Consistency Algorithms

Idea: prune the domains as much as possible before selecting values from them.

A variable is domain consistent if no value of the domain of the node is ruled impossible by any of the constraints.

Example: $\mathbf{D}_B = \{1, 2, 3, 4\}$ isn't domain consistent as $B = 3$ violates the constraint $B \neq 3$.

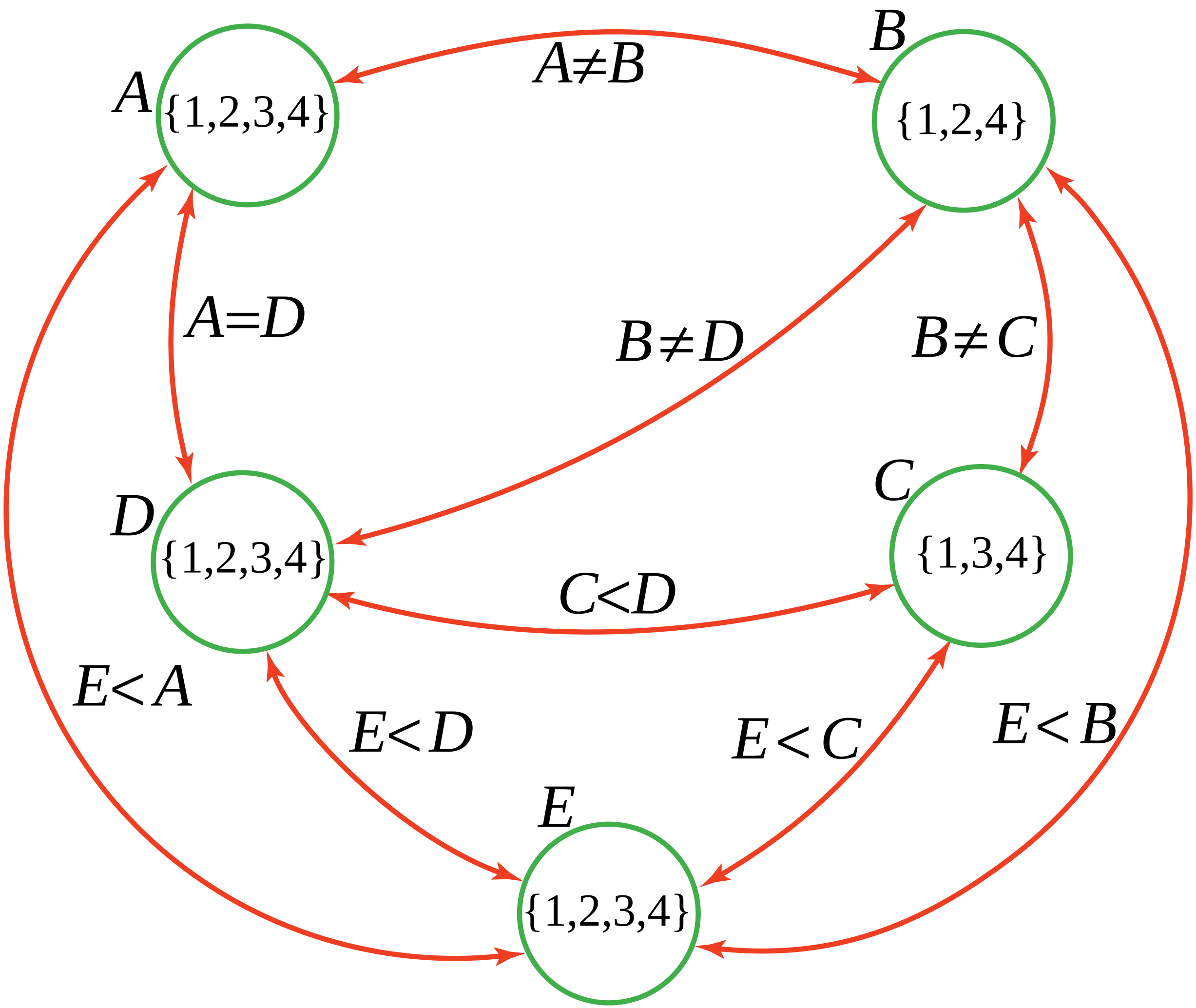
Arc Consistency

- A **constraint network** has nodes corresponding to variables with their associated domain. Each constraint relation $P(X, Y)$ corresponds to arcs $\langle X, Y \rangle$ and $\langle Y, X \rangle$.

- An arc $\langle X, Y \rangle$ is **arc consistent** if for each value of X in $\mathbf{D}_X$ there is some value for Y in $\mathbf{D}_Y$ such that $P(X, Y)$ is satisfied. A network is arc consistent if all its arcs are arc consistent.

- If an arc $\langle X, Y \rangle$ is *not* arc consistent, all values of X in $\mathbf{D}_X$ for which there is no corresponding value in $\mathbf{D}_Y$ may be deleted from $\mathbf{D}_X$ to make the arc $\langle X, Y \rangle$ consistent.

Example Constraint Network



Arc Consistency Algorithm

The arcs can be considered in turn making each arc consistent.

An arc $\langle X, Y \rangle$ needs to be revisited if the domain of Y is reduced.

Three possible outcomes (when all arcs are arc consistent):

- Each domain is empty $\Longrightarrow$ no solution
- Each domain has a single value $\Longrightarrow$ unique solution
- Otherwise, split a domain & apply arc consistency to each case.

Hill Climbing

Many search spaces are too big for systematic search.

A useful method in practice for some consistency and optimization problems is hill climbing:

- Assume a heuristic value for each assignment of values to all variables.
- Maintain a single node corresponding to an assignment of values to all variables.
- Select a neighbor of the current node that improves the heuristic value to be the next current node.

Selecting Neighbors in Hill Climbing

- When the domains are unordered, the neighbors of a node correspond to choosing another value for one of the variables.

- When the domains are ordered, the neighbors of a node are the adjacent values for one of the dimensions.

- If the domains are continuous, you can use 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 $v_i + \eta \frac{\partial h}{\partial X_i}$.
  Gradient descent: go downhill; v_i beomes $v_i - \eta \frac{\partial h}{\partial X_i}$.

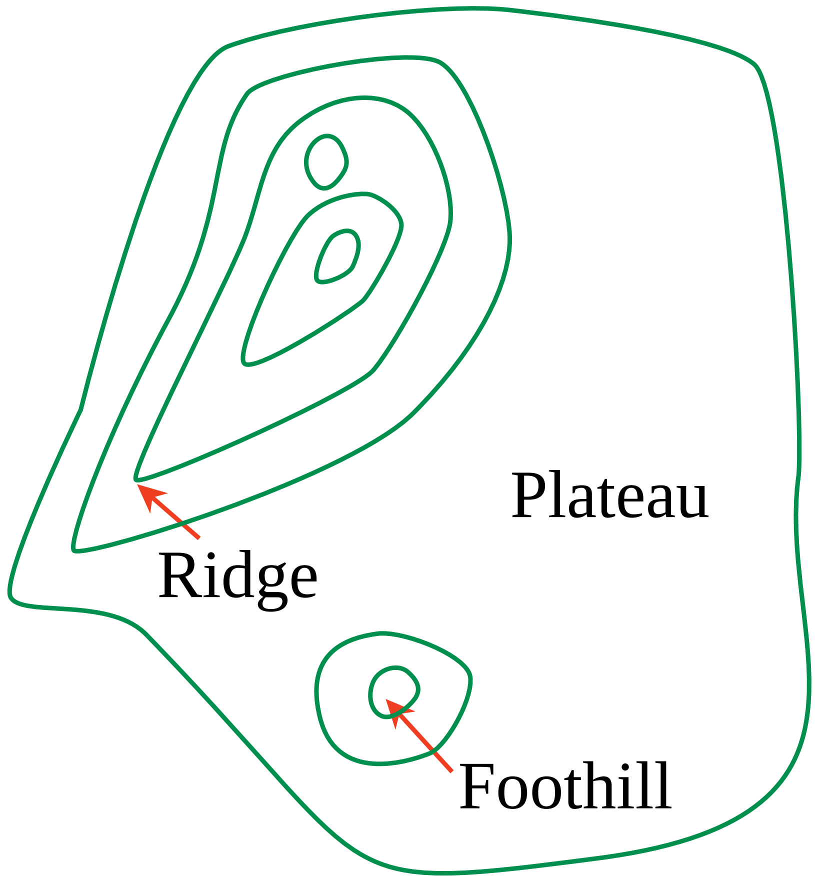
Problems with Hill Climbing

Foothills local maxima
that are not global
maxima

Plateaus heuristic values
are uninformative

Ridge foothill where
n-step lookahead
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

Combinations:

- random-restart hill climbing
- two-phase search: random search, then hill climbing
- maintain multiple nodes, perhaps combine them