When $A^*$ doesn’t work

AIMA 4.1

A few slides adapted from CS 471, UBMC and Eric Eaton (in turn, adapted from slides by Charles R. Dyer, University of Wisconsin-Madison)
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.

- **Useful for pure optimization problems.**
  - Find or approximate best state according to some *objective function*
  - *Optimal* if the space to be searched is *convex*
Outline

- **Local Search: Hill Climbing**
- Escaping Local Maxima: Simulated Annealing
- Genetic Algorithms
Hill climbing on a surface of states

\[ h(s): \text{Estimate of distance from a peak (smaller is better)} \]

OR: \[ f(s): \text{Height Defined by Evaluation Function (greater is better)} \]
Hill-climbing search

I. While (∃ uphill points):
   Move in the direction of increasing evaluation function \( f(s) \)

II. Let \( s_{next} = \arg \max_{s} f(s) \), \( s \) a successor state to the current state \( n \)
   - If \( f(n) < f(s_{next}) \) then move to \( s_{next} \)
   - Otherwise halt at \( n \)

- Extremely simple:
  - 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"
Toy hill climbing example I (minimizing $h$)
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: $h = \text{number of pairs of queens that are attacking each other}$
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

- objective function
- global maximum
- shoulder
- local maximum
- “flat” local maximum
- current state
- state space
Drawbacks of hill climbing

- **Local Maxima:** peaks that aren’t the highest point in the space
- **Plateaus:** broad flat regions with no indication of “up hill”
- **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.
Toy Example of a local "maximum"

- **start**
  - 4 1 2
  - 3 5
  - 6 7 8

- **goal**
  - 1 2
  - 3 4 5
  - 6 7 8

1. **4 1 2**
   - Start position.

2. **3 1 5**
   - Local maximum.

3. **6 7 8**
   - Goal position.

Each move represents a step in the solution process.
The Shape of an Easy Problem (*Convex*)

*This and next several slides from Goldberg '89*
Gradient ascent/descent

- Gradient descent procedure for finding the $\arg_{x} \min f(x)$
  - choose initial $x_0$ randomly
  - repeat
    - $x_{i+1} \leftarrow x_i - \eta f'(x_i)$
  - until the sequence $x_0, x_1, \ldots, x_i, x_{i+1}$ converges
- Step size $\eta$ (eta) is small (perhaps 0.1 or 0.05)
Gradient methods vs. Newton’s method

• A reminder of Newton's method from Calculus:
  \[ x_{i+1} \leftarrow x_i - \eta f'(x_i) / f''(x_i) \]

• Newton's method uses 2nd order information (the second derivative, or, curvature) to take a more direct route to the minimum.

• The second-order information is more expensive to compute, but converges quicker.


Contour lines of a function
Gradient descent (green)
Newton's method (red)

(this and previous slide from Eric Eaton)
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.
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 $\rightarrow$ 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.
Outline

- Local Search: Hill Climbing
- Escaping Local Maxima: Simulated Annealing
- Genetic Algorithms
Simulated annealing (SA)

- **Annealing**: the process by which a metal cools slowly and as a result freezes into a *minimum-energy* crystalline structure.

- Conceptually SA exploits an analogy between annealing and the search for a *minimum* in a more general system.

- 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 (SA) hill climbing

- **BUG IN TEXT!!!**
  - AIMA: Switch viewpoint from hill-climbing to gradient descent
  - *(But: AIMA algorithm hill-climbs & larger $\Delta E$ is good…)*

- SA uses a random search that occasionally accepts changes that decrease objective function $f$.

- *Probability of accepting lower $f$ decreases with $T$*

- **SA hill-climbing** can avoid becoming trapped at local maxima.
Simulated annealing (cont.)

- A "bad" move from A to B \((f(B)<f(A))\) is accepted with the probability

\[
P(\text{move}_{A\rightarrow B}) = e^{(f(B) - f(A)) / T}
\]

- 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.
Applicability

- Discrete Problems where state changes are transforms of local parts of the configuration
  - E.G. Travelling Salesman problem, where moves are swaps of the order of two cities visited:
    - Pick an initial tour randomly
    - Successors are all neighboring tours, reached by swapping adjacent cities in the original tour
    - Search using simulated annealing..
AIMA Simulated Annealing Algorithm

function SIMULATED-ANNEALING( problem, schedule) returns a solution state
input: problem, a problem
       schedule, a mapping from time to “temperature”

   current ← MAKE-NODE(problem.INITIAL-STATE)
   for t ← 1 to ∞ do
       T ← schedule(t)
       if T = 0 then return current
       next ← a randomly selected successor of current
       \( \Delta E \leftarrow next.VALUE - current.VALUE \)
       if \( \Delta E > 0 \) then current ← next
       else current ← next only with probability \( e^{\Delta E / T} \)

Nice simulation on web page of travelling salesman approximations via simulated annealing:

Outline

- Local Search: Hill Climbing
- Escaping Local Maxima: Simulated Annealing
- *Genetic Algorithms*
Genetic algorithms

1. Start with $k$ random states (the *initial population*)

2. New states are generated by either
   1. “Sexual Reproduction”: (combining) two *parent states* (selected proportionally to their *fitness*)
   2. “Mutation” of a single state or

   - 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  1101100100110110
  Chromosome 2  1101111000011110

- Each set of bits represents some dimension of the solution.
Example: Genetic Algorithm for Drive Train

Genes for:

- Number of Cylinders
- RPM: $1^{st}$ -> $2^{nd}$
- RPM $2^{nd}$ -> $3^{rd}$
- RPM $3^{rd}$ -> 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  10011 | 11000011110

  Offspring 1  11001 | 11000011110
  Offspring 2  10011 | 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: 1100111000001110
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

\[
\begin{array}{c}
\begin{array}{c}
\text{Chessboard 1} \\
\text{Chessboard 2}
\end{array}
\end{array} + \begin{array}{c}
\text{Chessboard 2} \\
\text{Chessboard 3}
\end{array} = \begin{array}{c}
\text{Chessboard 3}
\end{array}
\]
A Genetic Algorithm Simulation

BoxCar 2D

www.boxcar2d.com
The Chromosome Layout

- **Strengths:**
  - Vector Angles and Magnitudes adjacent
  - Adjacent vectors are adjacent
- **Weakness:**
- Wheel info (vertex, axle angles & wheel radiuses not linked to vector the wheel is associated with.)
Best from Generations 20-46: 594.7