Informed Search II

1. When A* fails – Hill climbing, simulated annealing
2. Genetic algorithms

Outline

• Local Search: Hill Climbing
• Escaping Local Maxima: Simulated Annealing
• Genetic Algorithms

Local search and optimization

• Local search:
  • Use single current state and move to neighboring states.
  • Idea: start with an initial guess at a solution and incrementally improve it until it is one
  • Advantages:
    • Use very little memory
    • Find often reasonable solutions in large or infinite state spaces.
  • Also useful for pure optimization problems.
    • Find best state according to some objective function.
    • e.g. survival of the fittest as a metaphor for optimization.

Hill Climbing on a surface of states

h(s): Estimate of distance from a peak (smaller is better)

Height Defined by Evaluation Function (greater is better)
Hill-climbing search

I. While (3 uphill points):
   • Move in the direction of increasing evaluation or, equivalently, decreasing h.

II. If (3 a successor \( s_i \) for the current state \( n \) such that
   — \( h(s_i) < h(n) \)
   — \( h(s_j) \geq h(s_i) \) for all successors \( s_j \) of \( n \), \( j \neq i \):
     • then move from \( n \) to \( s_i \).
     • Otherwise, halt at \( n \).

Properties:
• Terminates when a peak is reached.
• Does not look ahead of the immediate neighbors of the current state.
• Chooses randomly among the set of best successors, if there is more than one.
• Doesn’t backtrack, since it doesn’t remember where it’s been
• a.k.a. greedy local search

"Like climbing Everest in thick fog with amnesia"

Hill-climbing Example I

<table>
<thead>
<tr>
<th>Start state</th>
<th>Goal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 1 2</td>
<td>1 2</td>
</tr>
<tr>
<td>4 5 8</td>
<td>3 4 5</td>
</tr>
<tr>
<td>6 7</td>
<td>6 7 8</td>
</tr>
</tbody>
</table>

Hill-climbing Example: \( n \)-queens

\( n \)-queens problem: Put \( n \) queens on an \( n \times n \) board with no two queens on the same row, column, or diagonal

Good heuristic: \( h = \) number of pairs of queens that are attacking each other

Hill-climbing example: 8-queens

A state with \( h=17 \) and the \( h \)-value for each possible successor

A local minimum of \( h \) in the 8-queens state space (\( h=1 \)).

\( h \) = number of pairs of queens that are attacking each other

Search Space features

Drawbacks of hill climbing

• Local Maxima: peaks that aren’t the highest point in the space
• Plateaus: the space has a broad flat region that gives the search algorithm no direction (random walk)
• Ridges: dropoffs 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 maximum

The Shape of an Easy Problem (Convex)

Gradient ascent/descent

Gradient methods vs. Newton’s method

The Shape of a Harder Problem

The Shape of a Yet Harder Problem
One Remedy to Drawbacks of Hill Climbing: Random Restart

- In the end: Some problem spaces are great for hill climbing and others are terrible.

Simulated Annealing

Simulated annealing (SA)

- **Annealing**: the process by which a metal cools and freezes into a minimum-energy crystalline structure (the annealing process)
- SA exploits an analogy between annealing and the search for a minimum in a more general system.
  - Switch viewpoint from hill-climbing to gradient descent
- SA can avoid becoming trapped at local minima.
- SA uses a random search that accepts changes that decrease objective function \( f \), as well as some that increase it.
- SA uses a control parameter \( T \), which by analogy with the original application is known as the system "temperature."
- \( T \) starts out high and gradually decreases toward 0.

Simulated annealing (cont.)

- A "bad" move from A to B (\( f(B) < f(A) \)) is accepted with the probability
  \[
  P(\text{move}_{A \rightarrow B}) = e^{\frac{(f(B) - f(A))}{T}}
  \]
- The higher \( T \), the more likely a bad move will be made.
- As \( T \) tends to zero, this probability tends to zero, and SA becomes more like hill climbing
- If \( T \) is lowered slowly enough, SA is complete and admissible.

Local beam search

- Keep track of \( k \) states instead of one
  - Initially: \( k \) random states
  - Next: determine all successors of \( k \) states
  - If any of successors is goal → finished
  - Else select \( k \) best from successors and repeat.
- Major difference with random-restart search
  - Information is shared among \( k \) search threads.
- Can suffer from lack of diversity.
  - Stochastic variant: choose \( k \) successors proportionally to state success.

Genetic Algorithms
Genetic algorithms

1. Start with $k$ random states (the initial population)
2. New states are generated by either
   1. “Mutation” of a single state or
   2. “Sexual Reproduction”: (combining) two parent states (selected proportionally to their fitness)

- Encoding used for the “genome” of an individual strongly affects the behavior of the search
- Similar (in some ways) to stochastic beam search

Representation: Strings of genes

- Each chromosome
  - represents a possible solution
  - made up of a string of genes
- Each gene encodes some property of the solution
- There is a fitness metric on phenotypes of chromosomes
  - Evaluation of how well a solution with that set of properties solves the problem.
- New generations are formed by
  - Crossover: sexual reproduction
  - Mutation: asexual reproduction

Encoding of a Chromosome

- The chromosome encodes characteristics of the solution which it represents, often as a string of binary digits.

| Chromosome 1 | 11011000011010110 |
| Chromosome 2 | 110111100011110110 |

- Each set of bits represents some dimension of the solution.

Example: Genetic Algorithm for Drive Train

Genes for:
- Number of Cylinders
- RPM: 1st -> 2nd
- RPM 2nd -> 3rd
- RPM 3rd -> Drive
- Rear end gear ratio
- Size of wheels

A chromosome specifies a full drive train design

Reproduction

- Reproduction by crossover selects genes from two parent chromosomes and creates two new offspring.
- To do this, randomly choose a crossover point (perhaps none).
- For child 1, everything before this point comes from the first parent and everything after from the second parent.
- Crossover looks like this ($|$ is the crossover point):

| Chromosome 1 | 11001 | 00100110110 |
| Chromosome 2 | 10111 | 11000011110 |
| Offspring 1  | 11001 | 11000011110 |
| Offspring 2  | 10111 | 00100110110 |

Mutation

- Mutation randomly changes genes in the new offspring.
- For binary encoding we can switch randomly chosen bits from 1 to 0 or from 0 to 1.

| Original offspring | 1101111000011110 |
| Mutated offspring  | 1101111000011110 |
The Basic Genetic Algorithm

1. Generate random population of chromosomes
2. Until the end condition is met, create a new population by repeating following steps
   1. Evaluate the fitness of each chromosome
   2. Select two parent chromosomes from a population, weighed by their fitness
   3. With probability $p_c$ cross over the parents to form a new offspring.
   4. With probability $p_m$ mutate new offspring at each position on the chromosome.
   5. Place new offspring in the new population
3. Return the best solution in current population

Genetic algorithms: 8-queens

A Genetic Algorithm Simulation

The Chromosome Layout

- Strengths:
  - Vector Angles and Magnitudes adjacent
  - Adjacent vectors are adjacent
- Weakness:
  - Wheel info (vertex, axle angles & wheel radiiues not linked to vector the wheel is associated with.)

Car from Gen 4: Score: 160 (max)

Best from Generations 20-46: 594.7