

For Wednesday

- Read 6.1-6.3
- Homework:
 - Chapter 4, ex. 1 (may be done in groups)

Program 1

- Any questions?

Uniform Cost Search

- Similar to breadth first, but takes path cost into account

Depth First Search

- How does depth first search operate?
- How would we implement it?
- Performance:
 - Completeness
 - Optimality
 - Space Complexity
 - Time Complexity

Comparing DFS and BFS

- When might we prefer DFS?
- When might we prefer BFS?

Improving on DFS

- Depth-limited Search
- Iterative Deepening
 - Wasted work???
 - What kinds of problems lend themselves to iterative deepening?

Bi-directional Search

- What advantages are there to bi-directional search?
- What do we have to have to use bi-directional search?

Repeated States

- Problem?
- How can we avoid them?
 - Do not follow loop to parent state (or me)
 - Do not create path with cycles (check all the way to root)
 - Do not generate any state that has already been generated. -- How feasible is this??

Informed Search

- So far we've looked at search methods that require no knowledge of the problem
- However, these can be very inefficient
- Now we're going to look at searching methods that take advantage of the knowledge we have a problem to reach a solution more efficiently

Best First Search

- At each step, expand the most promising node
- Requires some estimate of what is the “most promising node”
- We need some kind of **evaluation function**
- Order the nodes based on the evaluation function

Greedy Search

- A **heuristic function**, $h(n)$, provides an estimate of the distance of the current state to the closest goal state.
- The function must be 0 for all goal states
- Example:
 - Straight line distance to goal location from current location for route finding problem

Heuristics Don't Solve It All

- NP-complete problems still have a worst-case exponential time complexity
- Good heuristic function can:
 - Find a solution for an average problem efficiently
 - Find a reasonably good (but not optimal) solution efficiently

Beam Search

- Variation on greedy search
- Limit the queue to the best n nodes (n is the **beam width**)
- Expand all of those nodes
- Select the best n of the remaining nodes
- And so on
- May not produce a solution

Focus on Total Path Cost

- Uniform cost search uses $g(n)$ --the path cost so far
- Greedy search uses $h(n)$ --the estimated path cost to the goal
- What we'd like to use instead is
$$f(n) = g(n) + h(n)$$
to estimate the **total** path cost

Admissible Heuristic

- An **admissible heuristic** is one that never overestimates the cost to reach the goal.
- It is always less than or equal to the actual cost.
- If we have such a heuristic, we can prove that best first search using $f(n)$ is both complete and optimal.
- **A* Search**

8-Puzzle Heuristic Functions

- Number of tiles out of place
- Manhattan Distance
- Which is better?
- Experiment
- Effective branching factor

Inventing Heuristics

- Relax the problem
- Cost of solving a subproblem
- Learn weights for features of the problem

Local Search

- Works from the “current state”
- No focus on path
- Also useful for optimization problems

Local Search

- Advantages?
- Disadvantages?

Hill-Climbing

- Also called **gradient descent**
- Greedy local search
- Move from current state to a state with a better overall value
- Issues:
 - Local maxima
 - Ridges
 - Plateaux

Variations on Hill Climbing

- Stochastic hill climbing
- First-choice hill climbing
- Random-restart hill climbing

Evaluation of Hill Climbing

Simulated Annealing

- Similar to hill climbing, but--
 - We select a random successor
 - If that successor improves things, we take it
 - If not, we may take it, based on a probability
 - Probability gradually goes down

Local Beam Search

- Variant of hill-climbing where multiple states and successors are maintained

Genetic Algorithms

- Have a **population** of k states (or **individuals**)
- Have a **fitness function** that evaluates the states
- Create new individuals by randomly selecting pairs and mating them using a randomly selected **crossover point**.
- More fit individuals are selected with higher probability.
- Apply random **mutation**.
- Keep top k individuals for next generation.

Other Issues

- What issues arise from continuous spaces?
- What issues do online search and unknown environments create?