Problem Reduction Search:
AND/OR Graphs & Game Trees

Course: CS40002
Instructor: Dr. Pallab Dasgupta

Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur
Problem Reduction Search

- Planning how best to solve a problem that can be recursively decomposed into sub-problems in multiple ways
  - Matrix multiplication problem
  - Tower of Hanoi
  - Blocks World problems
  - Theorem proving
Formulations

- **AND/OR Graphs**
  - An OR node represents a choice between possible decompositions
  - An AND node represents a given decomposition

- **Game Trees**
  - Max nodes represent the choice of my opponent
  - Min nodes represent my choice
The AND/OR graph search problem

- Problem definition:
  - Given: \( [G, s, T] \) where
    - \( G \): implicitly specified AND/OR graph
    - \( S \): start node of the AND/OR graph
    - \( T \): set of terminal nodes
    - \( h(n) \): heuristic function estimating the cost of solving the sub-problem at \( n \)

- To find:
  - A minimum cost solution tree
Algorithm AO*

1. Initialize: Set $G^* = \{s\}$, $f(s) = h(s)$
   If $s \in T$, label $s$ as SOLVED

2. Terminate: If $s$ is SOLVED, then Terminate

3. Select: Select a non-terminal leaf node $n$ from the marked sub-tree

4. Expand: Make explicit the successors of $n$
   For each new successor, $m$:
   Set $f(m) = h(m)$
   If $m$ is terminal, label $m$ SOLVED

5. Cost Revision: Call cost-revise($n$)

Cost Revision in AO*: cost-revise(n)

1. Create $Z = \{n\}$
2. If $Z = \emptyset$ return
3. Select a node $m$ from $Z$ such that $m$ has no descendants in $Z$
4. If $m$ is an AND node with successors $r_1, r_2, \ldots, r_k$:
   
   Set $f(m) = \sum [f(r_i) + c(m, r_i)]$
   
   Mark the edge to each successor of $m$
   
   If each successor is labeled SOLVED, then label $m$ as SOLVED
Cost Revision in AO*: \text{cost-revise}(n)

5. If \( m \) is an OR node with successors \( r_1, r_2, \ldots, r_k \):
   \[
   \text{Set } f(m) = \min \{ f(r_i) + c(m, r_i) \}
   \]
   Mark the edge to the best successor of \( m \)
   If the marked successor is labeled SOLVED, label \( m \) as SOLVED

6. If the cost or label of \( m \) has changed, then insert those parents of \( m \) into \( Z \) for which \( m \) is a marked successor

7. Go to Step 2.
Searching OR Graphs

- How does AO* fare when the graph has only OR nodes?
Searching Game Trees

- Consider an OR tree with two types of OR nodes, namely Min nodes and Max nodes.
- In Min nodes, select the min cost successor.
- In Max nodes, select the max cost successor.
- Terminal nodes are winning or loosing states.
  - It is often infeasible to search up to the terminal nodes.
  - We use heuristic costs to compare non-terminal nodes.
Shallow and Deep Pruning

**Shallow Cut-off**

- Root
  - A
    - B
      - C
- Max node
- Min node

**Deep Cut-off**

- Root
  - F
    - D
      - E
    - G
- Max node
- Min node
Alpha-Beta Pruning

- **Alpha Bound of J:**
  - The max current val of all MAX ancestors of J
  - Exploration of a min node, J, is stopped when its value equals or falls below alpha.
  - In a min node, we update beta

- **Beta Bound of J:**
  - The min current val of all MIN ancestors of J
  - Exploration of a max node, J, is stopped when its value equals or exceeds beta
  - In a max node, we update alpha

- **In both min and max nodes, we return when** $\alpha \geq \beta$
**Alpha-Beta Procedure: \( V(J; \alpha, \beta) \)**

1. If \( J \) is a terminal, return \( V(J) = h(J) \).
2. If \( J \) is a max node:
   - For each successor \( J_k \) of \( J \) in succession:
     - Set \( \alpha = \max \{ \alpha, V(J_k; \alpha, \beta) \} \)
     - If \( \alpha \geq \beta \) then return \( \beta \), else continue
   - Return \( \alpha \)
3. If \( J \) is a min node:
   - For each successor \( J_k \) of \( J \) in succession:
     - Set \( \beta = \min \{ \beta, V(J_k; \alpha, \beta) \} \)
     - If \( \alpha \geq \beta \) then return \( \alpha \), else continue
   - Return \( \beta \)