Local Search

Local Search:

- Maintain an assignment of a value to each variable.
- At each step, select a "neighbor" of the current assignment (e.g., one that improves some heuristic value).
- Stop when a satisfying assignment is found, or return the best assignment found.

Requires:

- What is a neighbor?
- Which neighbor should be selected?

(Some methods maintain multiple assignments.)

Selecting Neighbors in Local Search

- When the domains are small or unordered, the neighbors of an assignment can correspond to choosing another value for one of the variables.
- When the domains are large and ordered, the neighbors of an assignment are the adjacent values for one of the variables.
- If the domains are continuous, Gradient descent changes each variable proportional to the gradient of the heuristic function in that direction. The value of variable X_i goes from v_i to $v_i - \eta \frac{\partial h}{\partial X_i}$.

  Gradient ascent: go uphill; v_i becomes $v_i + \eta \frac{\partial h}{\partial X_i}$.

Local Search for CSPs

- Aim is to find an assignment with zero unsatisfied relations.
- Given an assignment of a value to each variable, a conflict is an unsatisfied constraint.
- The goal is an assignment with zero conflicts.
- Heuristic function to be minimized: the number of conflicts.

Greedy Descent Variants

- Find the variable-value pair that minimizes the number of conflicts at every step.
- Select a variable that participates in the most number of conflicts. Select a value that minimizes the number of conflicts.
- Select a variable that appears in any conflict. Select a value that minimizes the number of conflicts.
- Select a variable at random. Select a value that minimizes the number of conflicts.
- Select a variable and value at random; accept this change if it doesn't increase the number of conflicts.

Problems with Hill Climbing

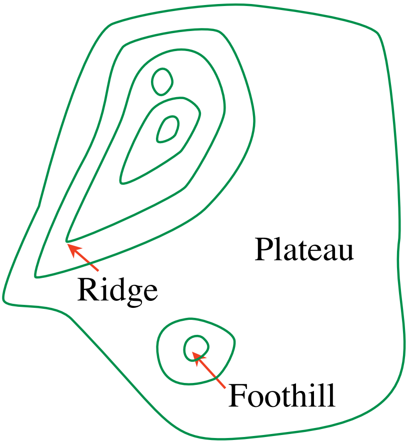
Greedy ascent (for finding maximum values) is called hill climbing.

Foothills local maxima that are not global maxima

Plateaus heuristic values are uninformative

Ridge foothill where *n*-step look-ahead might help

Ignorance of the peak



Randomized Algorithms

- Consider two methods to find a maximum value:
  - Hill climbing, starting from some position, keep moving uphill & report maximum value found
  - Pick values at random & report maximum value found
- Which do you expect to work better to find a maximum?
- Can a mix work better?

Randomized Hill Climbing

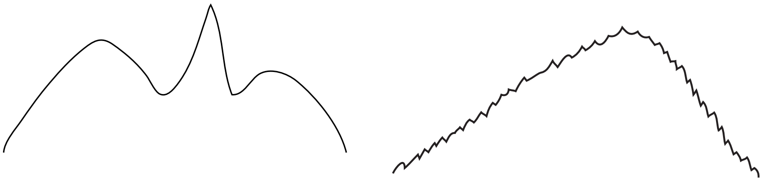
As well as uphill steps we can allow for:

- **Random steps:** move to a random neighbor.
- **Random restart:** reassign random values to all variables.

Which is more expensive computationally?

1-Dimensional Ordered Examples

Two 1-dimensional search spaces; step right or left:



- Which method would most easily find the maximum?
- What happens in hundreds or thousands of dimensions?
- What if different parts of the search space have different structure?

Stochastic Local Search

Stochastic local search is a mix of:

- Greedy descent: move to a lowest neighbor
- Random walk: taking some random steps
- Random restart: reassigning values to all variables

Random Walk

Variants of random walk:

- When choosing the best variable-value pair, randomly sometimes choose a random variable-value pair.
- When selecting a variable then a value:
  - Sometimes choose any variable that participates in the most conflicts.
  - Sometimes choose any variable that participates in any conflict (a red node).
  - Sometimes choose any variable.
- Sometimes choose the best value and sometimes choose a random value.

Comparing Stochastic Algorithms

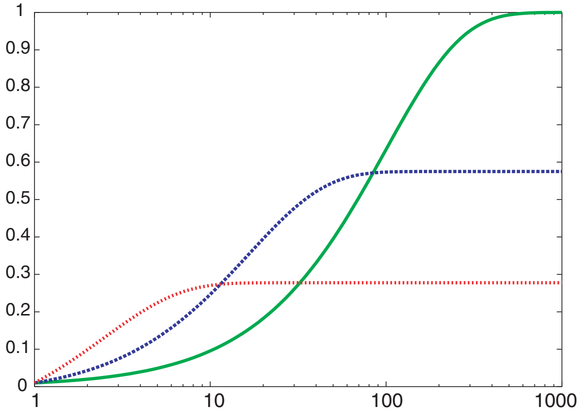
- How can you compare three algorithms when
  - one solves the problem 30% of the time very quickly but doesn't halt for the other 70% of the cases
  - one solves 60% of the cases reasonably quickly but doesn't solve the rest
  - one solves the problem in 100% of the cases, but slowly?

Comparing Stochastic Algorithms

- How can you compare three algorithms when
  - one solves the problem 30% of the time very quickly but doesn't halt for the other 70% of the cases
  - one solves 60% of the cases reasonably quickly but doesn't solve the rest
  - one solves the problem in 100% of the cases, but slowly?
- Summary statistics, such as mean run time, median run time, and mode run time don't make much sense.

Runtime Distribution

- Plots runtime (or number of steps) and the proportion (or number) of the runs that are solved within that runtime.



Variant: Simulated Annealing

- Pick a variable at random and a new value at random.
- If it is an improvement, adopt it.
- If it isn't an improvement, adopt it probabilistically depending on a temperature parameter, T.
  - With current assignment n and proposed assignment n' we move to n' with probability $e^{(h(n')-h(n))/T}$
- Temperature can be reduced.

Variant: Simulated Annealing

- Pick a variable at random and a new value at random.
- If it is an improvement, adopt it.
- If it isn't an improvement, adopt it probabilistically depending on a temperature parameter, T.
  - With current assignment n and proposed assignment n' we move to n' with probability $e^{(h(n')-h(n))/T}$
- Temperature can be reduced.

Probability of accepting a change:

Temperature	1-worse	2-worse	3-worse
10	0.91	0.81	0.74
1	0.37	0.14	0.05
0.25	0.02	0.0003	0.000005
0.1	0.00005	0	0

Tabu lists

- To prevent cycling we can maintain a **tabu list** of the k last assignments.
- Don't allow an assignment that is already on the tabu list.
- If $k = 1$, we don't allow an assignment of to the same value to the variable chosen.
- We can implement it more efficiently than as a list of complete assignments.
- It can be expensive if k is large.

Parallel Search

A total assignment is called an individual.

- Idea: maintain a population of k individuals instead of one.
- At every stage, update each individual in the population.
- Whenever an individual is a solution, it can be reported.
- Like k restarts, but uses k times the minimum number of steps.

Beam Search

- Like parallel search, with k individuals, but choose the k best out of all of the neighbors.
- When $k = 1$, it is hill climbing.
- When $k = \infty$, it is breadth-first search.
- The value of k lets us limit space and parallelism.

Stochastic Beam Search

- Like beam search, but it probabilistically chooses the k individuals at the next generation.
- The probability that a neighbor is chosen is proportional to its heuristic value.
- This maintains diversity amongst the individuals.
- The heuristic value reflects the fitness of the individual.
- Like asexual reproduction: each individual mutates and the fittest ones survive.

Genetic Algorithms

- Like stochastic beam search, but pairs of individuals are combined to create the offspring:
- For each generation:
  - Randomly choose pairs of individuals where the fittest individuals are more likely to be chosen.
  - For each pair, perform a cross-over: form two offspring each taking different parts of their parents:
  - Mutate some values.
- Stop when a solution is found.

Crossover

- Given two individuals:

$$X_1 = a_1, X_2 = a_2, \ldots, X_m = a_m$$

$$X_1 = b_1, X_2 = b_2, \ldots, X_m = b_m$$

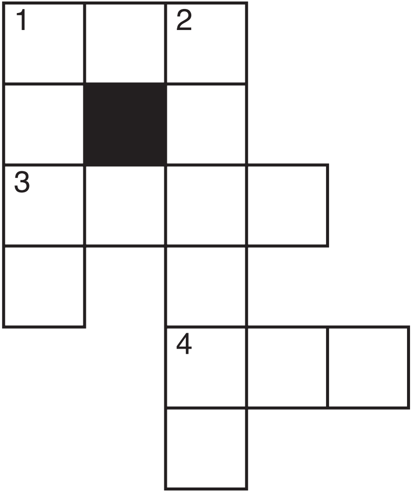
- Select i at random.
- Form two offspring:

$$X_1 = a_1, \ldots, X_i = a_i, X_{i+1} = b_{i+1}, \ldots, X_m = b_m$$

$$X_1 = b_1, \ldots, X_i = b_i, X_{i+1} = a_{i+1}, \ldots, X_m = a_m$$

- The effectiveness depends on the ordering of the variables.
- Many variations are possible.

Example: Crossword Puzzle



Words:

ant, big, bus, car, has
book, buys, hold,
lane, year
beast, ginger, search,
symbol, syntax