CIS 471/571(Fall 2020): Introduction to Artificial Intelligence

Lecture 5: Co [illegible] ction Problems (Part

Thanh H. Nguyen

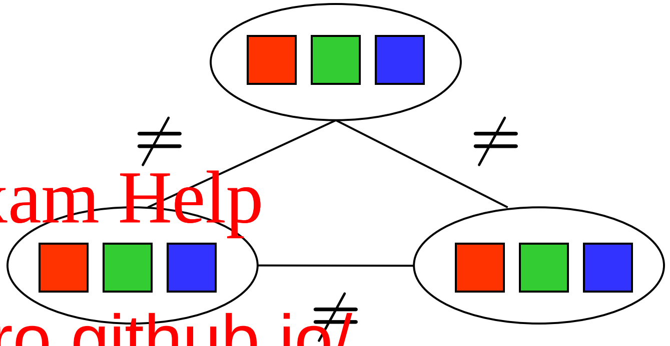
Source: http://ai.berkeley.edu/home.html

Announcements

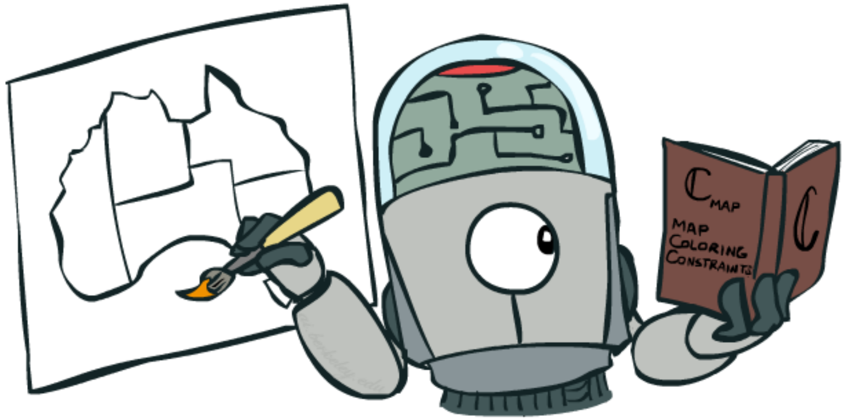
- Project 1:
  - Deadline: Oct 13th, 2020
- Homework 2:
  - Deadline: Oct 24th,
  - Will be posted today

Reminder: CSPs

- CSPs:
  - Variables
  - Domains
  - Constraints
    - Implicit (provide code to co
    - Explicit (provide a list of t tuples)
    - Unary / Binary / N-ary



- Goals:
  - Here: find any solution
  - Also: find all, find best, etc.



Backtracking Search

Improving Backtracking

- General-purpose ideas give huge gains in speed

- Filtering: Can we detect inevitable failure early?
  - Arc consistency
  - Forward checking
  - Constraint propagation

- Ordering:
  - Which variable should be assigned next?
  - In what order should its values be tried?

- Structure: Can we exploit the problem structure?

Example: Map Coloring

Example: Map Coloring

- An arc $X \rightarrow Y$ is consistent iff for *every* x in the tail there is *some* y in the head which could be assigned without violating a constraint
- Enforcing consistency of $X \rightarrow Y$: filter values of the tail X to make $X \rightarrow Y$ consistent
- Forward checking: Enforcing ing to each new assignment

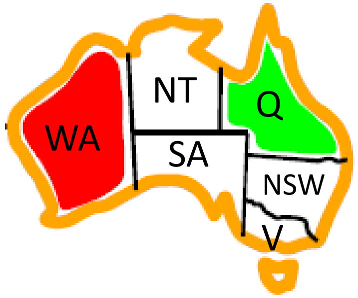
WA NT Q

SA NSW

V

Example: Map Coloring

- Constraint propagation: enforce arc consistency of entire CSP
  - Maintain a queue of arcs to enforce consistency
- Important: If X loses a value, neighbors of X need to be rechecked!
  - After enforcing consiste a value, all arcs pointing to X need to be added back



Ordering

Assignment Project Exam Help

https://eduassistpro.github.io/

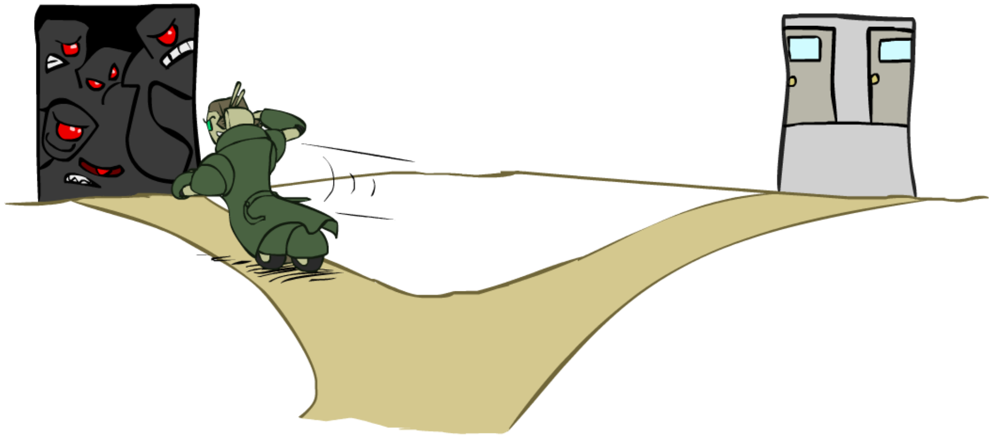
Add WeChat edu_assist_pro

Ordering: Minimum Remaining Values

- Variable Ordering: Minimum remaining values (MRV):
  - Choose the variable with the fewest legal left values in its domain

WA
NT
Q
SA
NSW
V

- Why min rather than max?
- Also called “most constrained variable”
- “Fail-fast” ordering

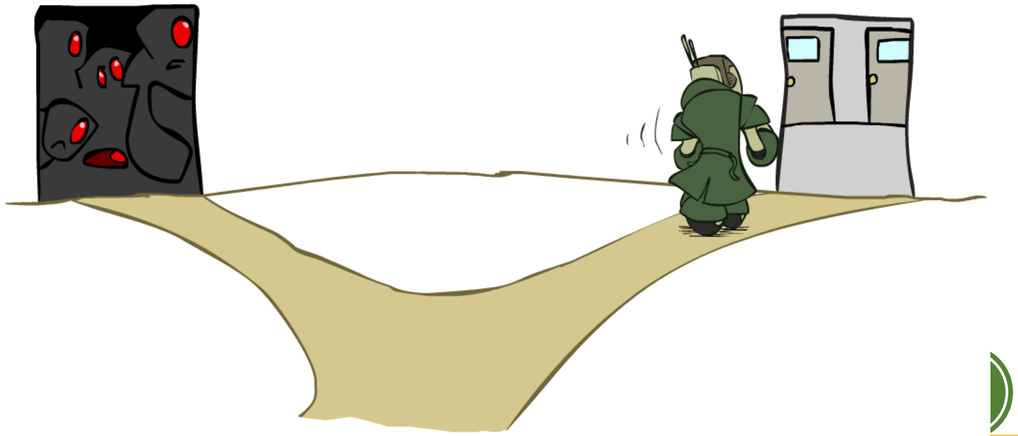


Ordering: Least Constraining Value

- Value Ordering: Least Constraining Value
  - Given a choice of variable, choose the *least constraining value*
  - I.e., the one that rules out the fewest values in the remaining variables
  - Note that it may take som
    determine this! (E.g., reru

- Why least rather than most?

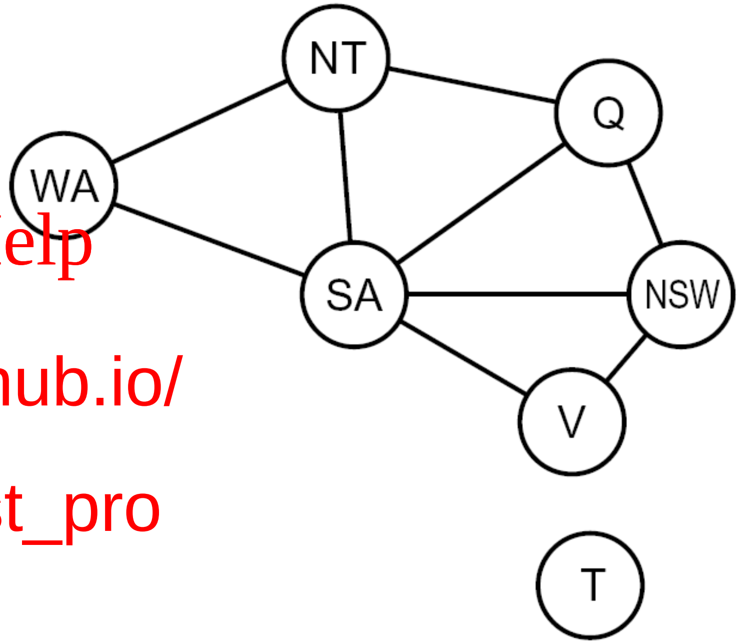
- Combining these ordering ideas makes 1000 queens feasible



Structure

Problem Structure

- Extreme case: independent subproblems
  - Example: Tasmania and mainland do not interact



- Independent subproblems
  connected components of c

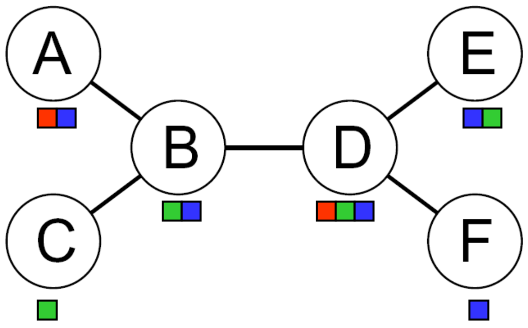
- Suppose a graph of n variables can be broken into subproblems of only c variables:
  - Worst-case solution cost is $O((n/c)(d^c))$, linear in n
  - E.g., n = 80, d = 2, c =20
  - 2^{80} = 4 billion years at 10 million nodes/sec
  - $(4)(2^{20})$ = 0.4 seconds at 10 million nodes/sec

Tree-Structured CSPs

- Theorem: if the constraint graph has no loops, the CSP can be solved in $O(n\, d^2)$ time
  - Compare to general CSPs, where worst-case time is $O(d^n)$

Tree-Structured CSPs

- Algorithm for tree-structured CSPs:
  - Order: Choose a root variable, order variables so that parents precede children



  - Remove backward: For $i = n : 2$, apply RemoveInconsistent(Parent(X_i),X_i)
  - Assign forward: For $i = 1 : n$, assign X_i consistently with Parent(X_i)
- Runtime: $O(n\ d^2)$ (why?)

Tree-Structured CSPs

- Claim 1: After backward pass, all root-to-leaf arcs are consistent
- Proof: Each X→Y was made consistent at one point and Y's domain could not have been reduced thereafter (because Y's children were processed before Y)

- Claim 2: If root-to-leaf arcs are consistent, forward assignment will not backtrack
- Proof: Induction on position

- Why doesn't this algorithm work with cycles in the constraint graph?

- Note: we'll see this basic idea again with Bayes' nets

Improving Structure

Nearly Tree-Structured CSPs

- Conditioning: instantiate a variable, prune its neighbors' domains
- Cutset conditioning: instantiate (in all ways) a set of variables such that the remaining constraint graph is a tree
- Cutset size c gives runtime $O((d^c) (n\text{-}c)\ d^2)$, very fast for small c

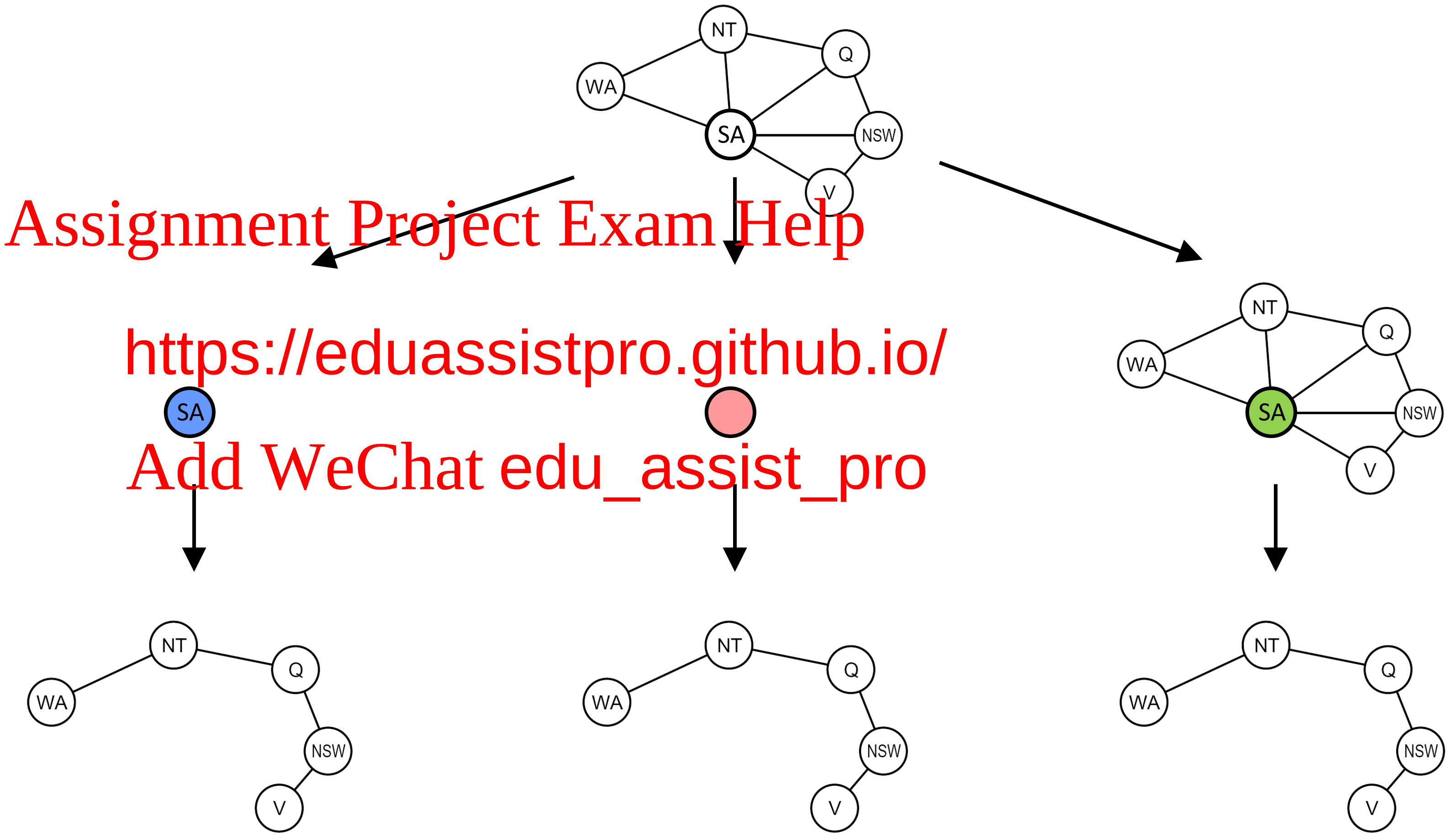
Cutset Conditioning

Choose a cutset

Instantiate the cutset (all possible ways)

Compute residual CSP for each assignment

Solve the residual CSPs (tree structured)

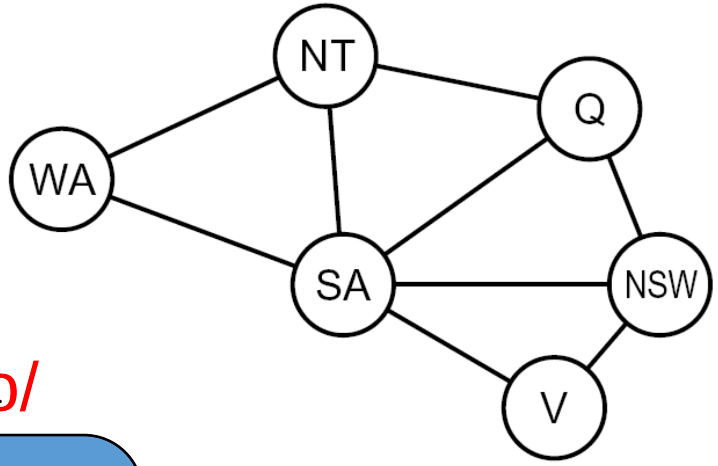


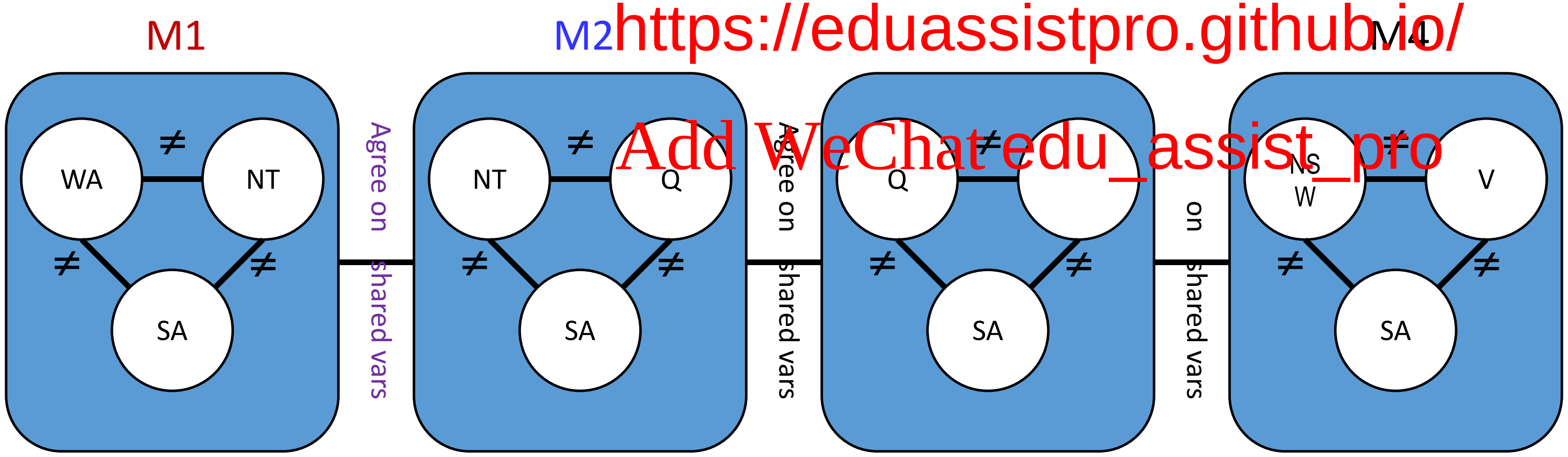
Cutset Quiz

- Find the smallest cutset for the graph below.

Tree Decomposition*

- Idea: create a tree-structured graph of mega-variables
- Each mega-variable encodes part of the original CSP
- Subproblems overlap to ensure consistent solutions





{(WA=r,SA=g,NT=b),
(WA=b,SA=r,NT=g),
…}

{(NT=r,SA=g,Q=b),
(NT=b,SA=g,Q=r),
…}

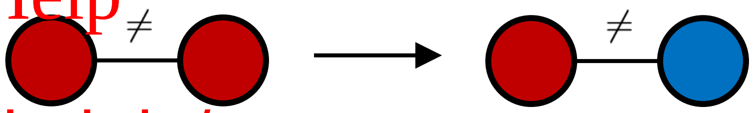
Agree: (M1,M2) ∈
{((WA=g,SA=g,NT=g), (NT=g,SA=g,Q=g)), …}

Iterative Improvement

Iterative Algorithms for CSPs

- Local search methods typically work with "complete" states, i.e., all variables assigned

- To apply to CSPs:
  - Take an assignment with u
  - Operators *reassign* variable
  - No fringe! Live on the edge.



- Algorithm: While not solved,
  - Variable selection: randomly select any conflicted variable
  - Value selection: min-conflicts heuristic:
    - Choose a value that violates the fewest constraints
    - I.e., hill climb with h(n) = total number of violated constraints

Example: 4-Queens

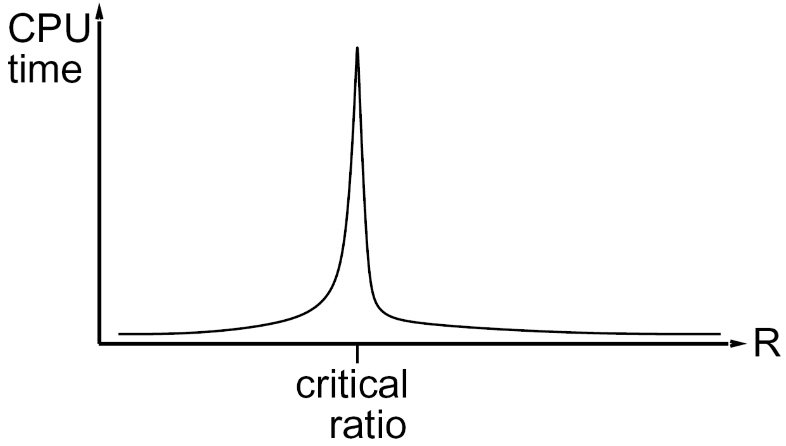
- States: 4 queens in 4 columns ($4^4 = 256$ states)
- Operators: move queen in column
- Goal test: no attacks
- Evaluation: c(n) = number of attacks

Performance of Min-Conflicts

- Given random initial state, can solve n-queens in almost constant time for arbitrary n with high probability (e.g., n = 10,000,000)!

- The same appears to be tr nerated CSP *except* in a narrow range of the ratio

$$R = \frac{\text{number of constraints}}{\text{number of variables}}$$



Summary: CSPs

- CSPs are a special kind of search problem:
  - States are partial assignments
  - Goal test defined by constraints

- Basic solution: backtr

- Speed-ups:
  - Ordering
  - Filtering
  - Structure

- Iterative min-conflicts is often effective in practice

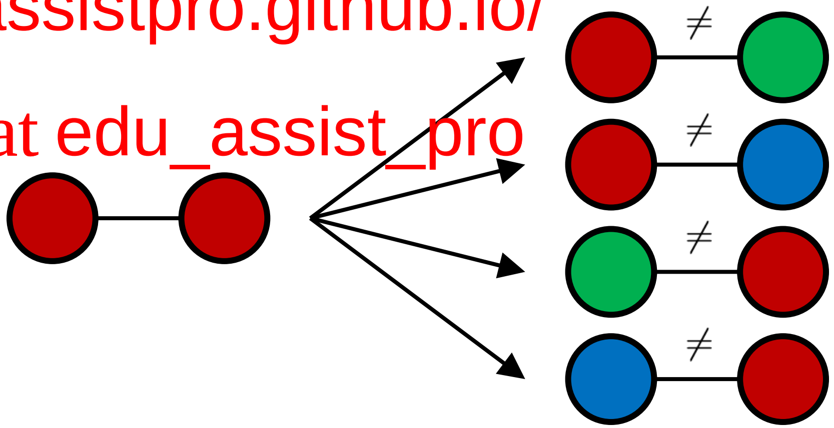
Local Search

Local Search

- Tree search keeps unexplored alternatives on the fringe (ensures completeness)

- Local search: improve a single option until you can't make it better (no fringe!)

- New successor function: loca



- Generally much faster and more memory efficient (but incomplete and suboptimal)

Hill Climbing

- Simple, general idea:
  - Start wherever
  - Repeat: move to the best neighboring state
  - If no neighbors better th

- What's bad about this approach?
  - Complete?
  - Optimal?

- What's good about it?

Hill Climbing Diagram

Hill Climbing Quiz

Starting from X, where do you end up ?

Starting from Y, where do you end up ?

Starting from Z, where do you end up ?

Simulated Annealing

- Idea: Escape local maxima by allowing downhill moves
  - But make them rarer as time goes on