CSE 473: Artificial Intelligence
Autumn 2016

Local Search

Dan Weld

With slides from
Dan Klein, Stuart Russell, Andrew Moore, Luke Zettlemoyer

Goal Based Agents

- Plan ahead
- Ask “what if”

- Decisions based on (hypothesized) consequences of actions

- Must have a model of how the world evolves in response to actions

- Act on how the world WOULD BE
Types of Environments

- Fully observable vs. partially observable
- Single agent vs. multiagent
- Deterministic vs. stochastic
- Episodic vs. sequential
- Discrete vs. continuous

Search thru a Problem Space (aka State Space)

- Input:
  - Set of states
  - Operators [and costs]
  - Start state
  - Goal state [or test]

- Output:
  - Path: start $\Rightarrow$ a state satisfying goal test
    - [May require shortest path]
    - [Sometimes just need a state that passes test]
N Queens Problem
Place N queens so they don’t attack each other (same row, same col, same diagonal)

- **States**
  Chess board with 0 or more queens

- **Operators**
  Add a queen

- **Initial**
  No queens

- **Goal**
  N queens

Search thru State Space
What if Robot is Blind?

Moving into wall → noop

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
```

“Conformant Planning”

[Has a talking compass – knows which way is N]

Conformant Planning

Sterilizing surgical gear
Bowl feeder
Search thru State Space

- **States**
  - SETS of states
  - “Belief state”

- **Operators**
  - Move actions

- **Initial State**
  - Set of all states

- **Goal State**
  - Set of just goal state(s)

Soln: R, D, D, R, R, U, U

- **States**
  - SETS of states
  - “Belief state”

- **Operators**
  - Move actions

- **Initial State**
  - Set of all states

- **Goal State**
  - Set of just goal states
Move Right

- **States**
  - SETS of states
  - “Belief state”
- **Operators**
  - Move actions
- **Initial State**
  - Set of all states
- **Goal State**
  - Set of just goal states

Move Down

- **States**
  - SETS of states
  - “Belief state”
- **Operators**
  - Move actions
- **Initial State**
  - Set of all states
- **Goal State**
  - Set of just goal states
Move Down

- **States**
  - SETS of states
  - “Belief state”

- **Operators**
  - Move actions

- **Initial State**
  - Set of all states

- **Goal State**
  - Set of just goal states

Move Right

- **States**
  - SETS of states
    - “Belief state”

- **Operators**
  - Move actions

- **Initial State**
  - Set of all states

- **Goal State**
  - Set of just goal states
Move Right

- **States**
  - SETS of states
  - “Belief state”

- **Operators**
  - Move actions

- **Initial State**
  - Set of all states

- **Goal State**
  - Set of just goal states

---

Move Up

- **States**
  - SETS of states
  - “Belief state”

- **Operators**
  - Move actions

- **Initial State**
  - Set of all states

- **Goal State**
  - Set of just goal states
Move Up

- **States**
  - SETS of states
  - “Belief state”

- **Operators**
  - Move actions

- **Initial State**
  - Set of all states

- **Goal State**
  - Set of just goal states

Heuristics?

**Relaxed Problem?**
- What if it weren’t blind?
- Max # moves from any state in belief state

**Also… (admissible?)**
- Number of states in belief state
Previous Search Methods

Systematic

- **Blind Search**
  - Depth first search
  - Breadth first search
  - Iterative deepening search
  - Uniform cost search
- **Informed Search**
  - Best First
  - A*
  - Beam Search
  - Hill Climbing

Local (Randomized)
Constraint Satisfaction (Factored)

Which Algorithm?

- Uniform cost search (UCS):
Which Algorithm?

- A*, Manhattan Heuristic:

Which Algorithm?

- Best First / Greedy, Manhattan Heuristic:
Demo

http://qiao.github.io/PathFinding.js/visual/

SUGGESTED BY Fernando Centurion

Goal State vs Path

• Previously: Search to find best path to goal
  ▪ Systematic exploration of search space.

• Today: a state is solution to problem
  ▪ for some problems path is irrelevant.
  ▪ E.g., 8-queens

• Different algorithms can be used
  ▪ Systematic Search
  ▪ Local Search
  ▪ Constraint Satisfaction
Local search algorithms

- State space = set of "complete" configurations
- Find configuration satisfying constraints,
  - e.g., all n-queens on board, no attacks
- In such cases, we can use local search algorithms
- keep a single "current" state, try to improve it.
- Very memory efficient
  - duh - only remember current state

Goal Satisfaction
- Constraint satisfaction
  - reach the goal node guided by heuristic fn

Optimization
- Constraint Optimization
  - optimize(objective fn)

You can go back and forth between the two problems
Typically in the same complexity class
Local Search and Optimization

- Local search
  - Keep track of single current state
  - Move only to “neighboring” state
    Defined by operators
  - Ignore previous states, path taken

- Advantages:
  - Use very little memory
  - Can often find reasonable solutions in large or infinite (continuous) state spaces.

- “Pure optimization” problems
  - All states have an objective function
  - Goal is to find state with max (or min) objective value
  - Does not quite fit into path-cost/goal-state formulation
  - Local search can do quite well on these problems.

Trivial Algorithms

- Random Sampling
  - Generate a state randomly

- Random Walk
  - Randomly pick a neighbor of the current state

- Why even mention these?
  - Both algorithms asymptotically complete.

Hill-climbing search

- “a loop that continuously moves towards increasing value”
  - terminates when a peak is reached
  - Aka greedy local search
- Value can be either
  - Objective function value
  - Heuristic function value (minimized)
- Hill climbing does not look ahead of the immediate neighbors
- Can randomly choose among the set of best successors
  - if multiple have the best value
- “climbing Mount Everest in a thick fog with amnesia”

Example: $n$-queens

- Put $n$ queens on an $n \times n$ board with no two queens on the same row, column, or diagonal
  - Note different search space... all states have N queens
- Is it a satisfaction problem or optimization?
Hill-climbing search: 8-queens problem

- Need heuristic function
  - Convert to an optimization problem
- $h = \text{number of pairs of queens attacking each other}$
- $h = 17$ for the above state

Hill-climbing search: 8-queens

A local minimum with $h = 1$
Hill Climbing Drawbacks

- Local maxima
- Plateaus
- Diagonal ridges

Hill Climbing Properties

- Not Complete
- Worst Case Exponential Time
- Simple, $O(1)$ Space & Often Very Fast!
Hill-climbing on 8-queens

- Randomly generated 8-queens starting states...
- 14% the time it solves the problem
- 86% of the time it get stuck at a local minimum

However...
- Takes only 4 steps on average when it succeeds
- And 3 on average when it gets stuck
- (for a state space with $8^8 \approx 17$ million states)

Escaping Shoulders: Sideways Move

- If no downhill (uphill) moves, allow sideways moves in hope that algorithm can escape
  - Must limit the number of possible sideways moves to avoid infinite loops
- For 8-queens
  - Allow sideways moves with limit of 100
  - Raises percentage of problems solved from 14 to 94%

However....
- 21 steps for every successful solution
- 64 for each failure
Escaping Local Optima - Enforced Hill Climbing

- Perform breadth first search from a local optima
  - to find the next state with better h function

- Typically,
  - prolonged periods of exhaustive search
  - bridged by relatively quick periods of hill-climbing

- Middle ground b/w local and systematic search

Hill Climbing: stochastic variations

- When the state-space landscape has local minima, any search that moves only in the greedy direction cannot be complete

- Random walk, on the other hand, is asymptotically complete

Idea: Combine random walk & greedy hill-climbing
Hill-climbing with random restarts

- If at first you don’t succeed, try, try again!
- Different variations
  - For each restart: run until termination vs. run for a fixed time
  - Run a fixed number of restarts or run indefinitely
- Analysis
  - Say each search has probability \( p \) of success
    - E.g., for 8-queens, \( p = 0.14 \) with no sideways moves
- Expected number of restarts?

<table>
<thead>
<tr>
<th>Restarts</th>
<th>0</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success?</td>
<td>14%</td>
<td>36%</td>
<td>53%</td>
<td>74%</td>
<td>92%</td>
<td>99%</td>
<td>99.994%</td>
</tr>
</tbody>
</table>
- Expected number of steps taken?

Hill-climbing with random walk

- At each step do one of the two
  - Greedy: With prob \( p \) move to the neighbor with largest value
  - Random: With prob \( 1-p \) move to a random neighbor

Hill-climbing with both

- At each step do one of the three
  - Greedy: move to the neighbor with largest value
  - Random Walk: move to a random neighbor
  - Random Restart: Start over from a new, random state