Title: Local Search
Required reading: AIMA, Chapter 4
LWH: Chapters 6, 10, 13 and 14.

Introduction to Artificial Intelligence
CSCE 476-876, Spring 2015
URL: www.cse.unl.edu/~choueiry/S15-476-876

Berthe Y. Choueiry (Shu-we-ri)
(402)472-5444
Outline

Iterative improvement search:

- Hill-climbing
- Simulated annealing
- ...

B.Y. Cheung

Instructor’s notes #8
February 18, 2015
Types of Search (I)

1- Uninformed vs. informed
2- Systematic/constructive vs. iterative improvement

xxx
Iterative improvement (a.k.a. local search)

→ Sometimes, the ‘path’ to the goal is irrelevant
   only the state description (or its quality) is needed

Iterative improvement search

- choose a single current state, sub-optimal
- gradually modify current state
- generally visiting ‘neighbors’
- until reaching a near-optimal state

Example: complete-state formulation of N-queens
Main advantages of local search techniques

1. Memory (usually a constant amount)
2. Find reasonable solutions in large spaces
   where we cannot possibly search the space exhaustively
3. Useful for optimization problems:
   best state given an objective function (quality of the goal)
**Intuition:** state-scape landscape

- All states are laid up on the surface of a landscape
- A state’s location determines its neighbors (where it can move)
- A state’s elevation represents its quality (value of objective function)
- Move from one neighbor of the current state to another state until reaching the highest peak
**Two major classes**

1. Hill climbing (a.k.a. gradient ascent/descent)
   → try to make changes to improve quality of current state

2. Simulated Annealing (physics)
   → things can temporarily get worse

Others: tabu search, local beam search, genetic algorithms, etc.

→ Optimality (soundness)? Completeness?
→ Complexity: space? time?

→ In practice, surprisingly good..
   (eroding myth)
Hill climbing

Start from any state at random and loop:

Examine all direct neighbors

If a neighbor has higher value then move to it else exit

Problems:

- Local optima: (maxima or minima) search halts
- Plateau: flat local optimum or shoulder
- Ridge

evaluation

objective function

global maximum

shoulder

local maximum

“flat” local maximum

current state

state space
Plateaux

Allow sideways moves

- For shoulder, good solution
- For flat local optima, may result in an infinite loop
  Limit number of moves
Ridges

Sequence of local optima that is difficult to navigate
Variants of Hill Climbing

- Stochastic hill climbing: random walk
  Choose to disobey the heuristic, sometimes
  Parameter: How often?

- First-choice hill climbing
  Choose first best neighbor examined
  Good solution when we have too many neighbors

- Random-restart hill climbing
  A series of hill-climbing searches from random initial states
Random-restart hill-climbing

→ When HC halts or no progress is made
   re-start from a different (randomly chosen) starting
   save best results found so far

→ Repeat random restart
   - for a fixed number of iterations, or
   - until best results have not been improved for a certain
     number of iterations
Simulated annealing (I)

**Basic idea:** When stuck in a local maximum allow few steps towards less good neighbors to escape the local maximum

Start from any state at random, start count down and loop until time is over:

- Pick up a neighbor at **random**
- Set $\Delta E = \text{value(neighbor)} - \text{value(current state)}$
- **If** $\Delta E > 0$ (neighbor is better)
  - *then* move to neighbor
- **else** $\Delta E < 0$ move to it with probability $< 1$

Transition probability $\sim e^{\Delta E/T}$

<table>
<thead>
<tr>
<th>$\Delta E$ is negative</th>
<th>$T$: count-down time</th>
</tr>
</thead>
</table>

as time passes, less and less likely to make the move towards ‘unattractive’ neighbors
Simulated annealing (II)

Analogy to physics:
Gradually cooling a liquid until it freezes
If temperature is lowered sufficiently slowly, material
will attain lowest-energy configuration (perfect order)

\[
\begin{align*}
\text{Count down} & \quad \longleftrightarrow \quad \text{Temperature} \\
\text{Moves between states} & \quad \longleftrightarrow \quad \text{Thermal noise} \\
\text{Global optimum} & \quad \longleftrightarrow \quad \text{Lowest-energy configuration}
\end{align*}
\]
How about decision problems?

<table>
<thead>
<tr>
<th>Optimization problems</th>
<th>Decision problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterative improvement</td>
<td>Iterative repair</td>
</tr>
<tr>
<td>State value</td>
<td>Number of constraints violated</td>
</tr>
<tr>
<td>Sub-optimal state</td>
<td>Inconsistent state</td>
</tr>
<tr>
<td>Optimal state</td>
<td>Consistent state</td>
</tr>
</tbody>
</table>
Local beam search

- Keeps track of $k$ states
- Mechanism:
  Begins with $k$ states
  At each step, all successors of all $k$ states generated
  Goal reached? Stop.
  Otherwise, selects $k$ best successors, and repeat.
- Not exactly a $k$ restarts: $k$ runs are not independent
- Stochastic beam search increases diversity
Genetic algorithms

- Basic concept: combines two (parent) states
- Mechanism:
  Starts with $k$ random states (population)
  Encodes individuals in a compact representation (e.g., a string in an alphabet)
  Combines partial solutions to generate new solutions (next generation)
Important components of a genetic algorithm

- Fitness function ranks a state’s quality, assigns probability for selection
- Selection randomly chooses pairs for combinations depending on fitness
- Crossover point randomly chosen for each individual, offsprings are generated
- Mutation randomly changes a state