Chapter 4 Beyond Classical Search
4.1 Local search algorithms and optimization problems

CS4811 - Artificial Intelligence

Nilufer Onder
Department of Computer Science
Michigan Technological University
Outline

Hill climbing search

Simulated annealing

Local beam search

Genetic algorithms
Iterative improvement algorithms

- In the problems we studied so far, the solution is the path. For example, the solution to the 8-puzzle is a series of movements for the “blank tile.” The solution to the traveling in Romania problem is a sequence of cities to get to Bucharest.

- In many optimization problems, the path is irrelevant. The goal itself is the solution.

- The state space is set up as a set of “complete” configurations, the optimal configuration is one of them.

- An *iterative improvement algorithm* keeps a single “current” state and tries to improve it.

- The space complexity is constant!
Example: Travelling Salesperson Problem

Start with any complete tour, perform pairwise exchanges

A C B D E

B C
D
A E
A B C D E

Variants of this approach get within 1% of optimal very quickly with thousands of cities.
Example: $n$-queens

Put $n$ queens on an $n \times n$ board with no two queens on the same row, column, or diagonal.

Move a queen to reduce the number of conflicts ($h$).

Almost always solves $n$-queens problems almost instantaneously for very large $n$, e.g., $n = 1$ million.
Example: $n$-queens (cont’d)

(a) shows the value of $h$ for each possible successor obtained by moving a queen within its column. The marked squares show the best moves.

(b) shows a local minimum: the state has $h = 1$ but every successor has higher cost.
Hill-climbing (or gradient ascent/descent)

function Hill-Climbing (problem) returns a state that is a local maximum

**inputs:** problem, a problem
**local variables:**
- current, a node
- neighbor, a node

\[ current \leftarrow \text{Make-Node}(\text{problem}.\text{Initial-State}) \]

**loop do**
- \[ neighbor \leftarrow \text{a highest-valued successor of current} \]
  - **if** neighbor.Value \( \leq \) current.Value **then return** current.State
  - \[ current \leftarrow \text{neighbor} \]
Hill-climbing (cont’d)

- “Like climbing Everest in thick fog with amnesia.”
- Problem: depending on initial state, can get stuck on local maxima
- In continuous spaces, problems with choosing step size, slow convergence
Difficulties with ridges

The “ridge” creates a sequence of local maxima that are not directly connected to each other. From each local maximum, all the available actions point downhill.
Hill-climbing techniques

- stochastic: choose randomly from uphill moves
- first-choice: generate successors randomly one-by-one until one better than the current state is found
- random-restart: restart with a randomly generated initial state
Simulated annealing

function Simulated Annealing (problem, schedule)
returns a solution state

inputs:
  problem, a problem
  schedule, a mapping from time to “temperature”

local variables:
  current, a node
  next, a node

current ← Make-Node(problem.Initial-State)
for $t = 1$ to $\infty$ do
  $T ← schedule(t)$
  if $T=0$ then return current
  next ← a randomly selected successor of current
  $\Delta E ← next.Value - current.Value$
  if $\Delta E > 0$ then current ← next
  else current ← next only with probability $e^{\Delta E/T}$
Simulated annealing (cont’d)

► Idea: escape local maxima by allowing some “bad” moves but gradually decrease their size and frequency.
► Devised by Metropolis et al., 1953, for physical process modelling.
► At fixed “temperature” $T$, state occupation probability reaches Boltzman distribution

$$p(x) = \alpha e^{\frac{E(x)}{kT}}$$

► When $T$ is decreased slowly enough it always reaches the best state $x^*$ because $e^{\frac{E(x^*)}{kT}}/e^{\frac{E(x)}{kT}} = e^{\frac{E(x^*)-E(x)}{kT}} \gg 1$ for small $T$. (Is this necessarily an interesting guarantee?)
► Widely used in VLSI layout, airline scheduling, etc.
Local beam search

- Idea: keep $k$ states instead of 1; choose top $k$ of all their successors
- Not the same as $k$ searches run in parallel! Searches that find good states recruit other searches to join them.
- Problem: quite often, all $k$ states end up on same local hill.
- To improve: choose $k$ successors randomly, biased towards good ones.
- Observe the close analogy to natural selection!
The genetic algorithm

**function** Genetic Algorithm (**problem**, **Fitness-Fn**)  
**returns** an individual

**inputs:**
- **population**, a set of individuals
- **Fitness-Fn**, a function that measures the fitness of an individual

**repeat**

**new-population** ← empty set

**for** *i* = 1 to **Size**(**population**) **do**

- *x* ← Random-Selection(**population**, **Fitness-Fn**)  
- *y* ← Random-Selection(**population**, **Fitness-Fn**)  
- **child** ← Reproduce(*x*,*y*)  
- **if** (small random probability) **then** **child** ← Mutate(**child**)  

add **child** to **new-population**

**population** ← **new-population**

**until** some individual is fit enough, or enough time has elapsed

**return** the best individual in **population**, according to **Fitness-Fn**
The crossover function

function REPRODUCE (x, y)
returns an individual

inputs:
   x, y, parent individuals

n ← LENGTH(x)
c ← random number from 1 to n
return APPEND(STRING(x, 1, c), STRING(y, c + 1, n))
Genetic algorithms (GAs)

- Idea: stochastic local beam search + generate successors from pairs of states
- GAs require states encoded as strings.
- Crossover helps iff substrings are meaningful components.
- GAs $\neq$ evolution.
  e.g., real genes encode replication machinery.
Genetic algorithm example

(a) Initial Population  
(b) Fitness Function  
(c) Selection  
(d) Crossover  
(e) Mutation
The genetic algorithm with the 8-queens problem
Summary

- Hill climbing is a steady monotonous ascent to better nodes.
- Simulated annealing, local beam search, and genetic algorithms are “random” searches with a bias towards better nodes.
- All need very little space which is defined by the population size.
- None guarantees to find the globally optimal solution.
Sources for the slides

- AIMA textbook (3rd edition)
- AIMA slides (http://aima.cs.berkeley.edu/)