Solving Problem By Searching

<@mit>
Jn-14

2012 : Final
+
2013 : I.C

Q) Explain why problem formulation must follow goal formulation. [4]

Ans:

Goal formulation:

- Goal formulation is the first step in problem solving.
- Goal formulation is based on the current situation & the agent's performance measure.
- Goals help organize behavior by limiting the objectives that the agent is trying to achieve.

Problem formulation:

- Problem formulation is the immediate next step after goal formulation.
- Problem formulation is the process of deciding what actions & states to consider, given a goal.

mple:

Consider an agent in the city of Dhaka, Bangladesh, enjoying a touring holiday. The agent's performance measure contains many factors: it wants to improve its Bengali, take in the sights, enjoy the foods, avoid hang-overs, & so on. The decision problem is a complex one involving many tradeoffs & careful reading of guidebooks. Now let's suppose that the agent has a nonrefundable ticket to fly out to Chittagong the following day.

In that case, it makes sense for the agent to adopt the goal of getting to Chittagong. Courses of action that don't reach Chittagong on time can be rejected without further consideration & the agent's decision problem is greatly simplified. This goal formulation has been done by the agent intuitively as the 1st step.

Report:

1. Complete the data table.
2. Calculate I for different values of E and R.
3. Calculate E from the data.
4. Calculate the % of error for each data.
5. Calculate the total power delivered by the source.

Here we can consider a goal to be a set of divisions - exactly those divisions in which the goal is satisfied. The agent's task is to find out which sequence of actions will get it to a goal division. Before it can do this, it needs to decide what sorts of actions & states to consider. Hence problem formulation has been done by the agent as step two.

From the above example we can see that in goal formation, we decide which aspects of the world we are interested in, & which can be ignored right away. Then in problem formulation we decide how to manipulate the important aspects & ignore the others.

If we did problem formulation first we would not know what to include & what to leave out. But it can happen that there is a cycle of iterations between goal formulation, problem formulation & problem solving until one arrives at a sufficiently useful & efficient solution.

2012

⑨ You have a program that output "illegal input record" ~~found~~ when fed a certain file of input record. Here processing of each record is independent of the other record. You want to get what record is illegal. Give initial state, goal ~~state~~, test, successor function & cost function for the above program. [4]

Ans:

- Initial state: Considering all input records.

- Goal test: Considering a single record, & it gives "illegal input" message.

- Successor function: Run again on the first half of the records; run again on the second half of the records.

- Cost function: Number of runs.

2012

(9) Discuss the complexities of different blind search algorithms - BFS, DFS, Uniform Cost Search, DLS & IDS.

Ans:

BFS:

BFS is a simple strategy in which the root node is expanded first, then all the successors of the root node are expanded next, then their successors, & so on. In general, all the nodes are expanded at a given depth in the search tree before any nodes at the next level are expanded.

· Completeness: BFS is complete. If the shallowest goal node is at some finite depth d, BFS will eventually find it after expanding all shallower nodes (provided the branching factor b is finite).

· Optimality: If ~~the~~ path cost is a nondecreasing function of the depth of the node, for example, when all actions have the same cost, BFS is optimal.

• **Time Complexity:**

Let's consider a hypothetical state space where every state has b successors.

The search tree generates –

b nodes at the 1st level

b^2 " " " 2nd "

b^3 " " " 3rd " & so on.

Now suppose that the solution is at depth d. In the worst case, we would expand all but the last node at level d (since the goal itself isn't expanded), generating $(b^{d+1} - b)$ nodes at level d+1. Then the total number of nodes generated is –

$$N(BFS) = b + b^2 + b^3 + \ldots\ldots + b^d + (b^{d+1} - b) = O(b^{d+1})$$

• **Space Complexity:**

Every node that is generated must remain in memory, because it is either part of the fringe or is an ancestor of a fringe node. The space complexity is, therefore, the same as the time complexity i.e. $O(b^{d+1})$.

DFS:

- DFS always expands the deepest node in the current fringe of the search tree. The search proceeds immediately to the deepest level of the search tree, where the nodes have no successors. As those nodes are expanded, they are dropped from the fringe, so then the search "backs up" to the next shallowest node that still has unexplored successors.

- **Completeness:** If the subtree which is being explored is of unbounded depth but contained no solutions, DFS would never terminate, hence it's not ~~opt~~ complete.

- **Optimality:** DFS can make a wrong choice & get stuck going down a very long (or even infinite) path when a different choice would lead to a solution near the root of the search tree. Hence DFS isn't optimal.

Time Complexity:

In the worst case, depth-first search will generate all of the nodes in the search tree.

Let, b = branching factor
m = maximum depth of any node
[m can be much larger than d (the depth of the shallowest solution), & is infinite if the tree is unbounded]

$\therefore$ Time complexity $= O(b^m)$

Space Complexity:

DFS has very modest memory requirements. It needs to store only a single path from the root to a leaf node, along with the remaining unexpanded sibling nodes for each node on the path. Once a node has been expanded, it can be removed from memory as soon as all its descendants have been fully explored.

$\therefore$ Space complexity $= O(bm)$

Uniform Cost Search:

- Uniform cost search is a modified version of BFS which is optimal with any step cost function.
- Instead of expanding the shallowest node, uniform cost search expands the node n with the lowest path cost.

Completeness: Completeness is guaranteed provided the cost of every step is greater than or equal to some small positive constant ϵ.

- **Optimality:** The algorithm expands nodes in order of increasing path cost. Therefore, the first goal node selected for expansion is the optimal solution.

- **Time & space complexity:**

Let, C^* = Cost of the optimal solution
ϵ = least cost of every action
b = branching factor

$\therefore$ Time complexity $= O(b^{[C^*/\epsilon]})$

& Space " $= O(b^{[C^*/\epsilon]})$.

Depth-limited Search (DLS):

- DLS is a modified version of DFS with a pre-determined depth limit l used to alleviate the problem of unbounded trees.
- Nodes at depth l are treated as if they have no successors.

Completeness:

If the goal node is at level d & we choose $l<d$ (shallowest goal is beyond the depth limit), DLS becomes incomplete.

Optimality:

DLS becomes non-optimal if we choose $l>d$.

Time complexity:

Let, b = branching factor
l = pre-determined depth limit

$\therefore$ Time complexity $= O(b^l)$

Space complexity:

Space complexity of DLS $= O(bl)$

Iterative Deepening Search (IDS):

- IDS is a general strategy, often used in combination with DFS, that finds the best depth limit.
- IDS combines the benefits of DFS & BFS.
- It's the preferred search method when there is a large search space & depth of the solution isn't known.
- **Completeness:**

Like BFS, IDS is complete when the branching factor is finite.

- **Optimality:**

Similar to BFS, IDS is optimal when the path cost is a non-decreasing function of the depth of the node.

- **Time complexity:**

In IDS, the nodes on the bottom level (depth d) are generated once, those on the next to bottom level are generated twice, & so on, upto the children of the root, which are generated d times

$$N(IDS) = (d)b + (d-1)b^2 + \dots + (1)b^d = O(b^d)$$

- **Space complexity:**

Like ~~BFS~~ DFS, it's memory requirement is very modest, $O(bd)$ to be precise

space complexity $= O(bd)$.

Summary:

To summarize, let's have a final look –

Criterion	BFS	Uniform Cost search	DFS	DLS	IDS
Complete?	Yes ~~(if b is finite)~~ ~~(if path cost = 1)~~	Yes	No	No	Yes
Optimal?	Yes	Yes	No	No	Yes
Time	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b^m)$	$O(b^l)$	$O(b^d)$
Space	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(bm)$	$O(bl)$	$O(bd)$