PROBLEM SOLVING METHODS

Problem Reduction:

Search strategies for OR graph is used to find a single path to a goal. Structures represent how to get from a node to a goal state if we can discover how to get from that node to a goal state along any one of the branches leaving it.

PROBLEM GRAPH

AND-OR-GRAPH :

- The structure, the AND-OR graph (or) tree is used to solve problem by decomposing them into set of smaller problems.
- This decomposition or reduction generates arcs that are called as AND arcs .
- AND arcs point to any number of successor nodes, which must be solved in order for the arc to point to a solution.
- All nodes emerging from the AND arcs must be solved.
- As in an OR graph, several arcs emerge from a single node , indicating a variety of ways the problem will be solved. OR arc means, nodes emerging from the OR arc are optional, any one of the node is to be chosen. This is the reason that this structure is called AND-OR graph.

- The example of AND -OR graph is shown below ,AND arcs are indicated with a line connecting all the components,

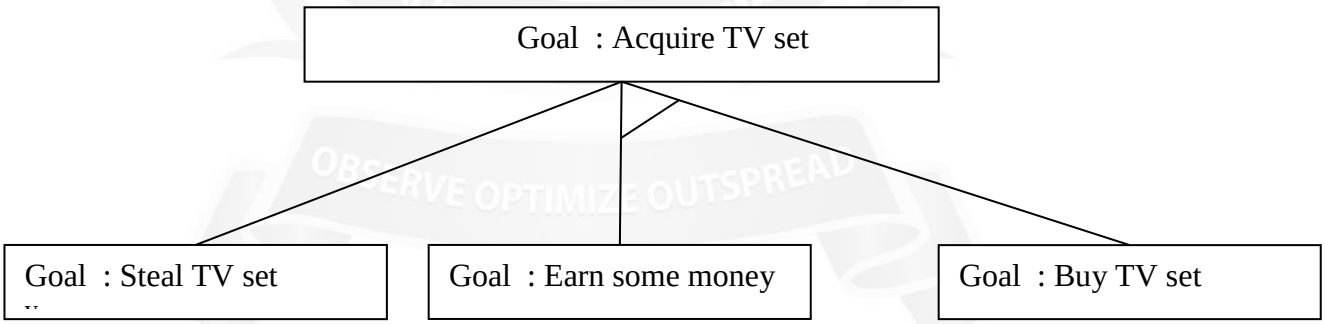


Fig 1.11 A simple AND –OR graph

- To find solution for AND-OR graph , the algorithm , with the ability to handle the AND arcs appropriately.
- The algorithm find a path from the starting node of the graph to a set of nodes representing solution states.

- It is necessary to get to more than one solution state since each arm of an AND arc must lead to its own solution node.
- From the above fig, the top node A, has been expanded, producing two arcs, one leading to B and one leading to C and D.
- The numbers at each node represent the value of f at that node.

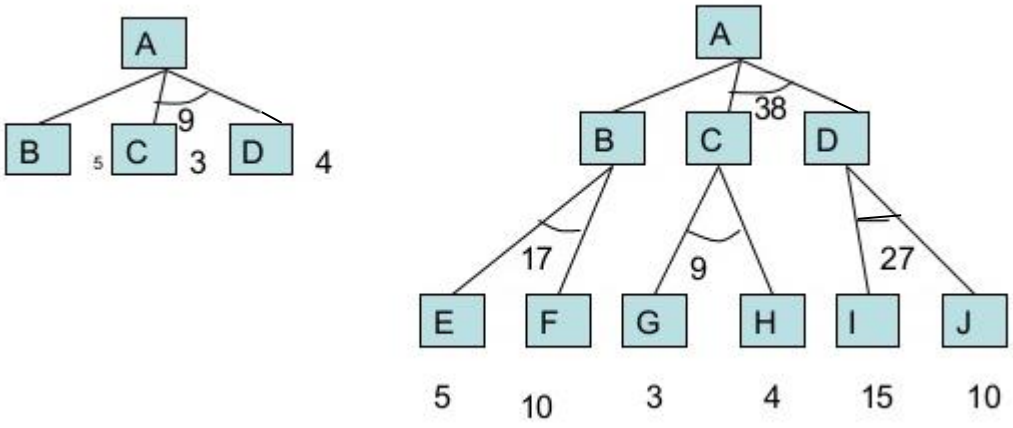


Fig 1.11 a) AND - OR Graphs

- Every operation has a uniform cost, so each arc with a single successor has a cost of 1 and each AND arc with multiple successors has a cost of 1 for each of its components.
- The node with the lowest f' value, must select C.
- By using the information , we should explore the path going through B since to use C we must also use D, for a total cost of 9 (C+D+2) compared to the cost of 6 that we get by going through B.
- The problem is that the choice of which node to expand next must depend not only on the f' value of that node. But also whether that node is part of the current best path from the initial node.
- The tree shown above makes this even clearer.
- The promising single node is G with an f' value of 3
- The path G-H with a total cost of 9
- Arc is not part of the current best path since to use it we must use the arc I-J ,with a cost of 27.
- The path from A, through B, to E and F is better, with a total cost of 18.
- So, should not expand G next, examine either E or F.
- An algorithm for searching an AND-OR graph ,need to exploit a value that we call FUTILITY.

- If the estimated cost of a solution becomes greater than the value of FUTILITY, then we stop the search.
- FUTILITY should be chosen to correspond to a thresholds such that any solution.

Algorithm : Problem Reduction

i) Initialize the graph to the starting node.

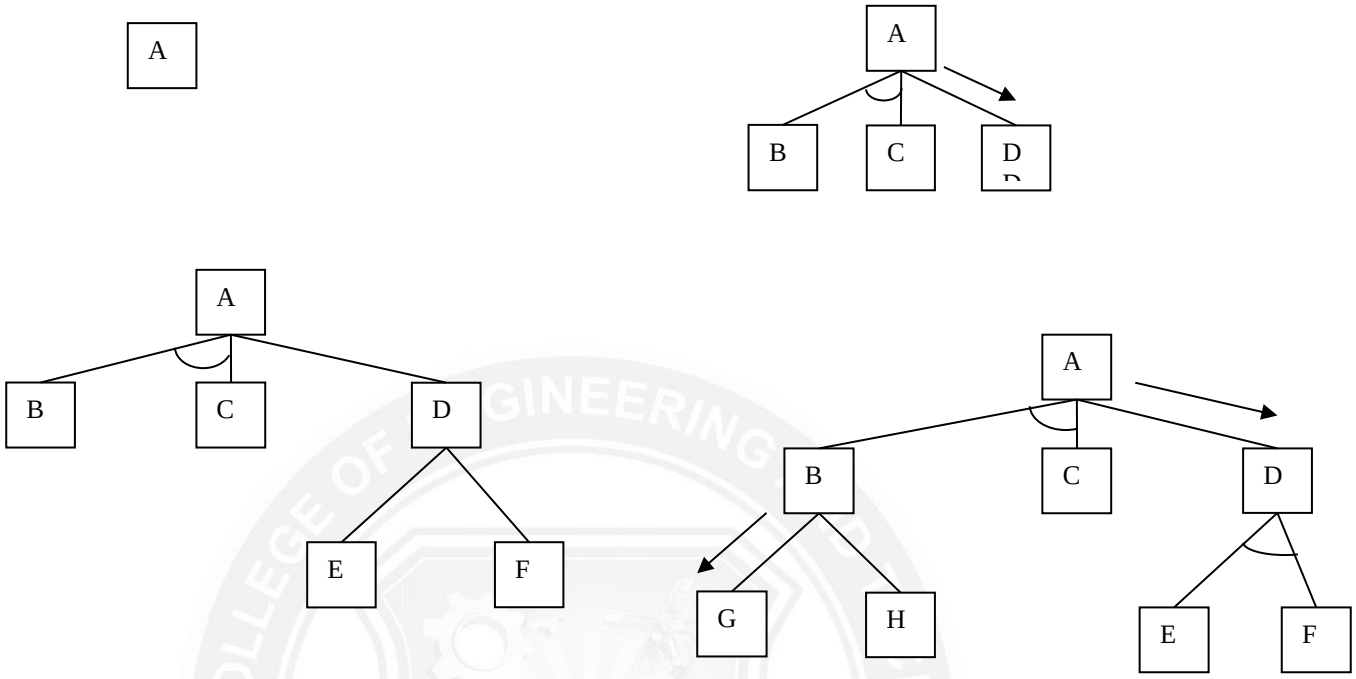
ii) Loop until the starting node is labeled SOLVED or until its cost goes above FUTILITY.

a) Traverse the graph starting at the initial node and following the current best path, and accumulate the set of nodes that are on the path and have not yet been expanded or labeled as solved.

b) Pick one of these unexpanded nodes and expand it. If there are no successors assign FUTILITY as the value of this node. Otherwise, add its successors to the graph and for each of them computer f '. if f' of any node is 0, mark that node as SOLVED.

c) Change the f' estimate of the newly expanded node to reflect the new information provided by its successors. Propagate his change backward through the graph. If any node contains successor arc whose descendants are all solved, label the node itself as SOLVED. At each node that is visited while going up the graph, decide which of its successor arcs is the most promising and mark it as part of the current best path.

This process is illustrated in the below fig.



Steps :

Step 1:

A is the only one, so it is at the end of the current best path. It is expanded yielding nodes B,C and D. The arc to D is labeled as the most promising one emerging from A. It costs 6 compared to B and C which costs 9.

Steps 2 :

Node D is chosen for expansion .This process produces are new arc, the AND arc to E and F, with a combined cost estimate of 10.So we update the f' value of D to 10. Back one level, this makes AND and B-C better than the arc to D. So it is labeled as the current best path.

Step 3 :

Traverse that arc from A and discover the unexpanded nodes B and c. To find a solution along this path , have to expand both B and C eventually. So B is chosen to explore first. This generates two new arcs, the ones to G and to H. Propagating their f'values backward ,we update f'of B to 6. It requires updating the cost of the AND arc B-C to 12 (6+4+2). The arc to D is the better path from A, we record that as the current best path and either node E or node F will be chosen for expansion in next step.

Step 4 :

This process continues until either a solution is found or all paths have led to deal ends, indicating that there is no solution.

Alternate way :

Algorithm for searching an AND-OR graph must differ from one for searching an OR graph. The different arises from the fact that individual paths from node to node cannot be considered independently of the paths through other nodes connected to the original ones by AND arcs. In best –first search algorithm, the path from one node to another was always the one with the lower cost. But not always when searching an AND-OR graph.

Consider the example shown in below fig.

The nodes generated in alphabetical order. If node J is expanded at the next step and one of its successors is node E, producing the graph shown in fig below.

This new path to E is longer than the previous path to E going through c. But the path through C will lead to a solution if there is also a solution to D, but the path J is better.

Limitation of Algorithm :

- It fails to take into account any interaction between sub goals.
- Example of failure is shown in below fog.
- Assume both node C and node E ultimately lead to a solution. Our algorithm will report a complete solution that includes both of them.
- The AND-OR graph states that for A to be solved, both C and D must be solved.
- Algorithm considers a solution of D as a completely separate process from the solution of C.
- The alternatives from D, E is the best path but it turns out that C is necessary , so its better to satisfy D.