Heuristic Search: A* and beyond

Course: CS40002

Instructor: Dr. Pallab Dasgupta

Department of Computer Science & Engineering

Indian Institute of Technology Kharagpur

Algorithm A*

1. **Initialize:** Set OPEN = {s}, CLOSED = { }, $g(s) = 0$, $f(s) = h(s)$
2. **Fail:** If OPEN = { }, Terminate & fail
3. **Select:** Select the minimum cost state, n, from OPEN. Save n in CLOSED
4. **Terminate:** If $n \in G$, terminate with success, and return $f(n)$

Algorithm A*

5. **Expand:** For each successor, m, of n

   If $m \notin [OPEN \cup CLOSED]$

   Set $g(m) = g(n) + C(n,m)$

   Set $f(m) = g(m) + h(m)$

   Insert m in OPEN

   If $m \in [OPEN \cup CLOSED]$

   Set $g(m) = \min \{ g(m), g(n) + C(n,m) \}$

   Set $f(m) = g(m) + h(m)$

   If f(m) has decreased and $m \in$ CLOSED,

   move m to OPEN

6. **Loop:** Go To Step 2.

Results on A*

A heuristic is called admissible if it always under-estimates, that is, we always have $h(n) \leq f^*(n)$, where $f^*(n)$ denotes the minimum distance to a goal state from state n

- For finite state spaces, A* always terminates

Results on A*

- At any time time before A* terminates, there exists in OPEN a state n that is on an optimal path from s to a goal state, with $f(n) \leq f^*(s)$

- If there is a path from s to a goal state, A* terminates (even when the state space is infinite)

Results on A*

- Algorithm A* is admissible, that is, if there is a path from s to a goal state, A* terminates by finding an optimal path
- If A_1 and A_2 are two versions of A* such that A_2 is more informed than A_1, then A_1 expands at least as many states as does A_2.
    - If we are given two or more admissible heuristics, we can take their max to get a stronger admissible heuristic.

Monotone Heuristics

- An admissible heuristic function, h( ), is monotonic if for every successor m of n:

  $$h(n) - h(m) \leq c(n,m)$$

- If the monotone restriction is satisfied, then A* has already found an optimal path to the state it selects for expansion.

- If the monotone restriction is satisfied, the f-values of the states expanded by A* is non-decreasing.

Pathmax

- Converts a non-monotonic heuristic to a monotonic one:
  - During generation of the successor, m of n we set:

$$h'(m) = \max \{ h(m), h(n) - c(n,m) \}$$

and use h'(m) as the heuristic at m.

Inadmissible heuristics

- Advantages:
  - In many cases, inadmissible heuristics can cause better pruning and significantly reduce the search time
- Drawbacks:
  - A* may terminate with a sub-optimal solution

Iterative Deepening A* (IDA*)

1. Set C = f(s)
2. Perform DFBB with cut-off C

   Expand a state, n, only if its f-value is less than or equal to C

   If a goal is selected for expansion then return C and terminate
3. Update C to the minimum f-value which exceeded C among states which were examined and Go To Step 2.

Iterative Deepening A*: *bounds*

- In the worst case, only one new state is expanded in each iteration
  - If A* expands N states, then IDA* can expand:

$$1 + 2 + 3 + \ldots + N = O(N^2)$$

- IDA* is asymptotically optimal

Memory bounded A*: MA*

- Whenever $|OPEN \cup CLOSED|$ approaches M, some of the least promising states are removed
- To guarantee that the algorithm terminates, we need to back up the cost of the most promising leaf of the subtree being deleted at the root of that subtree
- Many variants of this algorithm have been studied. Recursive Best-First Search (RBFS) is a linear space version of this algorithm

Multi-Objective A*: MOA*

- Adaptation of A* for solving multi-criteria optimization problems
  - Traditional approaches combine the objectives into a single one
  - In multi-objective state space search, the dimensions are retained

- Main concepts:
  - Vector valued state space
  - Vector valued cost and heuristic functions
  - Non-dominated solutions

Iterative Refinement Search

- We iteratively try to improve the solution
  - Consider all states laid out on the surface of a landscape
  - The notion of local and global optima
- Two main approaches
  - Hill climbing / Gradient descent
  - Simulated annealing

Hill Climbing / Gradient Descent

- Makes moves which monotonically improve the quality of solution
- Can settle in a local optima

- Random-restart hill climbing

Simulated Annealing

- Let T denote the temperature. Initially T is high. During iterative refinement, T is gradually reduced to zero.

1. Initialize T
2. If T=0 return current state
3. Set next = a randomly selected succ of current
4. ΔE = Val[next] – Val[current]
5. If $\Delta E > 0$ then Set current = next
6. Otherwise Set current = next with prob $e^{\Delta E/T}$
7. Update T as per schedule and Go To Step 2.