Local Search
AI Class 6 (Ch. 4.1-4.2)

Admissibility

- Admissibility is a property of heuristics
  - They are optimistic -- think goal is closer than it is
    - (Or, exactly right)
  - Admissible algorithms can be pretty bad!
- Is \( h(n) \): “1 kilometer” admissible?
- Using admissible heuristics guarantees that the first solution found will be optimal, for some algorithms (A*).

Admissibility and Optimality

- Intuitively:
  - When A* finds a path of length \( k \), it has already tried every other path which can have length \( \leq k \)
  - Because all frontier nodes have been sorted in ascending order of \( f(n) = g(n) + h(n) \)
- Does an admissible heuristic guarantee optimality for greedy search?
  - Reminder: \( f(n) = h(n) \), always choose node “nearest” goal
  - No sorting beyond that

Local Search Algorithms

- Sometimes the path to the goal is irrelevant
  - Goal state itself is the solution
    - 3 an objective function to evaluate states
  - In such cases, we can use local search algorithms
  - Keep a single “current” state, try to improve it

Today’s Class

- Iterative improvement methods
  - Hill climbing
  - Simulated annealing
  - Local beam search
  - Genetic algorithms
  - Online search

“If the path to the goal does not matter... we can use a single current node and move to neighbors of that node.”

~ R&N pg. 121

Local Search Algorithms

- Sometimes the path to the goal is irrelevant
  - Goal state itself is the solution
    - 3 an objective function to evaluate states
  - In such cases, we can use local search algorithms
  - Keep a single “current” state, try to improve it

Very efficient!

Why?
Iterative Improvement Search

- Start with an initial guess
- Gradually improve it until it is legal or optimal
- Some examples:
  - Hill climbing
  - Simulated annealing
  - Constraint satisfaction

Hill Climbing on State Surface

- Concept: trying to reach the “highest” (most desirable) point (state)
- “Height” Defined by Evaluation Function
- Look one step ahead to determine if any successor is “better” than current state
- If so, move to the best successor
- A kind of Greedy search in that it uses $h$
  - But, does not allow backtracking or jumping to an alternative path
  - Doesn’t “remember” where it has been.
- Not complete
  - Search will terminate at local minima, plateaux, ridges.

Hill Climbing Search

- If there exists a successor $s$ for the current state $n$ such that
  - $h(s) > h(n)$
  - $h(s) >= h(t)$ for all the successors $t$ of $n$
    then move from $n$ to $s$. Otherwise, halt at $n$.
- Look one step ahead to determine if any successor is “better” than current state
  - If so, move to the best successor

Hill Climbing Example

$f(n) = -(\text{number of tiles out of place})$
Exploring the Landscape

- **Local Maxima:** Peaks that aren't the highest point in the space
- **Plateaus:** A broad flat region that gives the search algorithm no direction (random walk)
- **Ridges:** Flat like a plateau, but with drop-offs to the sides; steps to the North, East, South, and West may go down, but a step to the NW may go up.

Example of a Local Optimum

- **Start:** $f = -6$
- **Goal:** $f = 0$

Some Extensions of Hill Climbing

- **Simulated Annealing:**
  - Escape local maxima by allowing some "bad" moves but gradually decreasing their frequency
- **Local Beam Search:**
  - Keep track of $k$ states rather than just one
  - At each iteration:
    - All successors of the $k$ states are generated and evaluated
    - Best $k$ are chosen for the next iteration
- **Stochastic Beam Search:**
  - Chooses semi-randomly from "uphill" possibilities
  - "Steeper" moves have a higher probability of being chosen
- **Random-Restart Climbing:**
  - Can actually be applied to any form of search
  - Pick random starting points until one leads to a solution
- **Genetic Algorithms:**
  - Each successor is generated from two predecessor (parent) states

Drawbacks of Hill Climbing

- **Problems:** local maxima, plateaus, ridges
- **Remedies:**
  - **Random restart:** keep restarting the search from random locations until a goal is found.
  - **Problem reformulation:** reformulate the search space to eliminate these problematic features
- **Some problem spaces are great for hill climbing; others are terrible**

Gradient Ascent / Descent

- **Take downward "steps" proportional to the negative of the gradient of the function at current state.**
- **"Steepest descent"**
- **Gradient descent procedure for finding the arg\text{min} f(x):**
  - choose initial $x_0$ randomly
  - repeat:
    - $x_{i+1} = x_i - \eta / \nabla f(x_i)$
    - until the sequence $x_0, x_1, \ldots, x_i, x_{i+1}$ converges
  - Step size $\eta$ (eta) is small (~0.1~0.05)
- **Good for differentiable, continuous spaces**
Gradient Ascent / Descent

Gradient Methods vs. Newton's Method

• A reminder of Newton's method from Calculus:
  \[ x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)} \]
  
  • Newton's method uses 2\textsuperscript{nd} order information (the second derivative, or, curvature) to take a more direct route to the minimum.
  
  • The second-order information is more expensive to compute, but converges more quickly.

Simulated Annealing

• Simulated annealing (SA): analogy between the way metal cools into a minimum-energy crystalline structure and the search for a minimum generally
  
  • In very hot metal, molecules can move fairly freely
  • But, they are slightly less likely to move out of a stable structure
  • As you slowly cool the metal, more molecules are “trapped” in place

• Conceptually: Escape local maxima by allowing some “bad” (locally counterproductive) moves but gradually decreasing their frequency

Simulated Annealing (II)

• Can avoid becoming trapped at local minima.

• Uses a random local search that:
  
  • Accepts changes that increase objective function \( f \)
  • As well as some that decrease it

• Uses a control parameter \( T \)
  
  • By analogy with the original application
  • Is known as the system “temperature”

  • \( T \) starts out high and gradually decreases toward 0

Simulated Annealing (IV)

• \( f(s) \) represents the quality of state \( s \) (high is good)

• A “bad” move from \( A \) to \( B \) is accepted with a probability

  \[ P_{\text{move } A \rightarrow B} = e^{\frac{f(B) - f(A)}{T}} \]

  • (Note that \( f(B) - f(A) \) will be negative, so bad moves always have a relative probability less than one. Good moves, for which \( f(B) - f(A) \) is positive, have a relative probability greater than one.)

• Temperature
  
  • Higher temperature = more likely to make a “bad” move
  • As \( T \) tends to zero, this probability tends to zero
  • SA becomes more like hill climbing

  • If \( T \) is lowered slowly enough, SA is complete and admissible.
    • domain-specific
    • sometimes hard to determine

Local Beam Search

• Begin with \( k \) random states
  
  • \( k \), instead of one, current state(s)

• Generate all successors of these states

• Keep the \( k \) best states

• Stochastic beam search
  
  • Probability of keeping a state is a function of its heuristic value
  • More likely to keep “better” successors
Genetic Algorithms

- The Idea:
  - New states are generated by "mutating" a single state or "reproducing" (somehow combining) two parent states
  - Selected according to their fitness
- Similar to stochastic beam search
- Start with k random states (the initial population)
  - Encoding used for the "genome" of an individual strongly affects the behavior of the search
  - Must have some combinable representation of state spaces
  - Genetic algorithms / genetic programming are a large and active area of research

Tabu Search

- Problem: Hill climbing can get stuck on local maxima
- Solution: Maintain a list of k previously visited states, and prevent the search from revisiting them
- Why not always do this?

Online Search

- Interleave computation and action (search some, act some)
  - Exploration: Can't infer outcomes of actions, must actually perform them to learn what will happen
  - Competitive ratio = Path cost found / Path cost that could be found**
    - *On average, or in an adversarial scenario (worst case)
    - **If the agent knew the nature of the space, and could use offline search
  - Relatively easy if actions are reversible
  - LRTA* (Learning Real-Time A*): Update h(s) (in state table) based on experience
  - More about online search and nondeterministic actions next time...

Summary: Informed Search (I)

- State space can be treated as a "landscape" of movement on quality of states where we are trying to find "high" points
- Best-first search is a general search where the minimum-cost nodes are expanded first.
- Greedy search uses minimal estimated cost h(n) to the goal state as measure of goodness.
  - Reduces search time, but is neither complete nor optimal.

Summary: Informed Search (II)

- Hill-climbing algorithms keep only a single state in memory, but can get stuck on local optima.
- Simulated annealing escapes local optima, and is complete and optimal given a "long enough" cooling schedule.
- Genetic algorithms search a space by modeling biological evolution.
- Online search algorithms are useful in state spaces with partial/no information.

Class Exercise:
Local Search for N-Queens

<table>
<thead>
<tr>
<th>Q</th>
<th>Q</th>
<th>Q</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>Q</td>
<td>Q</td>
<td>Q</td>
</tr>
<tr>
<td>Q</td>
<td>Q</td>
<td>Q</td>
<td>Q</td>
</tr>
</tbody>
</table>

Questions?