complicated checking of successors against the open list in A\* can be changed to simply  
discard any successor nodes already on open or closed lists because states are discovered in order of increasing cost, so the node already on the open or closed list will necessarily be at least as good as the successor being considered.
- D. 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 informed. In this case, we want to choose the most informed of the candidates.
  4. We say that a heuristic  $h_1$  is more informed than a heuristic  $h_2$  if  $h_1 \geq h_2$  for all nodes.

Ex: For the 8 puzzle, we used as a heuristic a 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.

A better heuristic would be to determine, for each tile, the distance between where it is and where it should be (the Manhattan distance). This is still admissible, and is more informed since it is always  $\geq$  our raw count heuristic.
  5. EXAMPLE: We can identify several possible heuristics for the 8 puzzle, with varying degrees of informedness. Using a better heuristic with an algorithm like A\* will may get an answer quicker.

- a. Re-demonstrate A\* with Count heuristic.
  - b. Demonstrate A\* sum of distances out of place heuristic. (Still admissible).
  - c. Several other heuristics are implemented, which you will explore on homework.
- E. 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.
1. For example, in an earlier book, Nilsson notes that for the eight puzzle the following heuristic has been shown to give good fast convergence on a solution for difficult problems: (Nilsson (1980) page 85)

$$P(n) + 3 S(n)$$

where, for a given state n:

P is the sum of the distances of each tile from home.  
 S is a sequence score arrived 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.

2. 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 P(n) is 1 and S(n) is 1, yielding an estimate of 4.

3. Nonetheless, the heuristic can produce good results.

DEMONSTRATE with hard puzzle.