Problem Solving by Search

Course: CS40002
Instructor: Dr. Pallab Dasgupta

Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur

Search Frameworks

- State space search
  - Uninformed / Blind search
  - Informed / Heuristic search
- Problem reduction search
- Game tree search
- Advances
  - Memory bounded search
  - Multi-objective search
  - Learning how to search

State space search

- **Basic Search Problem:**
  - Given: [S, s, O, G] where
    - S is the (implicitly specified) set of states
    - s is the start state
    - O is the set of state transition operators
    - G is the set of goal states
  - To find a sequence of state transitions leading from s to a goal state

8-puzzle problem

- **State description (S)**
  - **Location of each of the eight tiles (and the blank)**
- **Start state (s)**
  - **The starting configuration (given)**
- **Operators (O)**
  - **Four operators, for moving the blank left, right, up or down**
- **Goals (G)**
  - **One or more goal configurations (given)**

8-queens problem

Placing 8 queens on a chess board, so that none attacks the other

- **Formulation – I**
  - A state is any arrangement of 0 to 8 queens on board
  - Operators add a queen to any square

8-queens problem

- **Formulation – II**
  - **A state is any arrangement of 0-8 queens with none attacked**
  - **Operators place a queen in the left-most empty column**

8-queens problem

- **Formulation – III**
  - **A state is any arrangement of 8 queens, one in each column**
  - **Operators move an attacked queen to another square in the same column**

Missionaries and cannibals

- **Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to get everyone to the other side, without ever leaving a group of missionaries outnumbered by cannibals**

Missionaries and cannibals

- **State:** (#m, #c, 1/0)
  - #m: number of missionaries in the first bank
  - #c: number of cannibals in the first bank
  - The last bit indicates whether the boat is in the first bank.

- **Start state:** (3, 3, 1) **Goal state:** (0, 0, 0)
- **Operators:**

  Boat carries (1, 0) or (0, 1) or (1, 1) or (2, 0) or (0, 2)

Outline of a 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.

Basics of the search algorithm

- OPEN is a queue (FIFO) vs a stack (LIFO)
- Is this algorithm guaranteed to terminate?
- Under what circumstances will it terminate?

Complexity

- b: branching factor   d: depth of the goal
- Breadth-first search:
  - Time: $1 + b + b^2 + b^3 + \ldots + b^d = O(b^d)$
  - Space: $O(b^d)$
- Depth-first search:
  - Time: $O(b^m)$,

    where m: depth of state space tree
  - Space: $O(bm)$

Tradeoff between space and time

- **Iterative deepening**
  - **Perform DFS repeatedly using increasing depth bounds**
  - **Works in $O(b^d)$ time and $O(bd)$ space**

- **Bi-directional search**
  - **Possible only if the operators are reversible**
  - **Works in $O(b^{d/2})$ time and $O(b^{d/2})$ space**

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.**