5.7 HIERARCHICAL PLANNING

Q.1. Explain the real time application of hierarchical planning.

(MU – Q.3 (b), Dec. 19, 10 Marks, Q.5 (c), May 19, 5 Marks, Q.5 (a), Dec. 18, 5 Marks, Q.6 (c), May 17, 2 Marks)

Q.1. Explain Planning in AI. Compare Partial Order Planning with Conditional Planning. Also, explain the real time application of hierarchical planning. **(MU – Q.3 (b), Dec. 19, 10 Marks)**

1. The idea of **Hierarchical Problem-Solving** is to distinguish between goals and actions of different degrees of importance, and solve the most important problems first. It's main advantage derives from the fact that by emphasizing certain activities while temporarily ignoring others, it is possible to obtain a much smaller search space in which to find a plan.
2. As an example, suppose that in the household domain we would like to paint the ceiling white. Initially the number of conditions to consider may be overwhelming, ranging from the availability of various supplies, the suppliers for equipments and tools, to the position of the agent, the ladder, and the state of the ceiling. However, we could obtain a more manageable search space by first concentrating on whether we have the paint, the ladder, and a brush. Once a plan is found we then consider how to refine this plan by considering how to get to the rooms where each item is located. The process repeats until a full-blown plan is finally found.
3. Today, a large number of problem-solving systems have been implemented and studied based on the concept of hierarchical planning. They include **GPS, ABSTRIPS, LAWLY, NOAH, and ABTWEAK**.
4. There are two ways in which details are inserted in a hierarchical plan.
5. The first, precondition-elimination abstraction mimics the human intuition of exploring and solving sub-goals in order of importance, with the most important goals solved first.
6. A second method of hierarchical planning that we consider is called **hierarchical task-network planning (HTN),** where a planning problem and operators are organized into a set of tasks. A high-level task can be reduced to a set of ordered lower-level tasks, and a task can be reduced in several ways. These two methods are complementary-each can be very effective in solving a different type of problem.

5.7.1 A Hierarchical Planner - Algorithm

The intuition behind the operation of a hierarchical planner is shown in Fig. 5.8. In this Figure there are three levels of abstraction, an abstract level, an intermediate level and a concrete level. Each dashed box represents a problem-solver at a given level. A planning problem is first abstracted and solved at the most abstract level. The solution obtained at this level, an abstract plan, is taken as the input to a problem-solver at the next level. The process ends when a concrete-level solution is found.

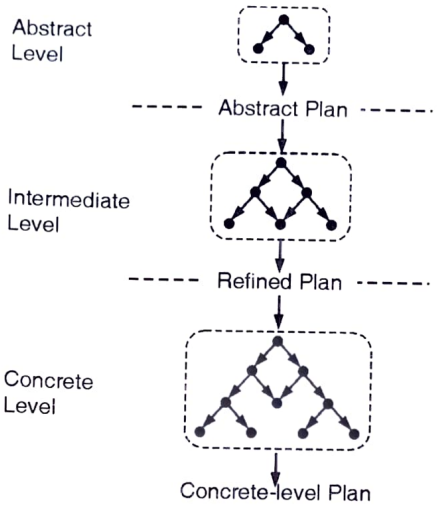

Fig. 5.8: Hierarchical Planning

In general, the abstraction levels could range from a single level to multiple levels. The former is identical to problem-solving without any abstraction. A top-level description of a hierarchical partial-order planner is described in algorithm as shown.

Algorithm:

Algorithm HIPLAN (π0)

Input: An initial abstract plan π_0

External Function: Refine (π, i) returns a set of refinements of plan at abstraction level i,

Output: A correct plan consisting of concrete-level plan steps; or Fail.

```
1.  open list := {π0}
2.  Repeat
3.  π := Remove(OpenList);
4.  if Level (π) = 0 then
5.      if Correct (π) then
6.          return (π);
7.      end if
8.  else
9.      Level (π):= Level (π) - 1;
10.     Successors := Refine (π, Level (π));
11.     OpenList : = Insert(Successors, Open List);
12  end if
13. until OpenList = θ;
14  return(Fail);
```

5.7.2 Problem: Given a Chair and a Table, the Goal is to have them Match — have the Same Color

1. In the initial state we have two cans of paint, but the colours of the paint and the furniture are unknown. Only the table is initially in the agent's field of view:

   Init(Object(Table) ∧ Object(Chair) ∧ Can(C1) ∧ Can(C2) ∧ InView (Table)) Goal (Color(Chair, c) ∧ Color (Table, c))

2. There are two actions: removing the lid from paint can and painting an object using the paint from an open can.

3. The action schemas are straightforward, with one exception: we now allow preconditions and effects to contain variables that are not part of the action's variable list. That is, **Paint(x, can)** does not mention the variable c, representing the colour of the paint in the can. In the fully observable case, this is not allowed — we would have to name the action **Paint(x, can, c).**

4. But in the partially observable case, we might or might not know what color is in the can. (The variable c is universally quantified, just like all the other variables in an action schema.)

   Action(RemoveLid (can),

   PRECOND:Can(can)

   EFFECT:Open(can))

   Action(Paint(x , can),

   PRECOND:Object(x) ∧ Can(can) ∧ Color (can, c) ∧ Open(can)

   EFFECT:Color (x, c))

5. To solve a partially observable problem, the agent will have to reason about the percepts it will obtain when it is executing the plan. The percept will be supplied by the agent's sensors when it is actually acting, but when it is planning it will need a model of its sensors.

5.7.3 A Tower of Hanoi Example

Tower of Hanoi is a mathematical puzzle where we have three rods and n disks. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
3. No disk may be placed on top of a smaller disk.

   Take an example for 2 disks:

   Let rod 1 = 'A', rod 2 = 'B', rod 3 = 'C'.

   Step 1: Shift first disk from 'A' to 'B'.

   Step 2: Shift second disk from 'A' to 'C'.

   Step 3: Shift first disk from 'B' to 'C'.

   The pattern here is:

   Shift 'n−1' disks from 'A' to 'B'.

   Shift last disk from 'A' to 'C'.

   Shift 'n−1' disks from 'B' to 'C'.

Image illustration for 3 disks is shown in Fig. 5.9:

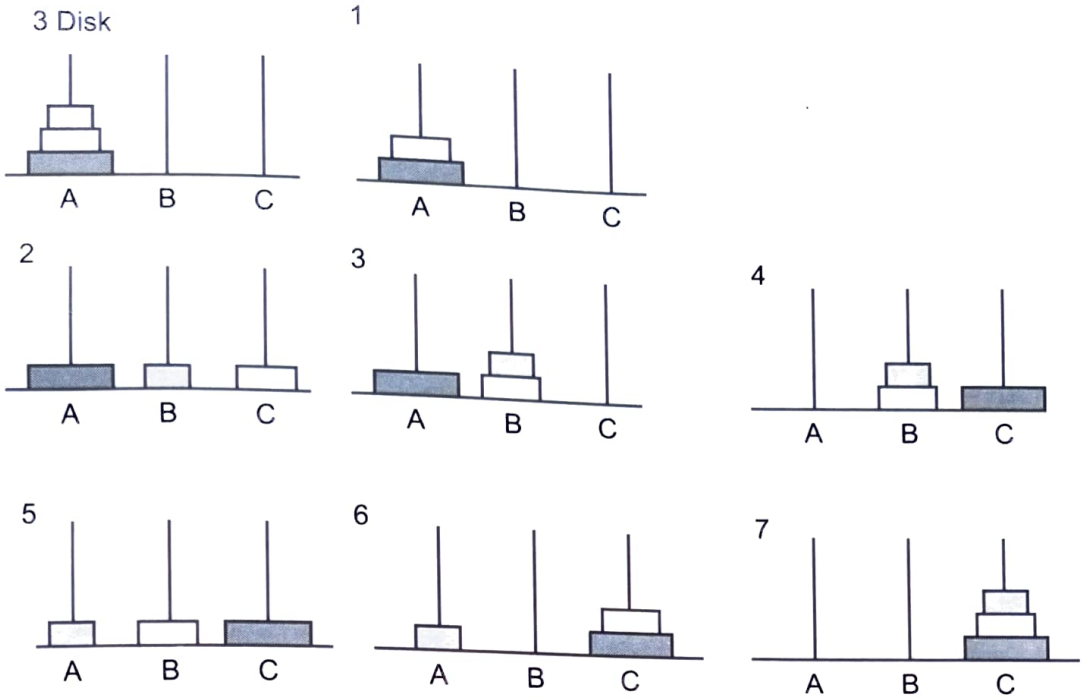

Fig. 5.9: Tower of Hanoi

Input: 2

Output: Disk 1 moved from A to B

Disk 2 moved from A to C

Disk 1 moved from B to C

Input: 3

Output: Disk 1 moved from A to C

Disk 2 moved from A to B

Disk 1 moved from C to B

Disk 3 moved from A to C

Disk 1 moved from B to A

Disk 2 moved from B to C

Disk 1 moved from A to C