Emacs Summary

- **M-x**: [Alt]-[x] or [ESC] then [x], **C-x**: [CTRL]-[x]
  - M-x shell (run shell within emacs)
  - C-p (↑), C-n (↓), C-b (←), C-f (→)
  - C-x C-f (load file)
  - M-x lisp-mode (environment for editing lisp code)
  - C-s (search forward) C-r (reverse search)
  - C-q (abort current command in scratch)
  - C-k (cut line) C-y (yank, or paste)
  - C-space (begin block) C-x C-x (end block) C-w (cut) C-y (yank, or paste)
  - C-x u or M-x undo (undo) ; C-x C-s (save) ; C-x C-c (exit)

Full reference card: http://www.cs.tamu.edu/faculty/choe/courses/02spring/refs

Complexity of A*: Revisited

A* is complete and optimal, but space complexity can become exponential if the heuristic is not good enough.

- Condition for **subexponential** growth:

  \[ |h(n) - h^*(n)| \leq O(\log h^*(n)) \]

  where \( h^*(n) \) is the true cost from \( n \) to the goal.

- That is, error in the estimated cost to reach the goal should be less than even linear, i.e. \( < O(h^*(n)) \).

Unfortunately, with most heuristics, error is at least proportional with the true cost, i.e. \( \geq O(h^*(n)) > O(\log h^*(n)) \).

Overview

- Complexity of A*
- **IDA** details
- Hill-climbing revisited
- Hill-climbing strategy
- Simulated annealing details

Linear vs. Logarithmic Growth Error

- Error in heuristic: \( |h(n) - h^*(n)| \).
- For most heuristics, the error is at least linear.
- For A* to have subexponential growth, the error in the heuristic should be on the order of \( O(\log h^*(n)) \).
**Problem with A**

Space complexity is usually exponential!

- we need a memory bounded version
- one solution is: Iterative Deepening A*, or IDA*

---

**IDA**

**function IDA**(problem)

\[
\begin{align*}
\text{root} & \leftarrow \text{Make-Node(Initial-State(problem))} \\
\text{f-limit} & \leftarrow \text{f-Cost(root)} \\
\text{loop do} & \\
\text{solution, f-limit} & \leftarrow \text{DFS-Contour(root, f-limit)} \\
\text{if solution} & \neq \text{NULL then return solution} \\
\text{if f-limit} & = \infty \text{ then return failure} \\
\text{end loop}
\end{align*}
\]

Basically, iterative deepening depth-first-search with depth defined as the \( f\)-cost \((f = g + n)\):

---

**DFS-Contour(root, f-limit)**

Find solution from node \( \text{root} \), within the \( f\)-cost limit of \( \text{f-limit} \). DFS-Contour returns solution sequence and new \( f\)-cost limit.

- if \( f\)-cost(\( \text{root} \)) \( > \) \( \text{f-limit} \), return fail.
- if \( \text{root} \) is a goal node, return solution and new \( f\)-cost limit.
- recursive call on all successors and return solution and minimum \( f\)-limit returned by the calls
- return null solution and new \( f\)-limit by default

Similar to the recursive implementation of DFS.

---

**IDA**: Evaluation

- complete and optimal (with same restrictions as in A*)
- space: proportional to longest path that it explores (because it is depth first!)
- time: dependent on the number of different values \( h(n) \) can assume.
IDA*: Time Complexity

Depends on the heuristics:

- small number of possible heuristic function values → small number of $f$-contours to explore → becomes similar to A*
- complex problems: each $f$-contour only contain one new node
  
  if $A^*$ expands $N$ nodes, $IDA^*$ expands
  
  $$1 + 2 + \ldots + N = \frac{N(N+1)}{2} = O(N^2)$$

- a possible solution is to have a fixed increment $\epsilon$ for the $f$-limit → solution will be suboptimal for at most $\epsilon$ ($\epsilon$-admissible)

Hill-Climbing Strategies

Problems of local maxima, plateau, and ridges:

- try random-restart: move to a random location in the landscape and restart search from there
- keep $n$ best nodes (beam search) *
- parallel search
- simulated annealing *

Hardness of problem depends on the shape of the landscape.

*: coming up next

Hill-Climbing: Revisited

- no queue, keep only the best node
- greedy, no back-tracking
- good for domains where all nodes are solutions:
  - goal is to improve quality of the solution
  - optimization problems

Beam Search

Best-first search with a fixed limited branching factor

- expand the first $n$ nodes with the best Eval-Fn value, where $n$ is a small number.
- $n$ is called the width of the beam
- good for domains with continuous time functions (like speech recognition)
- good for domains with huge branching factor (like above)
Simulated Annealing: Overview

Annealing:

- heating metal to a high-temperature (making it a liquid) and then allowing to cool slowly (into a solid); this relieves internal stresses and results in a more stable, lower-energy state in the solid.

- at high temperature, atoms move actively (large distances with greater randomness), but as temperature is lowered, they become more static.

Simulated annealing is similar:

- basically, hill-climbing with randomness that allows going down as well as the standard up

- randomness (as temperature) is reduced over time

Simulated Annealing (SA)

Goal: minimize the energy $E$, as in statistical thermodynamics. For successors of the current node,

- if $\Delta E \leq 0$, the move is accepted

- if $\Delta E > 0$, the move is accepted with probability $P(\Delta E) = e^{-\frac{\Delta E}{kT}}$, where $k$ is the Boltzmann constant and $T$ is temperature.

- randomness is in the comparison: $P(\Delta E) < \text{rand}(0, 1)$

$\Delta E = E_{\text{new}} - E_{\text{old}}$. The heuristic $h(n)$ or $f(n)$ represents $E$.

$T$ Reduction Schedule

High to low temperature reduction schedule is important:

- reduction too fast: suboptimal solution

- reduction too slow: wasted time

- question: does the form of the reduction schedule curve matter? linear, quadratic, exponential, etc.?

The proper values are usually found experimentally.
Simulated Annealing Applications

- VLSI wire routing and placement
- Various scheduling optimization tasks
- Traffic control
- Neural network training
- etc.

Key Points

- Complexity of $A^*$: relation to error in heuristics
- $IDA^*$ details: evaluation, time and space complexity (worst case)
- Hill-climbing basics and strategies
- Beam search concept
- Simulated annealing details: core algorithm, effect of $T$ and $\Delta E$, source of randomness.

Next Time

- Game playing: Chapter 5
- Minimax
- $\alpha$-$\beta$ pruning
- Probabilistic games