Beyond Classical Search

Joe Johnson, M.S., Ph.D., A.S.A.

Beyond Classical Search - Topics

- Local Search – Definition
- Measures of Local Search
  - Completeness
  - Optimality
- Types of local searches
  - Hill Climbing
  - Simulated Annealing
  - Local Beam search
  - Genetic Algorithms

Beyond Classical Search - Background

- In lectures 2 and 3, we looked at problem domains which had the following characteristics:
  - Fully Observable
  - Deterministic
  - Solution is a sequence of actions (i.e., our solution is a *path*).

Beyond Classical Search - Background

- No-Path Problem Domains
  - In many optimization problems, the path to the goal is irrelevant; the goal state itself is the solution
  - State space = set of "complete" configurations
  - Find configuration satisfying constraints, e.g., n-queens
  - In such cases, we can use local search algorithms
    - Keep a single "current" state, try to improve it
    - Do not retain all of the paths.

Local Search

- Local Search Algorithm
  - Definition
    - A local search algorithm is a type of search algorithm which maintains only a current state (no already-explored states) and which moves only to adjacent neighbors in accordance with optimizing some objective function.
    - Path is not important – only the goal is important
      - Thus, paths are not retained.
    - Not systematic
      - i.e., we do not traverse the state space in a systematic way, as we did in classical search (BFS, DFS, LDFS, etc.)

Local Search

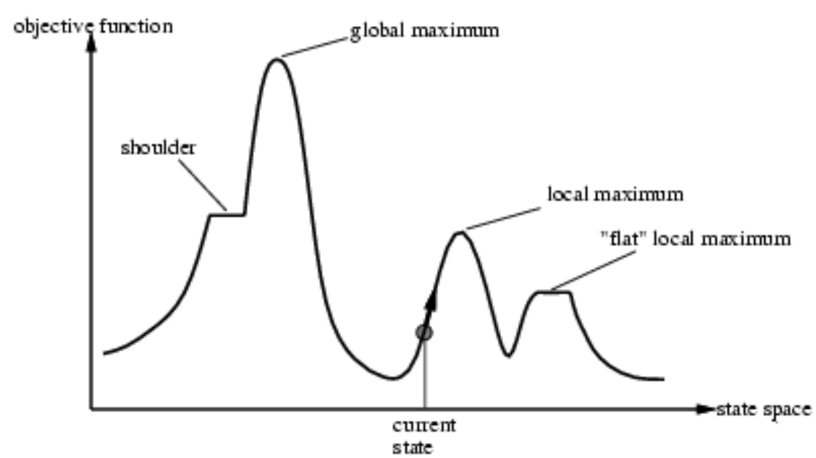
- Advantages
  - Use very little memory (usually constant).
  - Often find reasonable solutions in large or infinite (continuous) search spaces for which systematic algorithms are unsuitable.
  - Useful for solving pure optimization problems, where the aim is to find the best state according to an objective function.

Local Search

- What Local Search Algorithms are not good for:
  - Do not fit the problems we discussed in lecture 02 – Classical Search – in which our search result should be a path, i.e. a sequence of nodes in the search space.

Local Search

- State-space Landscape
  - Consists of both location (state) and elevation (value of the heuristic cost function or objective function)
  - If elevation corresponds to cost, try to find the lowest valley.
  - If elevation corresponds to objective function, find the highest peak.



Local Search

- Completeness
  - Always finds a goal if one exists.
- Optimality
  - Always finds a global minimum/maximum.

Example: *n*-queens

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

Types of Local Searches

- Hill Climbing
- Simulated Annealing
- Local Beam search
- Genetic Algorithms

Hill Climbing Search

- Hill Climbing Search
  - Definition
    - Is a form of local search which walks from node to neighboring node in the state space in which the neighbor selected *offers the largest increase in the objective function,* and if all neighbors have an objective function value less than that of the current node, the search terminates.

Hill Climbing search

- "Like climbing Everest in thick fog with amnesia"

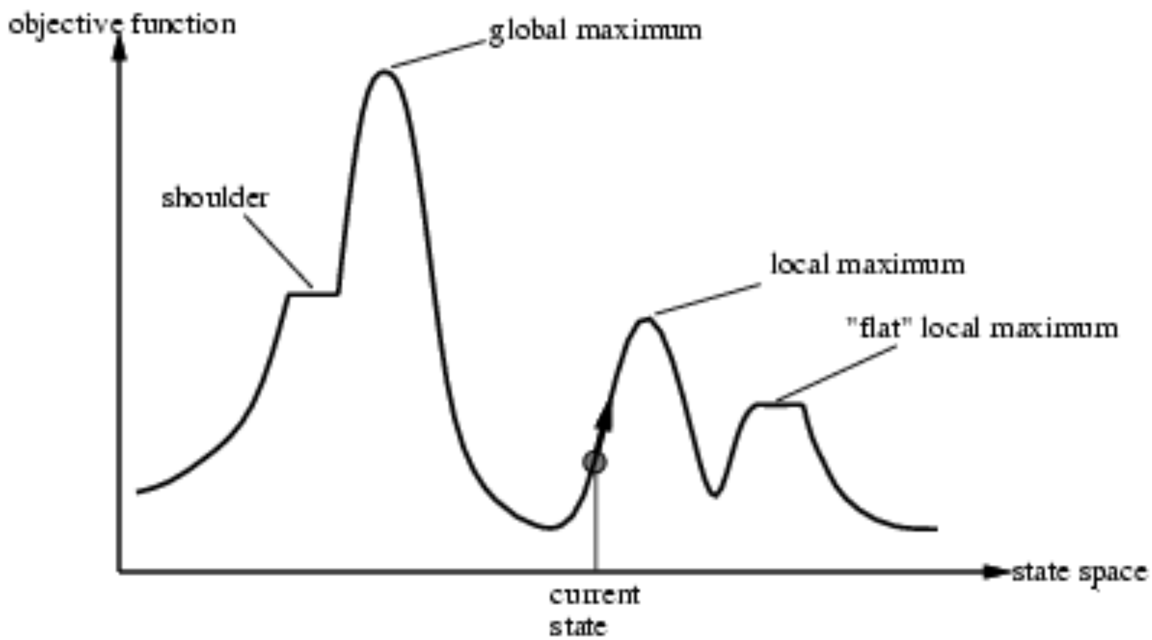
```
function HILL-CLIMBING(problem) returns a state that is a local maximum
   inputs: problem, a problem
   local variables: current, a node
                    neighbor, a node

   current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
       neighbor ← a highest-valued successor of current
       if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
       current ← neighbor
```

Hill Climbing Search

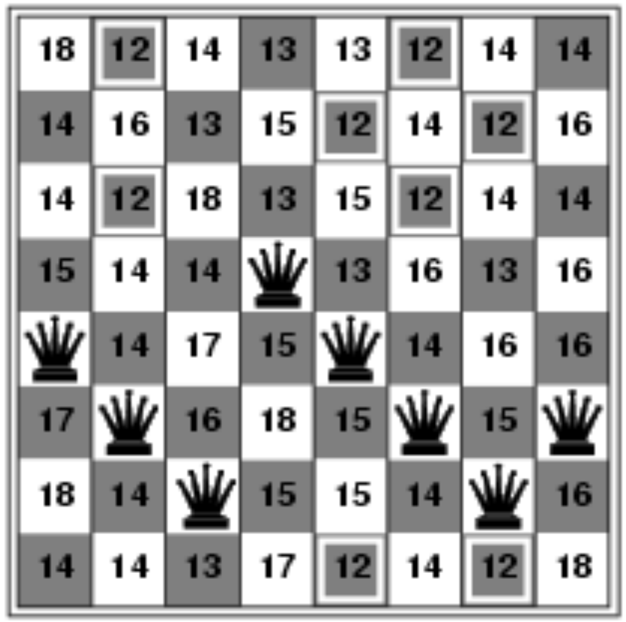
- Hill Climbing
  - Greedy Search Algorithm
    - Does NOT look beyond the set of adjacent nodes of the current node.
  - Complete State Formulation
    - Each state consists of entire layout of a possible solution
    - No state represents a partial solution.
    - Example – 8-queens
      - Each state consists of a complete layout of 8 queens on the board.

Hill Climbing Search

- Problem: depending on initial state, can get stuck in local maxima

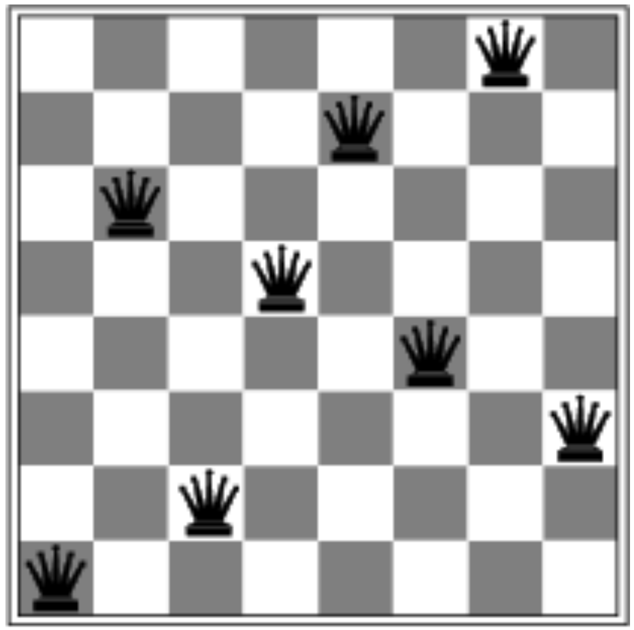


Hill Climbing Search: 8-queens problem



- *h* = number of pairs of queens that are attacking each other, either directly or indirectly
- *h = 17* for the above state

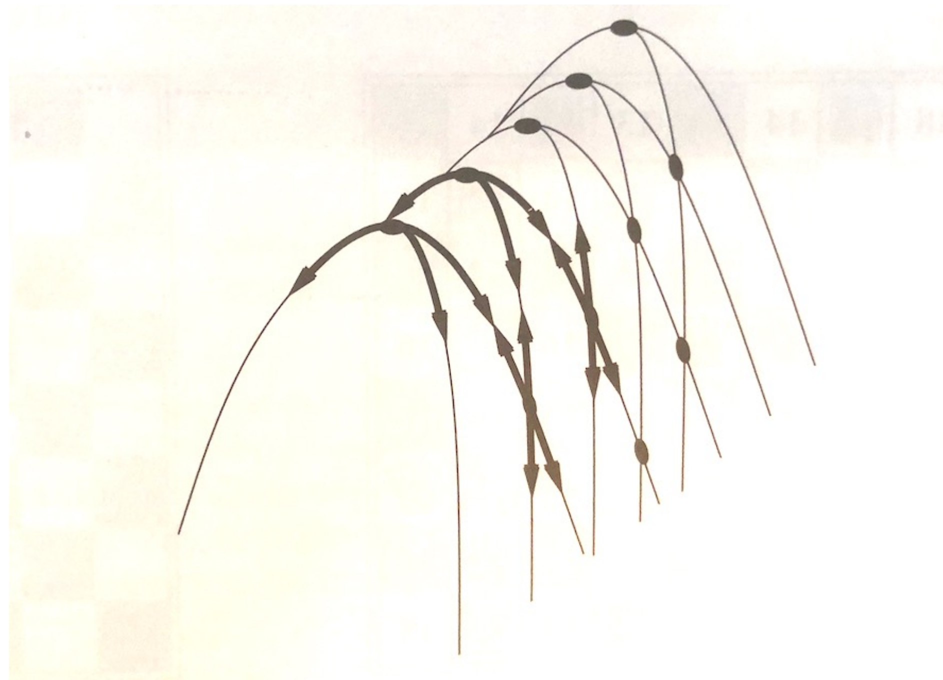
Hill Climbing Search: 8-queens problem



- A local minimum with $h = 1$, i.e., every successor node from this state has a higher cost.

Hill Climbing Search - Disadvantages

- Problems with Hill Climbing
  - Local maxima/minima.
  - Ridges – as in figure
  - Plateaux



Hill Climbing Search - Variants

- Variants of Hill Climbing
  - Stochastic Hill Climbing
  - First-choice Hill Climbing
  - Random-restart Hill Climbing

Stochastic Hill Climbing

- Stochastic hill climbing
  - Next node chosen is based on a probability distribution.
  - Probability may be weighted based on steepness of the uphill move.
  - Converges more slowly than the steepest ascent, but in some cases it finds better overall solutions.

First-Choice Hill Climbing

- First Choice Hill Climbing
  - Implements stochastic hill climbing by repeatedly generating successors randomly until one is created that is better than the current state.
  - Useful when there are a lot of successors (perhaps thousands).

Random-Restart Hill Climbing

- Random-restart Hill Climbing
  - After finding a max, starts the algorithms all over again with a new start node and climbs from there.
  - "If at first you don’t succeed, try, try again…”
  - Trivially complete with probability approaching 1 as r, the number of restarts, approaches infinity.
    - Eventually, the algorithm will generate the global maximum as the initial state.

Random-Restart Hill Climbing

- Number of restarts necessary to find a solution
  - How many restarts should we expect to have to make before finding a global solution?
  - Assume probability of success on any particular hill climbing is *p*.
    - For 8-queens, *p* is approximately 0.14.
  - What distribution does a process have in which we have the following characteristics?
    - The process consists of repeated *independent* trials, which continue until the first success occurs.
    - Each trial has probability of success = *p*.
    - Answer:
      - Geometric distribution – recall your stats material ☺.

Random-Restart Hill Climbing

- Number of restarts analysis
  - For a geometric distribution, with parameter, *p*, the expected number of trials before the first success is given by:
    - $E(X) = 1/p$
      - where *p* is the probability of success on a given trial
      - and X is a random variable with a geometric distribution
  - Example – 8-queens
    - *p* is roughly 0.14
    - $E(X) = 1 / 0.14$ = approx. 7. (6 failures and then a success on 7^{th} trial).

Simulated Annealing - Background

- Simulated Annealing
  - Background
    - Any hill-climbing algorithm that never makes a downhill move is guaranteed to be incomplete because it can get stuck on local maxima.
    - A random walk that moves completely randomly from one node to the other is complete but very inefficient.
    - We seek to combine these in some way.

Simulated Annealing

- Simulated Annealing
  - Definition
    - Is a local search algorithm which combines elements of hill climbing with random walk in order to side step incompleteness while at the same time being less inefficient than a completely random walk.
  - Inspired by metallurgy - annealing
    - In order to discover harder materials, they are heated (softened) to very high temperatures and then allowed to cool to form a new, harder crystalline structure that is harder than the material was originally.

Simulated Annealing – Description

- Perspective Change
  - Let’s change our perspective from *maximizing* an objective function to *minimizing it.*
    - Can be accomplished by simply changing the sign of the objective function.
    - Convert from a *hill climbing* approach to *gradient descent*.

Simulated Annealing – Description

- Ping-pong Ball Analogy
  - Imagine the task of getting a ping-pong ball into the deepest crevice on a bumpy surface.
  - Suppose we just let the ball roll.
    - It will come to rest at a local minimum.
  - Suppose we shake the surface, however.
    - The ball will bounce out of the local minimum and continue rolling around.
  - Trick – Shake the surface just the right amount.
    - Hard enough to bounce the ball out of a local minimum.
    - Not so hard that it bounces out of the global minimum once it has been reached.

Simulated Annealing - Description

- Simulated Annealing Approach
  - Start by shaking the surface hard (temperature high).
  - As time goes on, gradually shake the surface with less intensity (temperature cooling down).
  - If we slow the process down slowly enough, this process approaches being complete with probability, $p = 1$.

Simulated Annealing - Algorithm

- Simulated Annealing Algorithm
  - Consists of a nested loop – (outer loop and an inner loop).
  - Inner loop is similar to stochastic hill climbing.
    - We pick a move to a neighbor randomly (as opposed to the best move).
  - If neighbor improves our objective function, definitely make the move.
  - If neighbor does *not* improve our objective function, make the move based on a probability whose value gradually declines over successive iterations of the outer loop.
    - At the start of the outer loop, the probability is high.
    - As time goes on, the probability becomes very low.
  - Terminate the algorithm after prescribed number of iterations of the outer loop.

Simulated Annealing - Algorithm

```
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
                   T, a "temperature" controlling prob. of downward steps

  current ← Make-Node(Initial-State[problem])
  for t ← 1 to ∞ do
      T ← schedule[t]
      if T = 0 then return current
      next ← a randomly selected successor of current
      ΔE ← Value[current] – Value[next]           // we look for smaller values for Value[]
      if ΔE > 0 then current ← next               // thus, we prefer ΔE > 0
      else current ← next only with probability e^(ΔE/T)   //ΔE < 0 ⟹ 0 < e^(ΔE/T) < 1
```

Local Beam Search

- Local Beam Search - Background
  - Local Search features storing only one state in memory.
  - Seems a bit extreme as a response to managing memory limitations.
  - Local Beam Search addresses this by storing k > 1 states in memory at any given time.

Local Beam Search

- Local Beam Search
  - Definition
    - Local Beam Search is a form of local search in which $k > 1$ states are maintained at each step in the search.
    - Start with k randomly generated states.
    - At each step, generate all successors of all k states.
    - If any of the states is a goal, then terminate.
    - Otherwise, pick the best k states and repeat the process.

Local Beam Search

- Observations
  - At first glance, it may seem that local beam search is the same as k hill climbing algorithms running in parallel.
    - But this is not correct.
  - In local beam search, useful information is passed among the parallel search threads.
  - The states that generate the best successors say to the others, “Come on over here, the grass is greener!”
  - Algorithm quickly abandons unfruitful searches and moves its resources to where the most progress is being made.

Local Beam Search

- Local Beam Search - Variant
  - Stochastic Beam Search
    - Pick the k successor states at random
    - Each state is chosen with a probability p that increases with increasing value of the state.
    - Prevents concentration of the search among a cluster of nodes in one small area of the search space.
    - Bears some resemblance to the process of natural selection
      - The successors (offspring) of a state (organism) populate the next generation according to its value (fitness).

Genetic Algorithm

- Genetic Algorithm
  - Definition
    - Is a variant of stochastic beam search in which successor states are generated by combining *two* parent states rather than by modifying a single state.
  - Analogy to natural selection also applies to GAs as it does for Local Beam Search
    - Except now we are dealing with sexual reproduction instead of asexual reproduction.

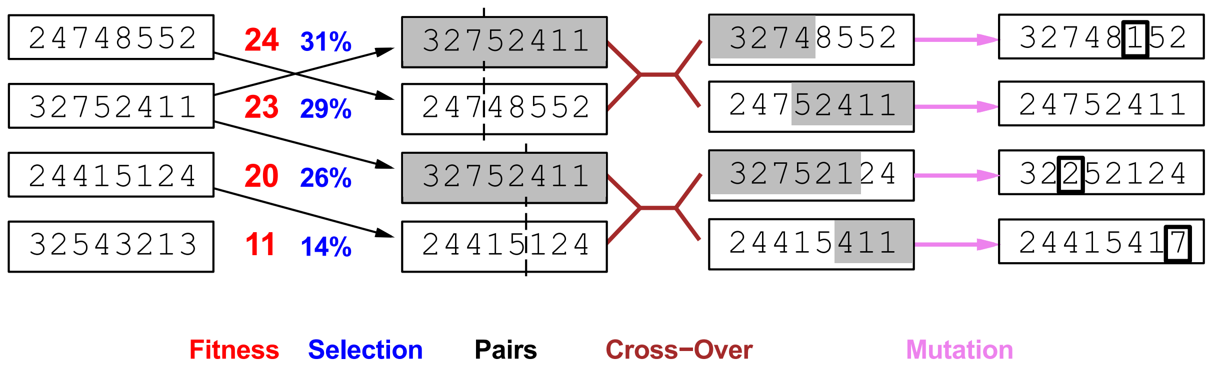
Genetic Algorithm

- Genetic Algorithm
  - Begin with a set of k randomly generated states, called a population.
  - Each state, or individual, is represented as a string over a finite alphabet – most commonly a string of 0s and 1s.
    - Example – 8 queens
      - Each queen is positioned at 1 of 8 possible locations in each column.
      - Thus, for each queen, we can represent its position as a $\log_2 8$ = 3-bit binary integer.
      - This approach applies for all 8 queens.
      - Thus, we can represent an 8-queen configuration on a board as a string consisting of 8 x $\log_2 8$ = 24 bits.
      - Alternatively, we could represent each queens position as a a digit on the interval 1-8, and thus, a complete configuration as an 8-digit string, each value 1-8.

Genetic Algorithm

- Genetic Algorithm
  - Each state is rated by its object function value, i.e., the fitness function.
    - Example – 8 queens
      - Fitness function could be the number of pairs of non-attacking queens.
      - What is the *max* value this function could have given we're working with 8 queens?
        - Answer: 8 choose 2 = 8!/(6!2!) = 56/2 = 28
  - Randomly select states from the set where the probability, p, of being selected is a function of the fitness function value for each state.
  - Select pairs at random to produce the next generation, and combine respective strings from the members of each pair to produce the offspring states.
  - Perform a genetic mutation randomly in the new states.

Genetic Algorithm – Example: 8 queens



- Start with k = 4 randomly generated states.
- Fitness function values: 24, 23, 20, 11
- Probabilities of selection are 0.31, 0.29, 0.26, and 0.14, respectively.
- Notice that the state 2 was chosen 2x, state 4 was chosen 0x for the pairing phase.
- Create new states by combining strings from parent pairs, which are randomly chosen.
- Crossover point is chosen randomly (the split point in the pairing states).
- Perform random mutations in the next generation of states.

Genetic Algorithm – Example: 8 queens

- 8-queens – Visual Representation