Local Search

CPSC 322 – CSPs 4

Textbook §4.8

Lecture Overview

Arc Consistency Algorithm

- Consider the arcs in turn making each arc consistent.
  - Arcs may need to be revisited whenever the domains of other variables are reduced.
- Regardless of the order in which arcs are considered, we will terminate with the same result: an arc consistent network.

Revisiting Edges

- When we change the domain of a variable X in the course of making an arc $\langle X, r\rangle$ arc consistent, we add every arc $\langle Z, r'\rangle$ where r' involves X and:
  - $r \neq r'$
  - $Z \neq X$
- Thus we don't add back the same arc:
  - This makes sense—it's definitely arc consistent.

Revisiting Edges

- When we change the domain of a variable X in the course of making an arc $\langle X, r\rangle$ arc consistent, we add every arc $\langle Z, r'\rangle$ where r' involves X and:
  - $r \neq r'$
  - $Z \neq X$

- We don't add back other arcs involving the same variable X
  - We've just *reduced* the domain of X
  - If an arc $\langle X, r\rangle$ was arc consistent before, it will still be arc consistent
    - in the "for all" we'll just check fewer values

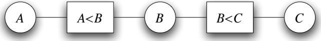
Revisiting Edges

- When we change the domain of a variable X in the course of making an arc $\langle X, r\rangle$ arc consistent, we add every arc $\langle Z, r'\rangle$ where r' involves X and:
  - $r \neq r'$
  - $Z \neq X$

- We don't add back other arcs involving the same constraint and a different variable:
  - Imagine that such an arc—involving variable Y—had been arc consistent before, but was no longer arc consistent after X's domain was reduced.
  - This means that some value in Y's domain could satisfy r only when X took one of the dropped values
  - But we dropped these values precisely because there were no values of Y that allowed r to be satisfied when X takes these values—contradiction!

Lecture Overview

Arc Consistency Example



- $dom(A) = \{1,2,3,4\}; dom(B) = \{1,2,3,4\}; dom(C) = \{1,2,3,4\}$
- Suppose you first select the arc $\langle A, A < B\rangle$.
  - Remove $A = 4$ from the domain of A.
  - Add nothing to TDA.
- Suppose that $\langle B, B < C\rangle$ is selected next.
  - Prune the value 4 from the domain of B.
  - Add $\langle A, A < B\rangle$ back into the TDA set (why?)
- Suppose that $\langle B, A < B\rangle$ is selected next.
  - Prune 1 from the domain of B.
  - Add no element to TDA (why?)
- Suppose the arc $\langle A, A < B\rangle$ is selected next
  - The value $A = 3$ can be pruned from the domain of A.
  - Add no element to TDA (why?)
- Select $\langle C, B < C\rangle$ next.
  - Remove 1 and 2 from the domain of C.
  - Add $\langle B, B < C\rangle$ back into the TDA set

The other two edges are arc consistent, so the algorithm terminates with $dom(A) = \{1,2\}$, $dom(B) = \{2,3\}$, $dom(C) = \{3,4\}$.

Lecture Overview

Local Search

- Many search spaces are too big for systematic search.
- A useful method in practice for some consistency and optimization problems is local search
  - idea: consider the space of complete assignments of values to variables
  - neighbours of a current node are similar variable assignments
  - move from one node to another according to a function that scores how good each assignment is

Local Search

Definition

A local search problem consists of a:

- CSP. In other words, a set of variables, domains for these variables, and constraints on their joint values. 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

Lecture Overview

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

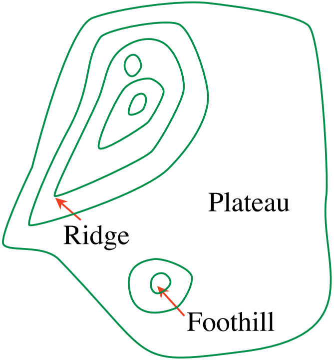
Problems with Hill Climbing

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



Lecture Overview

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?

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

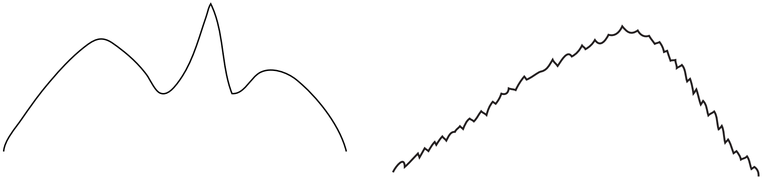
- We can bring these two ideas together to make a randomized version of hill climbing.
- As well as uphill steps we can allow for:
  - Random steps: move to a random neighbor.
  - Random restart: reassign random values to all variables.
- Which is more expensive computationally?

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

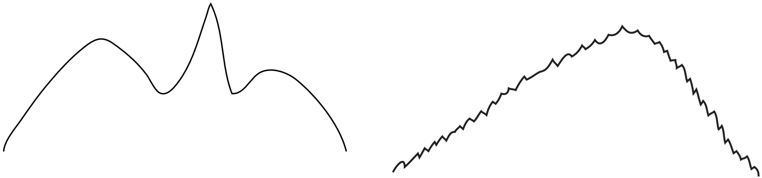
Two 1-dimensional search spaces; step right or left:



- Which of hill climbing with random walk and hill climbing with random restart would most easily find the maximum?

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

Random Walk

Some examples of ways to add randomness to local search for a CSP:

- 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.