Searching with costs

Course: CS40002
Instructor: Dr. Pallab Dasgupta

Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur

Our first search algorithm

1. **Initialize:** Set OPEN = {s}
2. **Fail:**
   If OPEN = { }, Terminate with failure
3. **Select:** Select a state, n, from OPEN
4. **Terminate:**
   If n $\in$ G, terminate with success
5. **Expand:**
   Generate the successors of n using O and insert them in OPEN
6. **Loop:** Go To Step 2.

Saving the explicit space

1. **Initialize:** Set OPEN = {s}, CLOSED = { }
2. **Fail:** If OPEN = { },
   Terminate with failure
3. **Select:** Select a state, n, from OPEN and
   save n in CLOSED
4. **Terminate:** If n $\in$ G, terminate with success
5. **Expand:**
   Generate the successors of n using O.
   For each successor, m, insert m in OPEN
   only if m $\notin$ [OPEN $\cup$ CLOSED]
6. **Loop:** Go To Step 2.

Search and Optimization

- Given: [S, s, O, G]
- To find:
  - A minimum cost sequence of transitions to a goal state
  - A sequence of transitions to the minimum cost goal
  - A minimum cost sequence of transitions to a min cost goal

Uniform Cost Search

This algorithm assumes that all operators have a cost:

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

Uniform Cost Search

5. Expand:

    Generate the successors of n using O.

    For each successor, m:

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

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

    and insert m in OPEN

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

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

    If C(m) has decreased and

    m $\in$ CLOSED, move it to OPEN

Searching with costs

- **If all operator costs are positive, then the algorithm finds the minimum cost sequence of transitions to a goal.**
  - **No state comes back to OPEN from CLOSED**
- **If operators have unit cost, then this is same as BFS**
- **What happens if negative operator costs are allowed?**

Branch-and-bound

1. **Initialize:** Set OPEN = {s}, CLOSED = { }.
   Set $C(s) = 0$, $C^* = \infty$
2. **Terminate:** If OPEN = { }, then return C^*
3. **Select:** Select a state, n, from OPEN and save in CLOSED
4. **Terminate:** If $n \in G$ and $C(n) < C^*$, then Set $C^* = C(n)$ and Go To Step 2.

Branch-and-bound

5. **Expand:**

    If $C(n) < C^*$ generate the successors of n

    For each successor, m:

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

    Set $C(m) = C(n) + C(n,m)$ and insert m in OPEN

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

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

    If $C(m)$ has decreased and m $\in$ CLOSED, move it to OPEN

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

The notion of heuristics

- Heuristics use domain specific knowledge to estimate the quality or potential of partial solutions

- Examples:
  - **Manhattan distance heuristic for 8 puzzle**
  - **Minimum Spanning Tree heuristic for TSP**
  - **Heuristics are fundamental to chess programs**

The informed search problem

- Given: [S, s, O, G, h] where
  - S is the (implicitly specified) set of states
  - s is the start state
  - O is the set of state transition operators each having some cost
  - G is the set of goal states
  - h( ) is a heuristic function estimating the distance to a goal
- To find:
  - A min cost seq. of transitions to a goal state

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.