Beyond “Classic” Search
Do you recognize these?
What were the pieces necessary for “classic” search
Single-State Problem Formulation

• A problem is defined by six items:
  1. A set of states
  2. initial state
  3. Set of actions available for each state
  4. successor function (which actually defines all reachable states)
  5. goal test
  6. path cost (additive)
Single-State Problem Formulation

• A problem is defined by six items:
  1. A set of states
  2. initial state
  3. Set of actions available for each state
  4. successor function (which actually defines all reachable states)
  5. goal test
  6. path cost (additive)

But what if you don't really know what the solution/goal will look like?
Suppose your job...

• Is working for a CHEAP company that is trying to rip off "Toy Story"
• As you may know, 3d models are created from putting together a mesh of vertices in space that form a ton of little polygons.
• Your job is to find the 150 triangles that, when put together, look like Buzz Lightyear [again, CHEAP]
Suppose your job…

• How can you SEARCH for the optimal solution to this problem?
Local Search Algorithms

- Hill-climbing (or gradient ascent/descent)
  “Like climbing Everest in thick fog with amnesia”

```plaintext
function HILL-CLIMBING( problem ) returns a state that is a local maximum
inputs: problem, a problem
local variables: current, a node
neighbor, a node

current ← MAKE-NODE( INITIAL-STATE[ problem ])
loop do
    neighbor ← a highest-valued successor of current
    if VALUE[ neighbor ] < VALUE[ current ] then return STATE[ current ]
    current ← neighbor
end
```
Local Search Algorithms

• Hill-climbing
  Problem: depending on initial state, can get stuck on local maxima

In continuous spaces, problems with choosing step size, slow convergence
Other Options

- “Sideways move” – allow a certain number of sideways moves
- “First-choice hill climbing” – pick first generated successor with higher value
- “Random-restart” – hill climbing with multiple restarts with new random starting states
Local Search Algorithms

• Simulated annealing
  Escape local maxima by allowing some “bad” moves but gradually decrease their size and frequency
Local Search Algorithms

\[
\text{function } \text{SIMULATED-ANNEALING}(\text{problem}, \text{schedule}) \text{ returns a solution state }
\]
\[
\text{inputs: } \text{problem}, \text{ a problem }
\]
\[
\text{schedule, a mapping from time to "temperature"}
\]
\[
\text{local variables: } \text{current}, \text{ a node }
\]
\[
\text{next, a node }
\]
\[
T, \text{ a "temperature" controlling prob. of downward steps }
\]
\[
\text{current} \leftarrow \text{MAKE-NODE}((\text{INITIAL-STATE}[\text{problem}])
\]
\[
\text{for } t \leftarrow 1 \text{ to } \infty \text{ do }
\]
\[
T \leftarrow \text{schedule}[t]
\]
\[
\text{if } T = 0 \text{ then return } \text{current}
\]
\[
\text{next} \leftarrow \text{a randomly selected successor of } \text{current}
\]
\[
\Delta E \leftarrow \text{VALUE}[\text{next}] - \text{VALUE}[\text{current}]
\]
\[
\text{if } \Delta E > 0 \text{ then } \text{current} \leftarrow \text{next}
\]
\[
\text{else } \text{current} \leftarrow \text{next only with probability } e^{\Delta E/T}
\]
Generating Pictures

• http://alteredqualia.com/visualization/evolve/