Local Search

CPSC 322 Lecture 12

January 30, 2006
Textbook §3.8

Lecture Overview

Local Search

A local search problem is defined by a:

- Set of Variables. A node in the search space will be a complete assignment to all of the variables.
- Neighbour relation. An edge in the search space will exist when the neighbour relation holds between a pair of nodes.
- Scoring function. This can be used to incorporate information about how many constraints are violated. It can also incorporate information about the cost of the solution in an optimization context.

Selecting Neighbours

How do we choose the neighbour relation?

- Usually this is simple: some small incremental change to the variable assignment
  - assignments that differ in one variable's value
  - assignments that differ in one variable's value, by a value difference of one
  - assignments that differ in two variables' values, etc.
- There's a trade-off: bigger neighbourhoods allow more nodes to be compared before a step is taken
  - the best step is more likely to be taken
  - each step takes more time: in the same amount of time, multiple steps in a smaller neighbourhood could have been taken
- Usually we prefer pretty small neighbourhoods

Hill Climbing

Hill climbing means selecting the neighbour which best improves the scoring function.

- For example, if the goal is to find the highest point on a surface, the scoring function might be the height at the current point.

Gradient Ascent

What can we do if the variable(s) are continuous?

- With a constant step size we could overshoot the maximum.
- Here we can use the scoring function h to determine the neighbourhood dynamically:
  - Gradient ascent: change each variable proportional to the gradient of the heuristic function in that direction.
  - The value of variable X_i goes from v_i to $v_i + \eta\frac{\partial h}{\partial X_i}$.
    - η is the constant of proportionality that determines how big steps will be
  - Gradient descent: go downhill; v_i becomes $v_i - \eta\frac{\partial h}{\partial X_i}$.
  - these partial derivatives may be estimated using finite differences

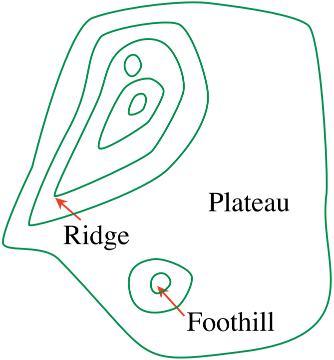
Problems with Hill Climbing

Foothills local maxima that are not global maxima

Plateaus heuristic values are uninformative

Ridge foothill where a larger neighbour relation would help

Ignorance of the peak no way of detecting a global maximum



Randomized Algorithms

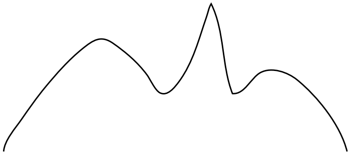
- Consider two methods to find a maximum value:
  - Hill climbing, starting from some position, keep moving uphill & report maximum value found
  - Pick values at random & report maximum value found
- Which do you expect to work better to find a maximum?
  - hill climbing is good for finding local maxima
  - selecting random nodes is good for finding new parts of the search space
- A mix of the two techniques can work even better

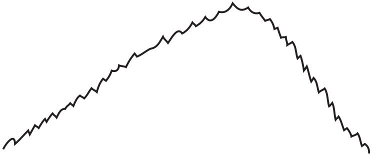
Stochastic Local Search

- We can bring these two ideas together to make a randomized version of 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?
  - usually, random restart (consider that there could be an extremely large number of neighbors)
  - however, if the neighbour relation is computationally expensive, random restart could be cheaper

1-Dimensional Ordered Examples

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





- Which of hill climbing with random walk and hill climbing with random restart would most easily find the maximum?
  - left: random restart; right: random walk
- As indicated before, stochastic local search often involves both kinds of randomization

Stochastic Local Search for CSPs

- Set of Variables: the same as the variables in the CSP
- Neighbour Relation: assignments that differ in the value assigned to one variable
- Goal is to find an assignment with all constraints satisfied.
  - Scoring function: the number of unsatisfied constraints.
  - We want an assignment with minimum score.

Greedy Descent

- Neighbour Relation: assignments that differ in the value assigned to one variable
- This means we have to evaluate our scoring function on a *lot* of different nodes for every step in the search
  - $\#$ variables $\times$ $\#$ values evaluations
- Instead, we might consider a restricted neighbourhood:
  - Values for the variable(s) that participate in the largest number of conflicts.
  - This alternative is easier to compute even if it doesn't always maximally reduce the number of conflicts.

Random Walk

You can add randomness:

- When choosing the best variable-value pair, randomly sometimes choose a random variable-value pair.
- When selecting a variable followed by a value:
  - Sometimes choose the variable which participates in the largest number of conflicts.
  - Sometimes choose, at random, any variable that participates in some conflict.
  - Sometimes choose a random variable.
  - Sometimes choose the best value for the chosen variable.
  - Sometimes choose a random value for the chosen variable.