343H: Honors AI

Week 3 – Beyond classical search
Today

- Review of A* and admissibility
- Graph search
- Consistent heuristics
- **Local search**
  - Hill climbing
  - Simulated annealing
  - Genetic algorithms
  - Continuous search spaces
Local Search Methods

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

- Local search: improve what you have until you can’t make it better

- Tradeoff: Generally much faster and more memory efficient (but incomplete)
Types of Search Problems

- **Planning problems:**
  - We want a path to a solution (examples?)
  - Usually want an optimal path
  - *Incremental formulations*

- **Identification problems:**
  - We actually just want to know what the goal is (examples?)
  - Usually want an optimal goal
  - *Complete-state formulations*
  - Iterative improvement algorithms
Hill Climbing

- Simple, general idea:
  - Start wherever
  - Always choose the best neighbor
  - If no neighbors have better scores than current, quit

- Why can this be a terrible idea?
  - Complete?
  - Optimal?

- What’s good about it?
Hill Climbing Diagram

- Sideways steps?
- Random restarts?
“Many real problems have a landscape that looks more like a widely scattered family of balding porcupines on a flat floor, with miniature porcupines living on the tip of each porcupine needle, ad infinitum.” [Russell & Norvig]
Quiz

- Hill climbing on this graph:

![Graph showing hill climbing](image)
Hill climbing Mona Lisa

Could the computer paint a replica of the Mona Lisa using only 50 semi transparent polygons?

Hill climbing Mona Lisa

Could the computer paint a replica of the Mona Lisa using only 50 semi transparent polygons?

Hill climbing Mona Lisa

Could the computer paint a replica of the Mona Lisa using only 50 semi transparent polygons?

Hill climbing Mona Lisa

Could the computer paint a replica of the Mona Lisa using only 50 semi transparent polygons?

Accepting bad moves
Simulated Annealing

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

```plaintext
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
inputs: problem, a problem
         schedule, a mapping from time to "temperature"
local variables: current, a node
                next, a node
                T, a "temperature" controlling prob. of downward steps

current ← MAKE-NODE(INITIAL-STATE[problem])
for t ← 1 to ∞ do
    T ← schedule[t]
    if T = 0 then return current
    next ← a randomly selected successor of current
    ΔE ← VALUE[next] - VALUE[current]
    if ΔE > 0 then current ← next
    else current ← next only with probability e^{ΔE/T}
```
Simulated Annealing

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

function Simulated-Annealing( problem, schedule) returns a solution state
inputs: problem, a problem
        schedule, a mapping from time to “temperature”
local variables: current, a node
                next, a node
                T, a “temperature” controlling prob. of downward steps

current ← Make-Node(Initial-State[problem])
for t ← 1 to ∞ do
    T ← schedule[t]
    if T = 0 then return current
    next ← a randomly selected successor of current
    ΔE ← VALUE[next] - VALUE[current]
    if ΔE > 0 then current ← next
    else current ← next only with probability \( e^{\Delta E/T} \)
Simulated Annealing

- **Theoretical guarantee:**
  - Stationary distribution: \( p(x) \propto e^{\frac{E(x)}{kT}} \)
  - If \( T \) decreased slowly enough, will converge to optimal state!

- Is this an interesting guarantee?

- Sounds like magic, but reality is reality:
  - The more downhill steps you need to escape, the less likely you are to ever make them all in a row
  - People think hard about *ridge operators* which let you jump around the space in better ways
Beam Search

- Like greedy hillclimbing search, but keep K states at all times:

  Greedy Search  Beam Search

- Variables: beam size, encourage diversity?
- The best choice in many practical settings
Genetic Algorithms

- Genetic algorithms use a natural selection metaphor
- Like beam search (selection), but also have pairwise crossover operators, with optional mutation
Example: N-Queens

- Why does crossover make sense here?
- When wouldn’t it make sense?
- What would mutation be?
- What would a good fitness function be?
Exercise 4.1
Continuous Problems

- Placing airports in Romania
  - States: \((x_1, y_1, x_2, y_2, x_3, y_3)\)
  - Cost: sum of squared distances to closest city
Gradient Methods

- How to deal with continuous (therefore infinite) state spaces?
- Discretization: bucket ranges of values
  - E.g. force integral coordinates
- Continuous optimization
  - E.g. gradient ascent

\[ \nabla f = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3} \right) \]

\[ x \leftarrow x + \alpha \nabla f(x) \]
Summary

- Graph search
  - Keep closed set, avoid redundant work
- A* graph search
  - Optimal if h is consistent
- Local search: Improve current state
  - Avoid local min traps (simulated annealing, crossover, beam search)