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

• Hill-climbing

• Simulated annealing

• ...
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 layed 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
Plateaux

Allow sideway 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)

$\text{then}$ move to neighbor

else $\Delta E < 0$ move to it with probability $< 1$

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

\[
\begin{align*}
\Delta E \text{ is negative} \\
T: \text{count-down time}
\end{align*}
\]

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)

Count down $\leftrightarrow$ Temperature
Moves between states $\leftrightarrow$ Thermal noise
Global optimum $\leftrightarrow$ Lowest-energy configuration
How about decision problems?

Optimization problems  Decision problems

Iterative improvement  ↔  Iterative repair
State value  ↔  Number of constraints violated
Sub-optimal state  ↔  Inconsistent state
Optimal state  ↔  Consistent state
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