PLANNING:

Planning in Artificial Intelligence can be defined as a problem that requires decision-making by intelligent systems to achieve a given target. This intelligent system can take the form of a robot or a computer program.

- For instance, consider a driver who needs to pick up and drop people from one place to another. If the driver has to pick up two people from two different places, a specific sequence of actions must be followed; both passengers cannot be picked up simultaneously.
- Another definition of planning characterizes it as an activity where an agent must determine a sequence of actions to achieve a particular target.
- When formulating a planning problem, information about the initial status of the agent, the goal conditions of the agent, and the set of actions the agent can take is available. The agent's objective is to find the proper sequence of actions that will lead from the starting state to the goal state, resulting in an efficient solution.

Simple Planning Agent

Consider an agent, such as a coffee maker, a printer, or a mailing system, with three users. If all three users simultaneously give commands to execute different tasks (e.g., making coffee, printing, and sending an email), the agent must decide the sequence of these actions.

Planning Problem.

the potential effects of agents actions and how they will impact future actions. This underscores the importance of an agent's ability to provide proper reasoning about its future actions and the states of surrounding environments.

Consider a simple game like Tic-Tac-Toe. Winning the game requires a sequence of actions, with each move influenced by past steps and the anticipation of future actions by the opponent. A player must consider both the opponent's probable future actions and the consequences of their own moves.

Classical planning makes certain assumptions about the task environment:

Fully Observable: The agent can observe the current state of the environment.

Deterministic: The agent can determine the consequences of its actions.

Finite: There is a finite set of actions that the agent can carry out at every state to achieve the goal.

Static: Events are steady, and external events that the agent cannot handle are not considered.

Discrete: Events of the agent are distinct from the starting step to the ending (goal) state in terms of time.

A planning problem essentially finds the sequence of actions to accomplish the goal based on these assumptions. Goals can also be specified as a union of sub-goals.Take, for example, a ping pong game where points are assigned to the opponent player when a player fails to return the ball within the rules. Winning a match may require winning a certain number of games, each with a minimum margin of 2 points.

How planning problem differs from search problem?

- The main difference between problem-solving and planning lies in the openness of the process and the use of logic-based representation in planning.
- Planning is considered more powerful than problem-solving due to these two key reasons.A planning agent involves situations (states), goals (target end conditions), and operations (actions performed).
- These parameters are decomposed into sets of sentences and further into sets of words based on the system's requirements.
- Planning agents exhibit more efficient handling of situations/states due to their explicit reasoning capability, and they can interact with the world. Agents can reflect on their targets, and the complexity of the planning problem can be minimized by independently planning for sub-goals of an agent.
- Agents possess information about past and present actions, and importantly, they can predict the effects of actions by inspecting the operations.
- In essence, planning is a logical representation, grounded in situations, goals, and operations, of the broader concept of problem-solving. Therefore, we can express planning as the combination of problem-solving and logical representation:
- Planning=Problem solving+Logical representation

Planning Graphs

- A planning graph is a specialized data structure employed to achieve better accuracy in the planning process.
- It is a directed graph that proves beneficial for obtaining improved heuristic estimates. Various search techniques can leverage planning graphs, and GRAPHPLAN is one such technique capable of extracting a solution directly.
- Planning graphs are designed for propositional problems without variables, similar to how Gantt charts are utilized.
- In planning graphs, there are a series of levels that correspond to a timeline in the plan.
- Each level comprises a set of literals and a set of actions. Level 0 represents the initial state of the planning graph.

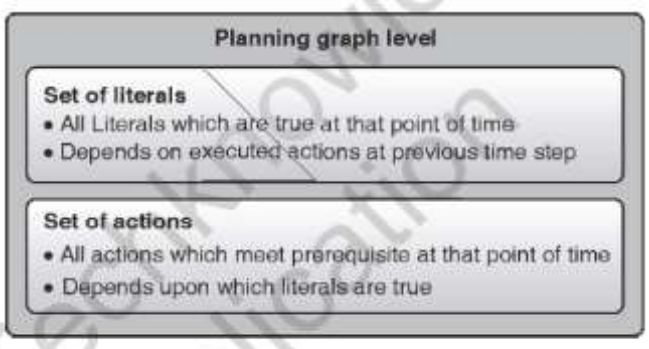


Fig. 3.15.1 : Planning graph level

initial State:

Coffee: None

Coffee Maker: Off

Water: Reservoir Full

Goal State:

Coffee: Present

Actions:

Turn On Coffee Maker

Grind Coffee Beans

Brew Coffee

Level 0 (InitialState):

Coffee: None

Coffee Maker: Off

Water: Reservoir Full

Level 1 (Possible Actions):

Turn On Coffee Maker

Grind Coffee Beans

Brew Coffee

Level 2 (Possible States after Level 1 Actions):

Coffee: None or Ground

Coffee Maker: On or Off

Water: Reservoir Full or Empty

Level 3 (Possible States after Level 2 Actions):

Coffee: None or Ground or Brewed

Coffee Maker: On or Off

Water: Reservoir Full or Empty

Level 4 (Possible States after Level 3 Actions):

Coffee: None or Ground or Brewed or Present

Coffee Maker: On or Off

Water: Reservoir Full or Empty

The planning graph starts from the initial state and expands to show possible states and actions at each level. In this example, turning on the coffee maker, grinding coffee beans, and brewing coffee

are the actions that can be taken. The graph represents the progression of the system's state based on these actions, and the goal is achieved when the "Coffee: Present" state is reached.

Example

- Init(Have(Apple))
- Goal(Have(Apple) ∧Ate(Apple))
- Action(Eat(Apple), PRECOND: Have(Apple)
- EFFECT: ¬Have(Apple) ∧Ate(Apple))
- Action(Cut(Apple), PRECOND: ¬ Have(Apple)
- EFFECT: Have(Apple))

Planning as State-Space Search

- Imagine we have a smart office agent that can do tasks like printing, sending emails, and making coffee. Let's call it the "office agent."
- Now, if three people ask the office agent to do these tasks simultaneously, we want to figure out how the agent plans and searches for solutions in a limited space.

Imagine the office agent can find tasks and devices more easily if they are closer together in the space where it operates. To achieve this, the agent needs to know its current location, where the people assigning tasks are located, and where the required devices are.

There are two ways to represent the agent's situation:

1. Complete World Description:

Describes the situation by assigning a value to each suggestion.This method needs a lot of space, and it might not be practical for real-world problems.

2. Path from an Initial State:

Represents the situation by showing the sequence of actions taken to reach a state from the starting point.This method doesn't explicitly say "what's true in every state," which can make it tricky to figure out if two states are the same.

In simple terms, the office agent plans by figuring out how to move from its starting point to the desired tasks. However, both methods have their drawbacks: one takes up a lot of space, and the other might not always tell us exactly what's happening in every situation.

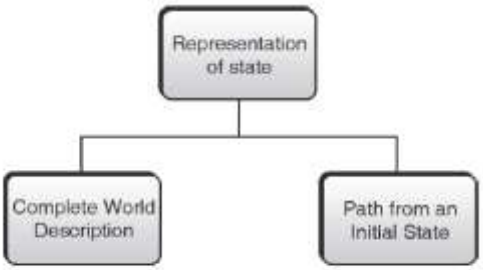


Fig. 3.16.2 : Representation of states

The Jug problem is a classic example in AI and planning, often referred to as the Water Jug problem. In this problem, you have two jugs of different capacities, and the goal is to measure out a specific quantity of water using these jugs. Let's detail the problem using planning:

Problem Description:

You have two jugs, one with a capacity of 4 gallons (Jug4) and the other with a capacity of 3 gallons (Jug3). There are no measuring marks on the jugs. You also have access to a water source (a pump) to fill the jugs.

State Space Representation:

The state of the system can be represented as an ordered pair (x, y), where x is the amount of water in Jug4 (0 to 4 gallons), and y is the amount of water in Jug3 (0 to 3 gallons). The initial state is (0, 0), and the goal is to reach a state (2, n), where n can be any value for Jug3 (since the problem does not specify how many gallons need to be in Jug3).

Rule set

1. (x,y) ⟶ (4,y) fill the 4- gallon jug
   If x<4
2. (x,y)⟶ (x,3) fill the 3-gallon jug
   If x<3
3. (x,y)⟶ (x-d,y) pour some water out of the 4- gallon jug
   If x>0
4. (x,y)⟶(x-d,y) pour some water out of the 3- gallon jug
   If y>0
5. (x,y) ⟶ (0,y) empty the 4- gallon jug on the ground
   If x>0
6. (x,y) ⟶(x,0) empty the 3- gallon jug on the ground
   If y>0
7. (x,y) ⟶ (4,y-(4-x)) pour water from the 3- gallon jug into the 4- gallon
   If x+y>=4 and y>0 jug until the 4-galoon jug is full
8. (x,y) ⟶ (x-(3-y),3)) pour water from the 4- gallon jug into the 3-gallon
   If x+y>=3 and x>0 jug until the 3-gallon jug is full

---

9. (x,y) ⟶ (x+y,0) pour all the water from the 3 -gallon jug into
   If x+y<=4 and y>0 the 3-gallon jug
10. (x,y) ⟶ (0,x+y) pour all the water from the 4 -gallon jug into
    If x+y<=3 and x>0 the 3-gallon jug
11. (0,2) ⟶ (2,0) pour the 2-gallon from the 3 -gallon jug into
    the 4-gallon jug
12. (2,y) ⟶(0,x) empty the 2 gallon in the 4 gallon on the ground

Here's a sequence of steps to reach the goal state (2, n):

Fill Jug3:

(0,0)→(0,3)

(0,0)→(0,3) (Applying Rule 2)

Pour from Jug3 to Jug4:

(0,3)→(2,1)

(0,3)→(2,1) (Applying Rule 7)

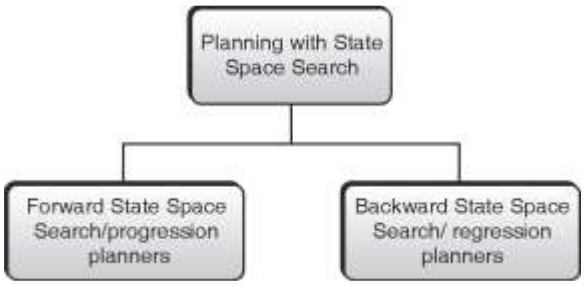
Empty Jug3 on the ground:

(2,1)→(2,0)

(2,1)→(2,0) (Applying Rule 6)

This is one possible solution to the Water Jug problem. The planning involves applying the rules systematically to reach the desired state, considering the constraints and capacities of the jugs.

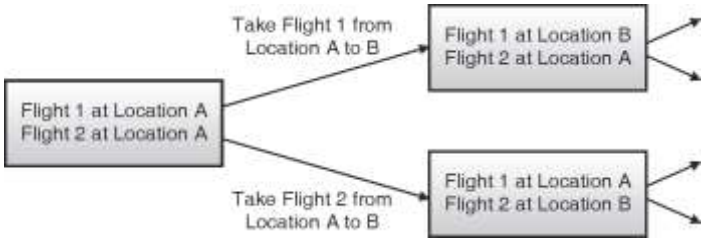
Classification of Planning with State Space Search



Forward state-space search

- "Forward state-space search," often referred to as a "progression planner," is a deterministic planning technique where a sequence of actions is planned from the initial state to attain the goal.
- In this approach, we start with the initial state and progress towards the final goal state, taking into account the potential effects of actions at each state.
- To execute a forward state space search, certain prerequisites must be met, including having information about the initial world state, knowledge of the available actions of the agent, and a description of the goal state.
- The details of available actions encompass both the preconditions required for executing an action and the effects resulting from that action.

Consider Fig. 3.18.1, which depicts a state-space graph of a progression planner using a simple example involving two flights, flight 1 and flight 2, initially located at point A. Both flights are moving from location A to location B. In the first scenario, only flight 1 moves to location B, resulting in flight 1 being at B, and flight 2 remaining at its original location, A. In the second scenario, only flight 2 moves to location B, resulting in flight 2 being at B, while flight 1 remains at A.



The rectangles in the figure represent the states of the flights (i.e., their current locations), and the lines illustrate the corresponding actions between states (i.e., the movement from one location to another). Each line originating from a state corresponds to all permissible actions that can be taken when the agent is in that state.

This graphical representation effectively captures the progression of the system's states and the actions taken to achieve the desired goal.

Progression planner algorithm

1.State Space Search Problem Formulation:

- Define the initial state, representing the first state of the planning problem with a set of positive literals. Absent literals are considered false.
- If preconditions are satisfied, actions are favored. For satisfied preconditions, positive effect literals are added; otherwise, negative effect literals are deleted.
- Perform goal testing by checking if the state satisfies the specified goal.

2.Consider example of A* algorithm. A complete graph search is considered as a complete planning algorithm. Functions are not used.

3. Progression planner algorithm is supposed to be inefficient because of the irrelevant action problem and requirement of good heuristics for efficient search.

Backward state-space search or Regression planner

"Backward state-space search" is also called as **"regression planner"** from the name of this method you can make out that the processing will start from the finishing state and then you will go backwards to the initial state.

– So basically we try to backtrack the scenario and find out the best possibility, in-order to achieve the goal to achieve this we have to see what might have been correct action at previous state.

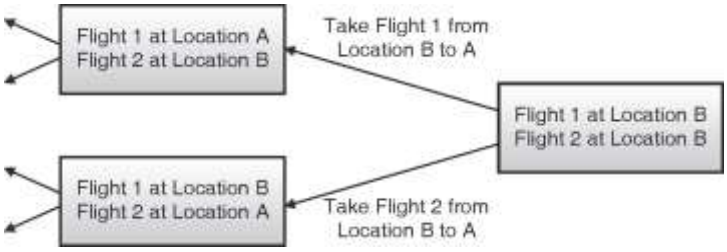
– In forward state space search we used to need information about the successors of the current state now, for backward state-space search we will need information about the predecessors of the current state.

– Here the problem is that there can be many possible goal states which are equally acceptable. That is why this approach is not considered as a practical approach when there are large numbers of states which satisfy the goal.

– Let us see flight example, here you can see that the goal state is flight 1 is at location B and flight 2 is also at location B.

We can see in Fig. 3.19.1. If this state is checked backwards we have two acceptable states in one state only flight 2 is at location B, but flight 1 is at location A and similarly in 2nd possible state flight 1 is already at location B, but flight 2 is at location A.

– As we search backwards from goal state to initial state, we have to deal with partial information about the state, since we do not yet know what actions will get us to goal. This method is complex because we have to achieve a conjunction of goals.



– In this Fig. 3.19.1, rectangles are goals that must be achieved and lines shows the corresponding actions.

Regression algorithm

1. Firstly predecessors should be determined :
   - To do this we need to find out which states will lead to the goal state after applying some actions on it.
   - We take conjunction of all such states and choose one action to achieve the goal state.
   - If we say that "X" action is relevant action for first conjunct then, only if pre-conditions are satisfied it works.
   - Previous state is checked to see if the sub-goals are achieved.

2.Actions must be consistent it should not undo preferred literals. If there are positive effects of actions which appear in goal then they are deleted. Otherwise Each precondition literal of action is added, except it already appears.

3. Main advantage of this method is only relevant actions are taken into consideration. Compared to forward search,backward search method has much lower branching factor.

Heuristics for State-Space Search

Progression or Regression is not very efficient with complex problems. They need good heuristics to achieve better efficiency. The best solution is NP Hard (NP stands for Non-deterministic and Polynomial-time).

- There are two ways to make state space search efficient:
- Use a linear method: Add the steps which build on their immediate successors or predecessors.
- Use a partial planning method: As per the requirement at execution time ordering constraints are imposed on the agent.

Partial Ordered Planning (POP)

- In the context of Partial Ordered Planning (POP), the ordering of actions is partial, and it does not explicitly specify the sequence in which two actions will occur within the plan.
- Unlike Total Order Planning (TOP), where a complete sequence is determined, POP allows for partial ordering of actions. This flexibility is advantageous in scenarios where the environment is non-cooperative and presents challenges for total ordered planning.
- To illustrate partial ordered planning, consider the example of wearing shoes

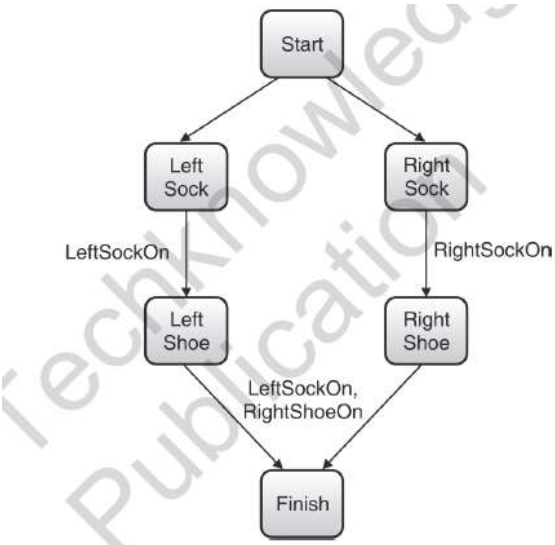


Fig. 3.21.1 : Partial order planning of Wearing Shoe

A partial order planning combines two action sequences

- First branch covers left-sock and left-shoe.
- In this case to wear a left shoe, wearing left sock is the precondition, similarly.
- Second branch covers right-sock and right-shoe.
- Here, wearing a right sock is the precondition for wearing the right shoe.

– Once these actions are taken we achieve our goal and reach the finish state.

Partial Ordered Plan (POP) as a Search Problem

components of a Partial Ordered Plan (POP) as a Search Problem

If we consider Partial Ordered Planning (POP) as a search problem, the states can be conceptualized as small plans, typically representing unfinished actions. An empty plan comprises only starting and finishing actions. Every plan consists of four main components:

Set of Actions:

Actions are the steps of a plan that can be performed in a particular order to achieve a goal. For instance, the set of actions for the task of wearing shoes could be {Start, Rightsock, Rightshoe, Leftsock, Leftshoe, Finish}.

Set of Ordering Constraints/ Preconditions:

Preconditions are considered as ordering constraints, indicating the necessary order in which actions must occur. For example, the set of ordering could be {Right-sock < Right-shoe; Left-sock < Left-shoe}, signifying that wearing a sock must precede wearing a shoe.

If constraints form a cycle, it represents inconsistency. A consistent plan should be free of cyclic preconditions.

Set of Causal Links:

Causal links define the relationships between actions, where action A achieves an effect that is a precondition for action B. For example, buying an apple may have the effect of eating the apple, and the precondition for eating the apple is cutting it.

There can be conflict if there is an action C that has an effect ¬E and, according to the ordering constraints it comes after action A and before action B.

– Say we don't want to eat an apple instead of that we want to make a decorative apple swan. This action can be

between A and B and It does not have effect "E".

- **For example:** Set of Causal Links = {Right-sock->Right-sock-on → Right-shoe, Leftsock → Leftsockon → Leftshoe, Rightshoe → Rightshoeon → Finish, leftshoe → leftshoeon → Finish }.
- To have a consistent plan there should not be any conflicts with the causal links.

Set of Open Preconditions:

Preconditions are considered open if they cannot be achieved by actions in the current plan. The least commitment strategy may be used by delaying the choice during the search process.
A consistent plan should not have any open preconditions.

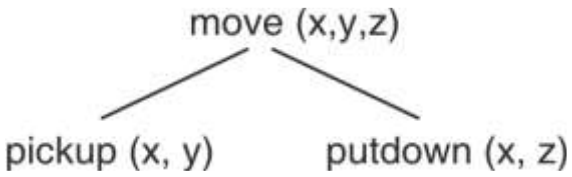
Consistent Plan is a Solution for POP Problem

In the context of Partial Ordered Planning (POP), a consistent plan serves as a solution due to its absence of cyclic constraints, lack of conflicts in causal links, and the elimination of open preconditions. The process of solving a POP problem involves operators adding links and steps from existing plans to fulfill open preconditions. These steps are then ordered in relation to other steps to avoid potential conflicts. If an open precondition proves unattainable, a backtracking approach is employed to reevaluate the steps and attempt to solve the problem within the POP framework.

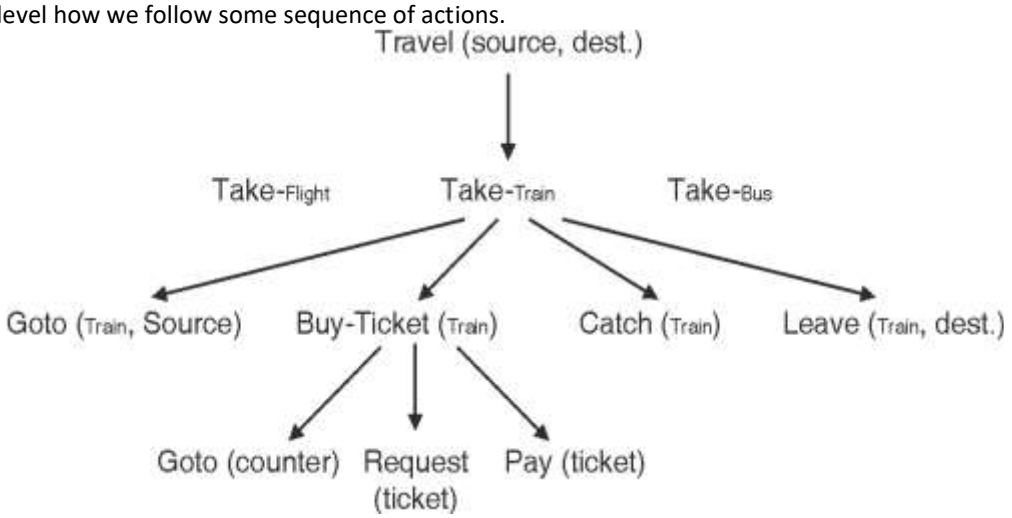
The efficiency of partial ordered planning lies in its ability to progress from a vague plan to a complete and correct solution swiftly. This is achieved by efficiently navigating the state space, addressing interactions between sub-plans, and solving large-scale planning problems with fewer steps. The method's effectiveness stems from its capacity to adapt and refine plans iteratively, optimizing the planning process and facilitating quicker convergence to a viable solution.

Hierarchical planning

- Hierarchical planning is also called as plan decomposition. Generally plans are organized in a hierarchical format.
- Complex actions can be decomposed into more primitive actions and it can be denoted with the help of links between various states at different levels of the hierarchy. This is called as operator expansion.



shows, how to create a hierarchical plan to travel from some source to a destination. Also you can observe, at every level how we follow some sequence of actions.



POP One Level Planner

If the type of problem is simple then we can make use of one level planner. For complex problems, one level planner cannot provide good solution.

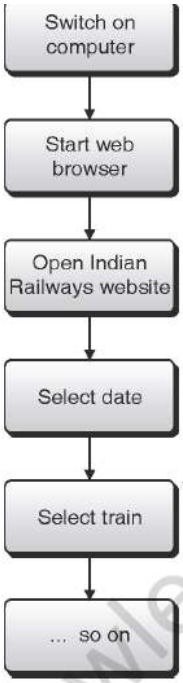


Fig. 3.22.3 : One Level planning example

Planner

1. First identify a hierarchy of major conditions.
2. Construct a plan in levels (Major steps then minor steps), so we postpone the details to next level.
3. Patch major levels as detail actions become visible.
4. Finally demonstrate.

Example :

Actions required for "Travelling to Rajasthan" can be given as follows :

- – Opening yatra.com (1)
- – Finding train (2)
- – Buy Ticket (3)
- – Get taxi(2)
- – Reach railway station(3)
- – Pay-driver(1)
- – Check in(1)
- – Boarding train(2)
- – Reach Rajasthan (3)

1st level plan

Buy Ticket (3), Reach Railway Station(3), Reach Rajasthan (3)

2nd level plan

Finding train (2), Buy ticket (3), Get taxi(2), Reach railway station (3), Boarding train(2), Reach Rajasthan (3).

3rd level plan (final)

Opening yatra.com (1), Finding train (2), Buy ticket (3), Get taxi(2), Reach Railway station (3), Pay-driver(1), Checkin(1), Boarding train(2), Reach Rajasthan (3).

Planning and Acting in Nondeterministic Domain

Determining the condition of state depends on available knowledge. In real world, knowledge availability is always limited, so most of the time, conditions are non deterministic.

– The amount or degree of indeterminacy depends upon the knowledge available. The inter determinacy is called "bounded indeterminacy" when actions can have unpredictable effects.

– Four planning strategies are there for handling indeterminacy :

1. **Sensorless Planning:**

In sensorless planning, the agent lacks complete information about its environment. The challenge is to devise a plan without relying on sensors to observe the current state. This requires the incorporation of robust strategies and alternative actions to handle uncertainties effectively. For instance, a mobile robot navigating an unfamiliar environment without precise sensors must plan its movements based on a model of possible states and actions.

2. **Contingent Planning:**

Contingent planning deals with situations where the outcome of actions is uncertain, and plans must be adaptable. Plans incorporate branches or contingent actions, allowing for different sequences based on diverse outcomes. This flexibility is crucial in domains where unforeseen events may impact the success of actions. An example is a logistics system planning for contingencies like delayed shipments or unexpected obstacles.

3. **Online Replanning:**

Online replanning involves adapting plans in real-time as new information becomes available. In dynamic environments, the execution of actions may reveal unexpected changes, necessitating adjustments to the plan. Real-time replanning allows agents to respond promptly to unforeseen circumstances. For instance, a self-driving car may replan its route if road conditions change due to traffic or construction.

4. **Multiagent Planning**:

Multiagent planning addresses scenarios where multiple agents interact, each pursuing its objectives. Coordination and cooperation among agents are crucial to achieving collective goals. This involves dealing with communication constraints, conflicting interests, and uncertainties in the actions of other agents. Applications range from collaborative robotics to multiplayer gaming scenarios.

Whatever planning we have discussed so far, belongs to single user environment. Agent acts alone in a single user environment.

- When the environment consists of multiple agent to, then the way a single agent plan its action get changed.
- We have a glimpse of environment where multiple agents have to take actions based on current state. The environment could be co-operative or competitive. In both the cases agent's action influences each other.
- Few of the multi agent planning strategies are listed below :

(i) Co-operation

In co-operation strategy agents have joint goals and plans. Goals can be divided into sub goals but ultimately combined to achieve ultimate goal.

(ii) Multibody planning

Multi body planning is the strategy of implementing correct joint plan.

(iii) Co-ordination mechanisms

These strategies specify the co-ordination between co-operating agents. Co-ordination mechanism is used in several co-operating plannings.

(iv) Competition

Competition strategies are used when agents are not co-operating but competing with each other. Every agent want to achieve the goal first.