IARE
INSTITUTE OF AERONAUTICAL ENGINEERING

Course Title - Logic Programming for Artificial Intelligence

Topic Title – Planning with State-Space Search

Presenter's Name – Ms. Bidyutlata Sahoo
Presenter's ID – IARE11028
Department Name – CSE (AI & ML)
Lecture Number - 01
Presentation Date – 23/11/2024

Course Outcome

At the end of the course, students should be able to:

CO5: Develop a comprehensive understanding of advanced AI planning strategies and diverse learning paradigms to solve complex problems.

Topic Learning Outcome

Analyze the sequence of actions needed to achieve specific goals.

Introduction to Planning

Introduction to Planning

- Planning is an ***important topic in Artificial Intelligence.***
- Planning is ***required for every task (Job).*** For example, reaching a particular destination requires planning.
- To reach a particular destination, first we should *find the best route(or strategy) and then identify a set of actions* to be done *at a particular time*, is called as ***planning.***
- *Planning is nothing but deciding a set of actions to be performed, in agent environment based on the perception to achieve the decided goal.*

Introduction to Planning (Contd..)

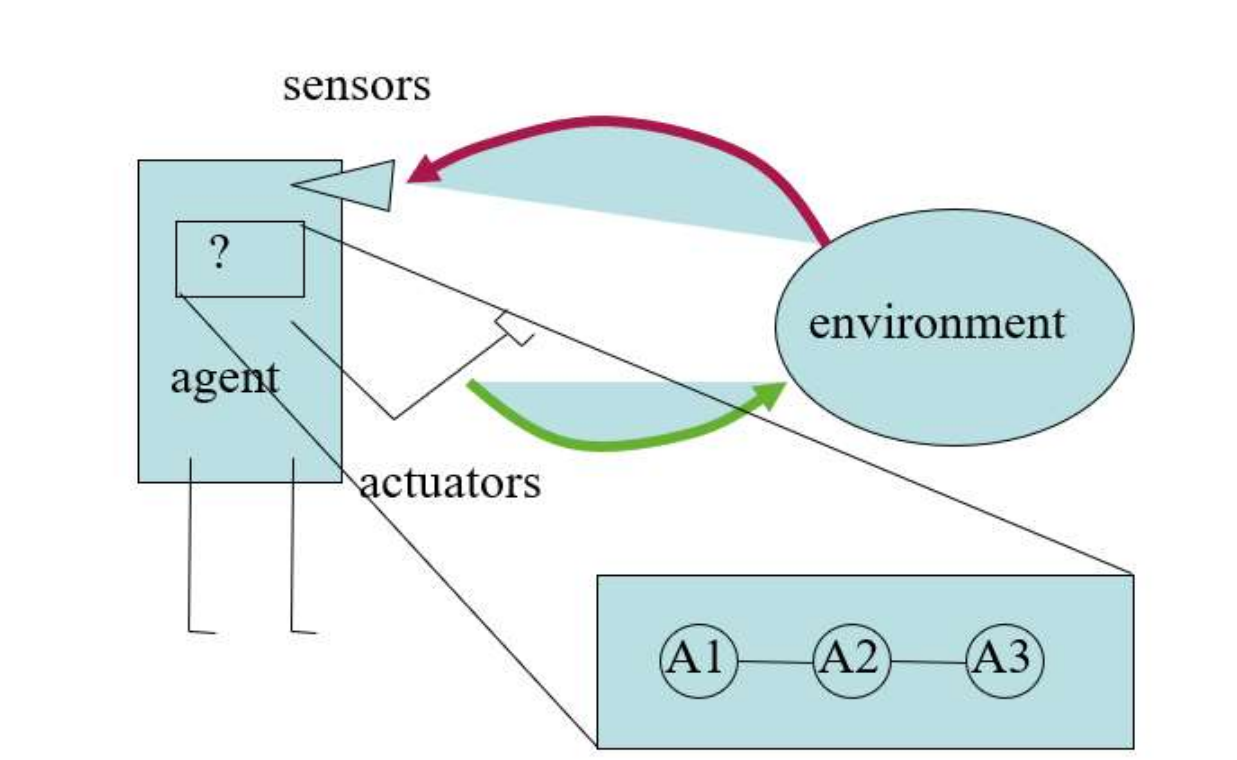
- For any planning system, we need the
  - ✓*Domain description*
  - ✓*Action specification and*
  - ✓*Goal description.*
- A plan is assumed to be a *sequence of actions.*
- Each action has its own set of *preconditions.*
- *All the preconditions to be satisfied before performing the action.*
- Some *effects* can ***be*** *either positive or negative*.

Introduction to Planning (Contd..)

There are 2 types of planning:

- *Classical planning* environments.
  - ***Fully observable***, ***deterministic, finite, static and discrete.***
  - **Ex:** A robot in a warehouse environment with clearly defined tasks.
- *Nonclassical planning* is for ***partially observable or stochastic*** environments.
  - The nonclassical planning involves a different set of algorithms and agent designs.
  - **Ex:** An autonomous vehicle navigating through a city.

Planning Agent



Planning Agent

In planning, the idea is that ***you're given some description of a*** *starting state or states****; a*** *goal state or states****; and some*** *set of possible actions* ***that the agent can take***.

And ***you want to find the sequence of actions*** that get you from the start state to the goal state.

Planning vs. Problem Solving

- *Planning* and *problem-solving* methods can often solve the same sorts of problems.
- *Planning is more powerful* because of the representations and methods used.
- States, goals, and actions are decomposed into sets of sentences (usually in first-order logic)
- Search often proceeds through plan space rather than state space (though first we will talk about state-space planners)
- Subgoals can be planned independently, reducing the complexity of the planning problem.

Planning
with
State Space Search

Planning with State Space Search

- The most straightforward ***approach of planning*** algorithm is state space search.
  - **Forward state space search(Progression)**
  - **Backward state space search(Regression)**
- The descriptions of actions in a planning problem and specify both preconditions and effects.
- It is possible to search in both direction: ***either forward from the initial state or backward from the goal.***
- We can also use the explicit action and goal representation to derive effective heuristics automatically.

Problem formulation for Progression

- **Initial state:**
  - **Initial state of the planning problem.**
- **Actions:**
  - **Applicable to the current state.**
  - **First actions preconditions are satisfied then successor states are generated.**
  - **Add positive literals to add list and negative literals to delete list.**
- **Goal test:**
  - **Whether the state satisfies the goal of the planning.**
- **Step Cost:**
  - **Each action is 1(assumed)**

Progression

- From initial state, **search forward** by selecting operators / actions whose preconditions can be unified with literals in the state.
- New state include positive literals of effect, the negated literals of effect are deleted.
- ***Search forward until goal unifies*** with resulting state.
- This is forward state space search uses ***STRIPS (Stanford Research Institute Problem Solver)*** operators.

Progression

Example : Transportation of air cargo between airports.

$$
\begin{aligned}
&Init(At(C_1, SFO) \land At(C_2, JFK) \land At(P_1, SFO) \land At(P_2, JFK) \\
&\quad \land Cargo(C_1) \land Cargo(C_2) \land Plane(P_1) \land Plane(P_2) \\
&\quad \land Airport(JFK) \land Airport(SFO)) \\
&Goal(At(C_1, JFK) \land At(C_2, SFO)) \\
&Action(Load(c, p, a), \\
&\quad \text{PRECOND}: At(c, a) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a) \\
&\quad \text{EFFECT}: \neg At(c, a) \land In(c, p)) \\
&Action(Unload(c, p, a), \\
&\quad \text{PRECOND}: In(c, p) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a) \\
&\quad \text{EFFECT}: At(c, a) \land \neg In(c, p)) \\
&Action(Fly(p, from, to), \\
&\quad \text{PRECOND}: At(p, from) \land Plane(p) \land Airport(from) \land Airport(to) \\
&\quad \text{EFFECT}: \neg At(p, from) \land At(p, to))
\end{aligned}
$$

Progression

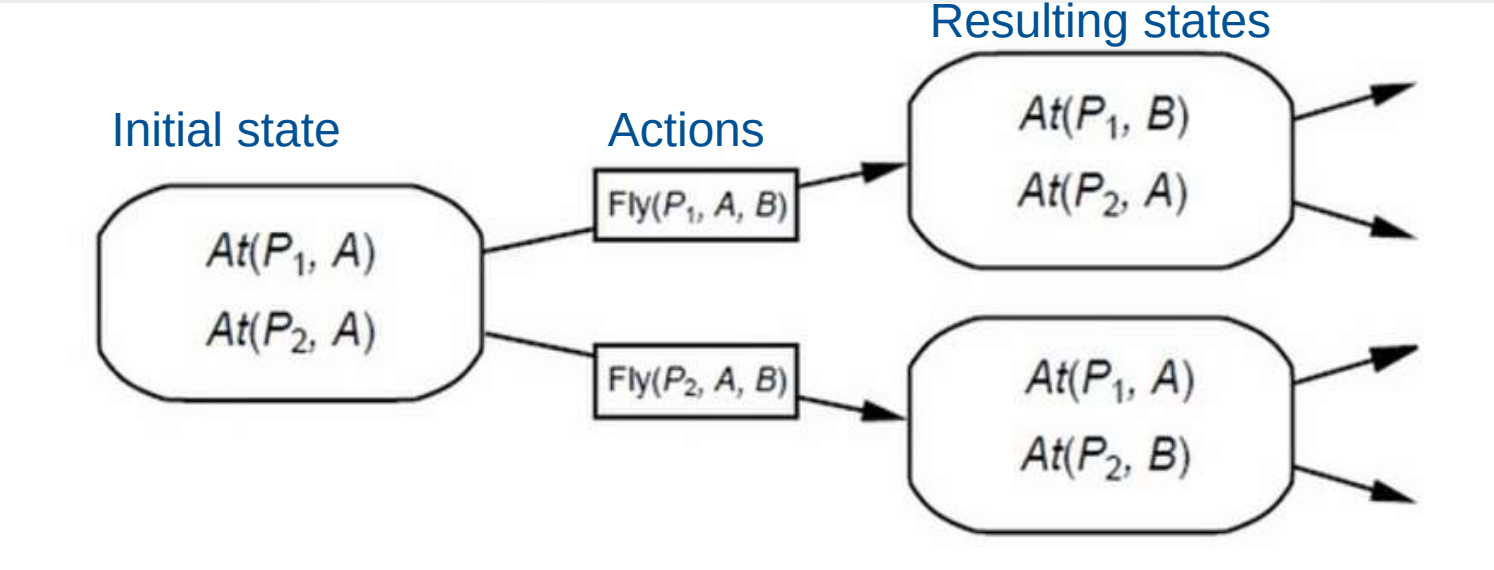
Example: Blocks world

Progression

$$Init(On(A, Table) \land On(B, Table) \land On(C, A)$$
$$\land Block(A) \land Block(B) \land Block(C) \land Clear(B) \land Clear(C) \land Clear(Table))$$
$$Goal(On(A, B) \land On(B, C))$$
$$Action(Move(b, x, y),$$
$$\text{PRECOND: } On(b, x) \land Clear(b) \land Clear(y) \land Block(b) \land Block(y) \land$$
$$(b \neq x) \land (b \neq y) \land (x \neq y),$$
$$\text{EFFECT: } On(b, y) \land Clear(x) \land \neg On(b, x) \land \neg Clear(y))$$
$$Action(MoveToTable(b, x),$$
$$\text{PRECOND: } On(b, x) \land Clear(b) \land Block(b) \land Block(x),$$
$$\text{EFFECT: } On(b, Table) \land Clear(x) \land \neg On(b, x))$$

Forward(Progression) state space search

- Starting from the initial state and using the problems actions to search forward for the goal state.



FSSS Algorithm

- Compute whether or not a state is a goal state.
- Find the set of all actions that are applicable to a state and Compute a successor state that is the result of applying an action to a state.
- Algorithm takes as input the statement P = (O, s0, g) of a planning problem P. (O contains a list of actions)
- If P is solvable, then forward-search (O, s0, g) returns a solution plan; otherwise it returns failure.

Problem formulation for Regression

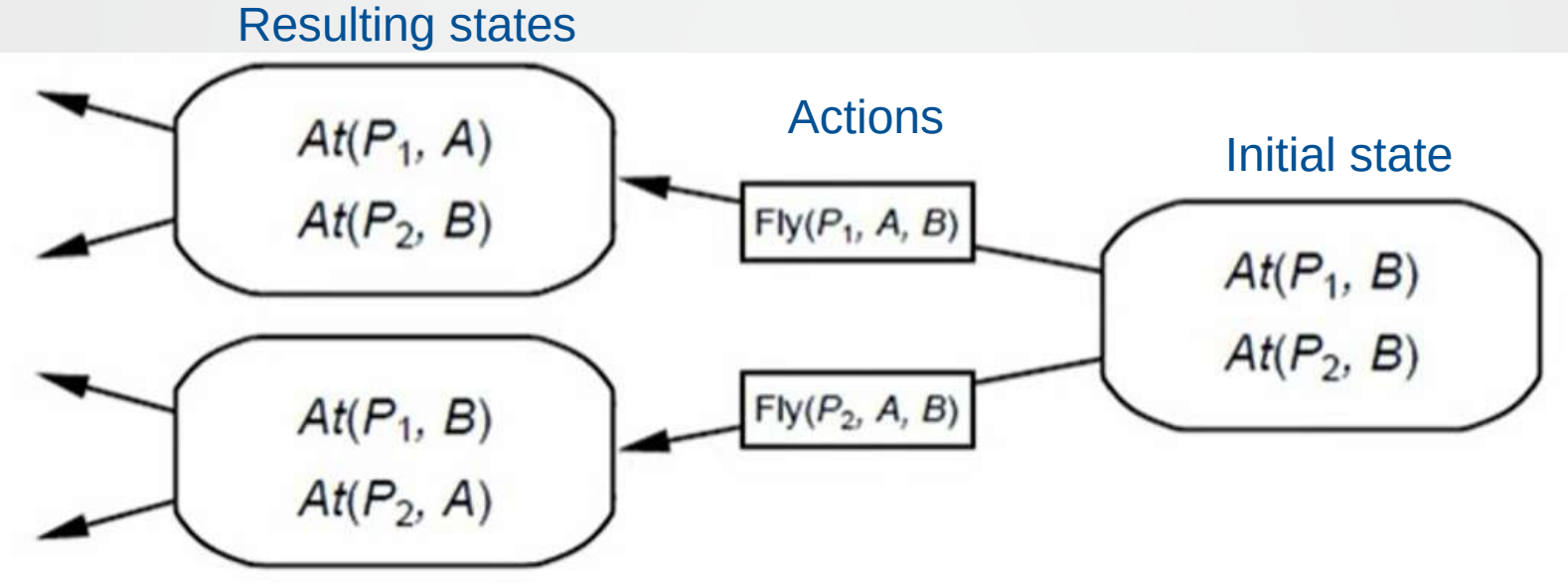
- **Initial state:**
  - **The goal state is considered as the "initial state" for regression search.**
- **Actions:**
  - **Actions or operators are applied in reverse.**
  - **Each action must have a well-defined inverse or precondition, which means identifying the actions that could have led to the current state. The problem solver needs to ensure these preconditions are satisfied in the previous states.**
- **Goal test:**
  - **The goal state is where the search starts.**
- **Step Cost:**
  - **Each action is 1(assumed)**

Regression

- The goal state must unify with at least one of the positive literals in the operator's effect.
- Its preconditions must hold in the previous situation, and these become subgoals which might be satisfied by the initial conditions.
- ***Perform backward chaining*** from goal.
- Again, this is just state space search using ***STRIPS operators.***

Backward(regression) state space search

- Backward state space search starting from the goal state(s).
- Using the inverse of the actions to search backward for the initial state.



BSSS Algorithm

- Starts at the goal.
- Test the goal is initial state, otherwise
- Apply inverses of the planning operators to produce subgoals.
- The algorithm will stop, if we produce a set of subgoals that satisfies the initial state.

Languages for Planning Problems

STRIPS

- ***Stanford Research Institute Problem Solver.***
- Historically important.
- Use first-order logic and theorem proving to plan strategies from start to goal.
- STRIPS language:
  -"*Classical" approach that most planners use*.
  -Lends itself to efficient planning algorithms

Languages for Planning Problems

STRIPS REPRESENTATION

- States are represented as conjunctions

  In(Robot,room) $\wedge$ $\neg$ In(Charger, r) $\wedge$ ...
- Goals are represented as conjunctions:

  (implicit $\exists$ r) In(Robot, r) $\wedge$ In(Charger, r)
- Actions (operators):
  - Name: Go(here, there)
  - Preconditions: expressed as conjunctions
    At(Robot, here) $\wedge$ Path(here, there)
  - Postconditions (effects): expressed as conjunctions
    At(Robot, there) $\wedge$ $\neg$ At(Robot, here)
- Variables can only be instantiated with objects of the correct type

Languages for Planning Problems

STRIPS REPRESENTATION

- Operators have a name, preconditions and postconditions or effects
- Preconditions are conjunctions of *positive literals*
- Postconditions/effects are represented in terms of:
  - *Add-list*: list of propositions that become true after the action
  - *Delete-list*: list of propositions that become false after the action

Languages for Planning Problems(Contd..)

ADL

- ***Action Description Languages.***
- Relaxed some of the restrictions that made STRIPS inadequate for real-world problems.

PDDL

- ***Planning Domain Definition Language.***
- Revised & enhanced for the needs of the International Planning Competition.
- Currently version 3.1
- Includes STRIPS and ADL

References

- Stuart Russell and Peter Norvig, “Artificial Intelligence”, 2nd edition, Pearson Education, 2003.
- Saroj Koushik, “Artificial intelligence”.
- NPTEL

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Thank You

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Course Title - Logic Programming for Artificial Intelligence

Topic Title – Partial Order Planning(POP)

Presenter's Name – Ms. Bidyutlata Sahoo
Presenter's ID – IARE11028
Department Name – CSE (AI & ML)
Lecture Number - 02
Presentation Date – 25/11/2024

Course Outcome

At the end of the course, students should be able to:

CO5: Develop a comprehensive understanding of advanced AI planning strategies and diverse learning paradigms to solve complex problems.

Topic Learning Outcome

Solve complex planning task which involves concurrency, flexibility to obtain more optimal and feasible solutions.

Partial Order Planning (POP)

Partial Order Planning

- ***Progression and Regression are totally ordered plan search forms. (Linear Planner)***
  - They can't take advantage of problem decomposition.
  - Decisions must be made *on how to sequence actions* on all the subproblems.
- ***POP: (Non-Linear Planner)***
  - Work on subgoals independently.
  - Construct plan by working on obvious or important decisions first.
  - Least commitment strategy: delay choice during search.

Partial Order Planning

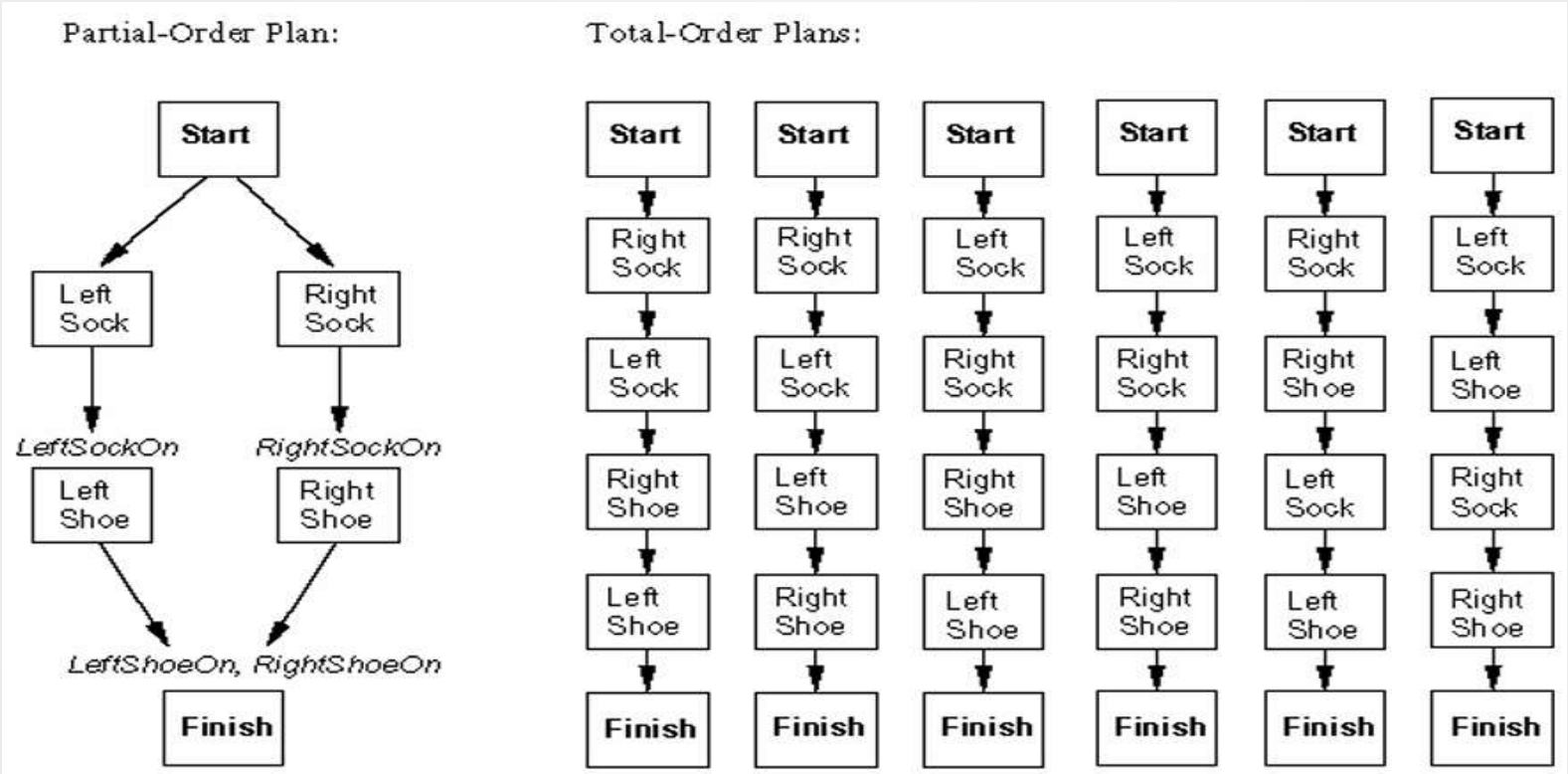
- In Partial order planning *ordering of action is partial.*
- Partial order planning *does not specify which action will come first* out of the two actions which are in plan.
- It is also called as ***Non-linear Planner.***
- ***Works on several sub-goals independently.***
- The ***sub-goals*** are completed with ***sub-plans.***
- The ***sub-plans are combined in any order .***
- *It contrast with total-order planning (**Linear Planner**), which produces an exact ordering of actions.*

Partial Order Planning builds on these ideas

- *Adopts a plan space representation* – States in the state space correspond to partial plan.
- *Relaxes the requirement that the solutions are constructed sequentially* – steps can be added to the plan in any order.
- *Adopts the principle of least commitment* – decisions about steps ordering and variable bindings are delayed as long as possible.
- *Returns a plan in which some steps are ordered and other steps are unordered.*

Partial Order Planning – Shoe Example-1

Wearing shoe to understand partial order planning.



Partial Order Planning – Shoe Example

Solution:

- Goal(RightShoeOn ^ LeftShoeOn)
- Init()
- Action: RightShoe
  - PRECOND: RightSockOn
  - EFFECT: RightShoeOn
- Action: RightSock
  - PRECOND: None
  - EFFECT: RightSockOn
- Action:LeftShoe
  - PRECOND: LeftSockOn
  - EFFECT: LeftShoeOn
- Action: LeftSock
  - PRECOND: None
  - EFFECT: LeftSockOn

How to define POP?

- A set of actions, that make up the steps of the plan
- A set of ordering constrain $A \prec B$
  - A before B
- A set of causal links: $A \xrightarrow{p} B$
  - A achieves P for B $RightSock \xrightarrow{RightSockOn} RightShoe$
  - May be conflicts if C has the effect of $\neg$P and if C comes after A and before B
- A set of open preconditions:
  - A precondition is open, if it is not achieved by some action in the plan

Shoe Example – The final Plan

- The final plan has the following components:
- Actions: {RightSock, RightShoe, LeftSock, LeftShoe, Start, Finish}
- Orderings: {RightSock ‹ RightShoe, LeftSock ‹ LeftShoe}
- Open preconditions: {}
- Links:

$$RightSock \xrightarrow{RightSockOn} RightShoe$$

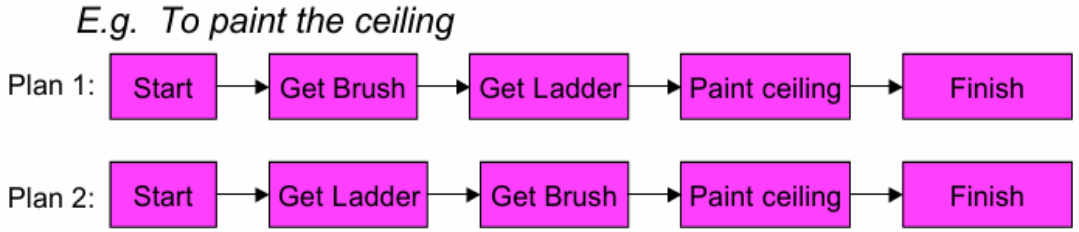
$$LeftSock \xrightarrow{LeftSockOn} LeftShoe$$

$$RightShoe \xrightarrow{RightShoeOn} Finish$$

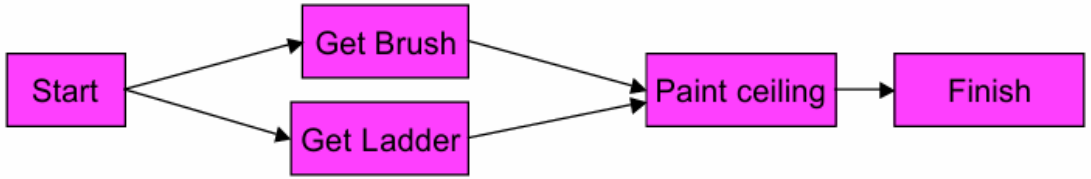
$$LeftShoe \xrightarrow{LeftShoeOn} Finish$$

Partial Order Planning – Example-2

- Total order: Plan is always a strict sequence of actions



- Partial order: Plan steps may be unordered



Partial Order Planning - Advantages

- *Plan steps may be unordered*(plan will be ordered, or linearized, before execution)
- *Handles concurrent plans*.
- *Least commitment can lead to shorter search times*.
- *Sound and complete.*
- *Typically produces the optimal plan.*

Partial Order Planning - Disadvantages

- There is *no order to solve a problem.*
- There is *no perfect solution*.

References

- Stuart Russell and Peter Norvig, “Artificial Intelligence”, 2nd edition, Pearson Education, 2003.
- Saroj Koushik, “Artificial intelligence”.
- NPTEL

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Thank You

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Course Title - Logic Programming for Artificial Intelligence

Topic Title –Planning Graphs

Presenter's Name – Ms. Bidyutlata Sahoo
Presenter's ID – IARE11028
Department Name – CSE (AI & ML)
Lecture Number - 03
Presentation Date – 28/11/2024

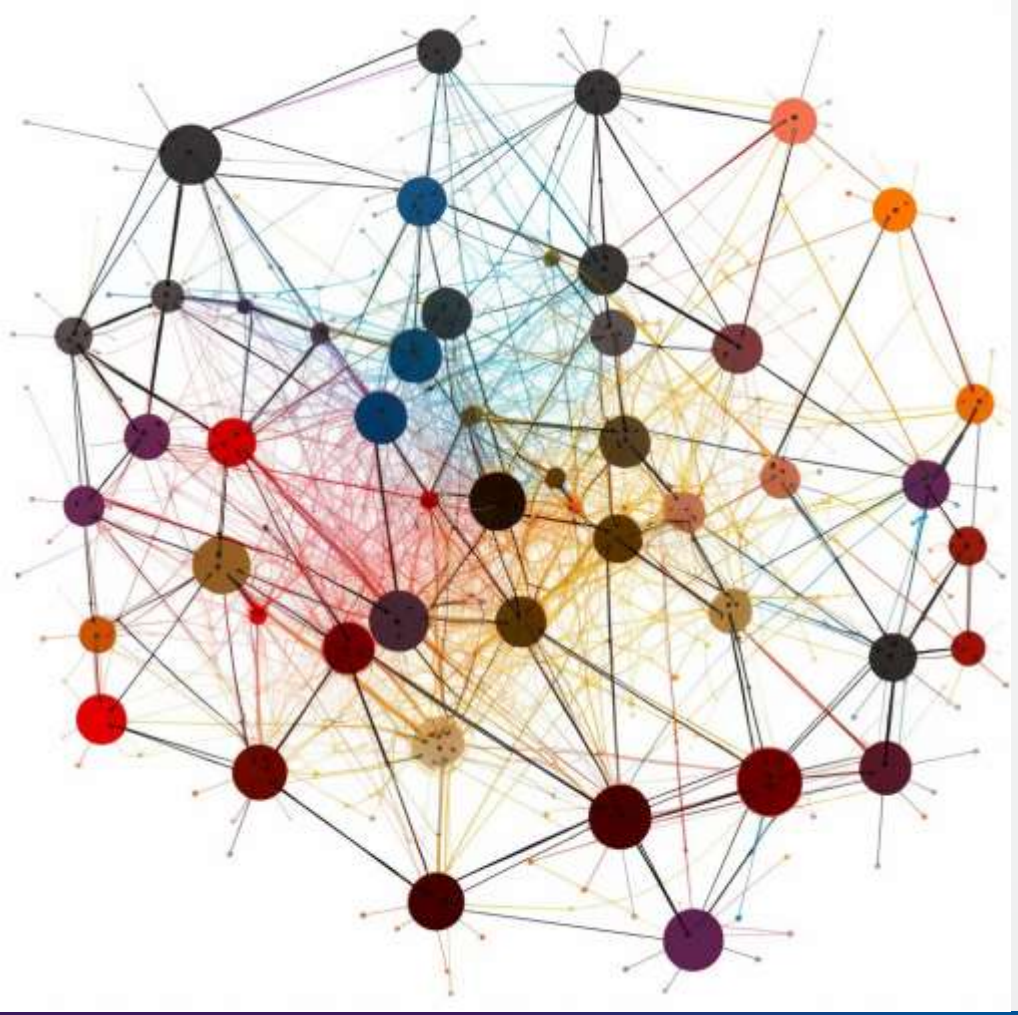
Course Outcome

At the end of the course, students should be able to:

CO5: Develop a comprehensive understanding of advanced AI planning strategies and diverse learning paradigms to solve complex problems.

Topic Learning Outcome

Construct the planning graph in a planning problem to represent the states and actions for better visualization.



Planning Graphs

Planning Graphs

- In Artificial Intelligence (AI), a planning graph *is a data structure used in automated planning*, which ***helps to solve planning problems efficiently by representing*** *actions and states* ***over time***.
- It was developed by *Avrim and Merrick in 1995.*
- It provides a compact way to explore the possible actions and their effects in a sequence, *making the planning process faster and more effective.*
- It is also called as *graph plan***.**

Planning Graphs (Contd..)

- Planning graphs are also *used as a source of heuristics* (an estimate of how many steps it takes to reach the goal).
- Planning graph is *an approximation of a complete tree of all possible actions and their results.*
- It gives the *relation* between *action and states*, the precondition must be satisfied the action.

Planning Graphs (Contd..)

- Planning graph is ***organised into levels.***
- **Level S_0**: initial state, consisting of nodes representing each fluent that holds in S_0.
- **Level A_0**: each ground action that might be applicable in S_0.
- Then **alternate Si and Ai.**
- **Si** *contains fluents (or propositions)* which could hold at time i, (may be both P and ¬P); literals may show up too early but never too late.
- **Ai** contains *actions* which could have them preconditions satisfied at i.

Planning Graphs (Contd..)

- Propositional problem will look at, what is the *starting state*, what are the *objects* in the domain, and it will produce all the *possible actions* and works with those actions.
- **We construct the planning graph from left to right.**
  - *We keep inserting actions and propositions.*
  - *Then continue with next level actions and propositions…*
  - *Until we get the goal proposition appear on the propositional layer.*
  - *And they are not mutually exclusive.*

Planning Graphs (Contd..)

- There are **2 states in the planning graph problem:**
  - *Construct the planning graph.*
  - *Search for solution*
- If we can’t get solution, then extend the planning graph and search for solution, and keep doing that until we get the solution.
- *Planning graphs only work for propositional problems.*

Planning Graph – Example
(The “have cake and eat cake too” problem)

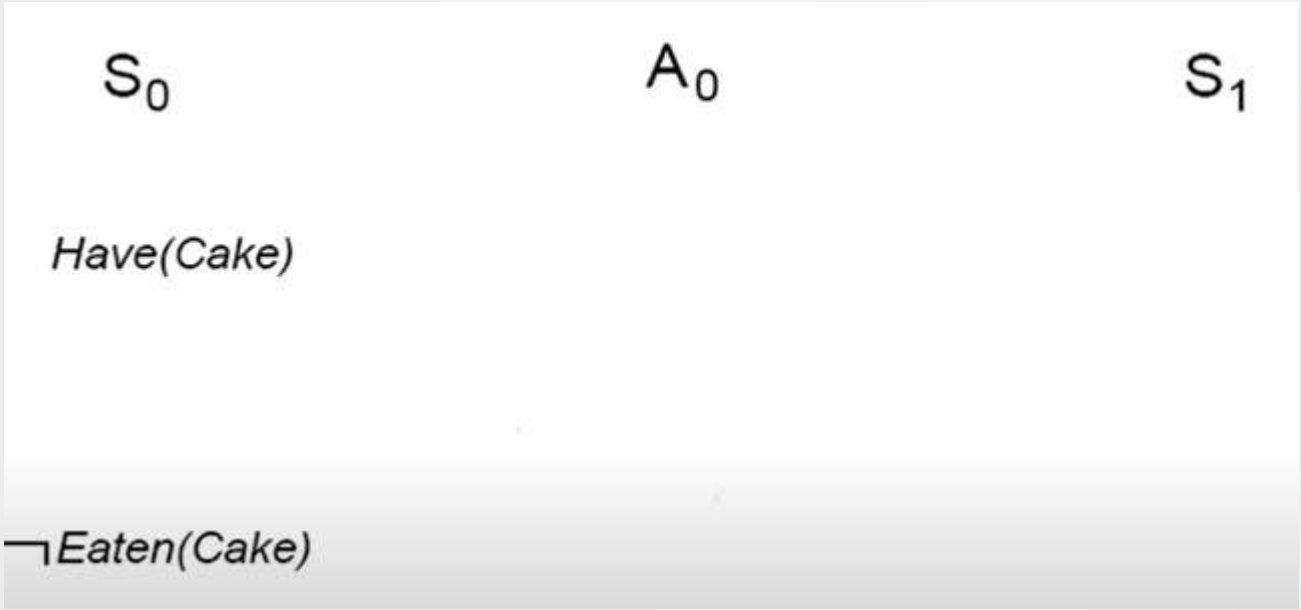
Initial state: $Have(Cake)$

Goal: $Have(Cake) \wedge Eaten(Cake)$

$Eat(Cake)$:
PRECOND: $Have(Cake)$
EFFECT: $\neg Have(Cake) \wedge Eaten(Cake)$

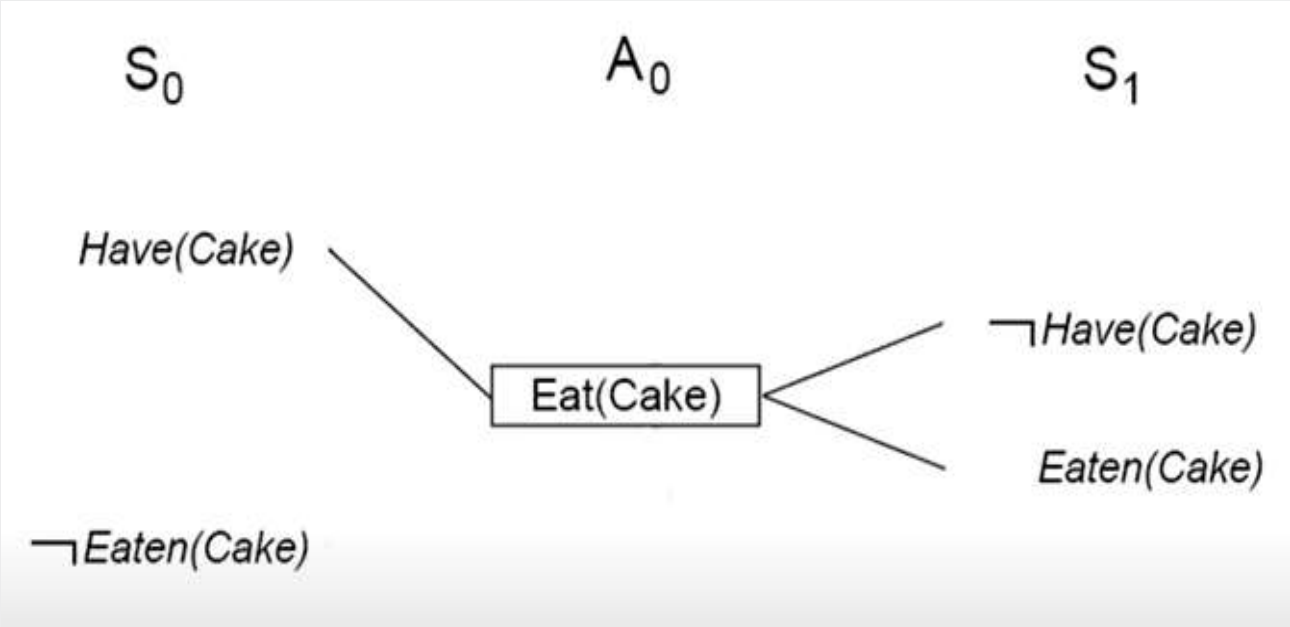
$Bake(Cake)$:
PRECOND: $\neg Have(Cake)$
EFFECT: $Have(Cake)$

Planning Graph – Example (Contd..)



Create level 0 from initial problem state.

Planning Graph – Example (Contd..)

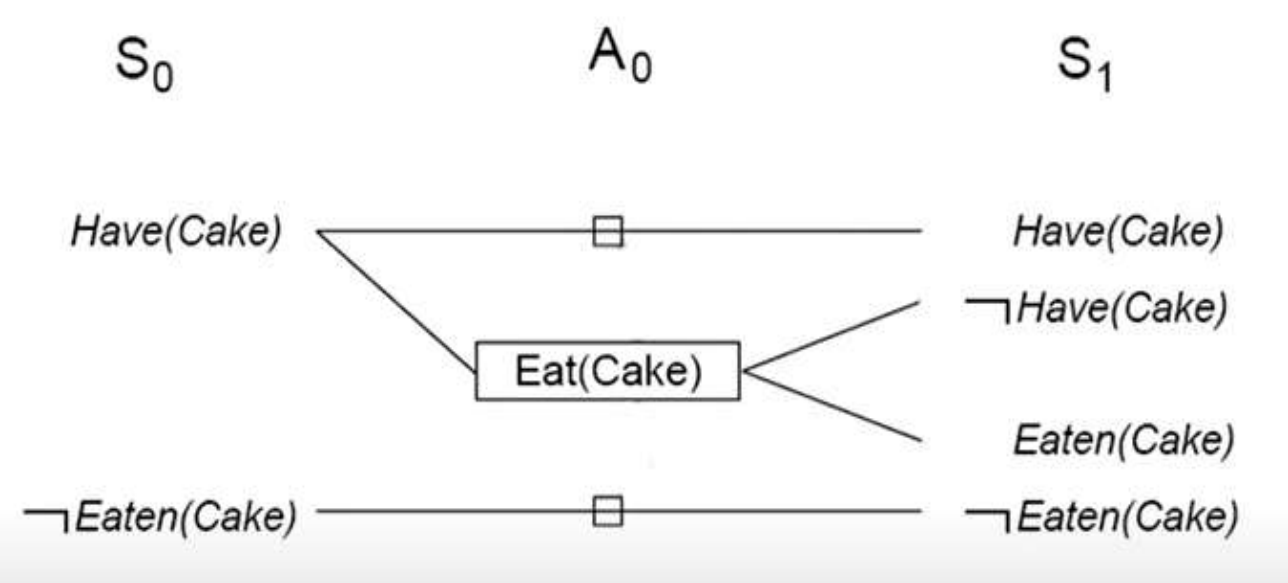


Add all applicable actions.
Add all effects to the next state.

Planning Graph – Example (Contd..)

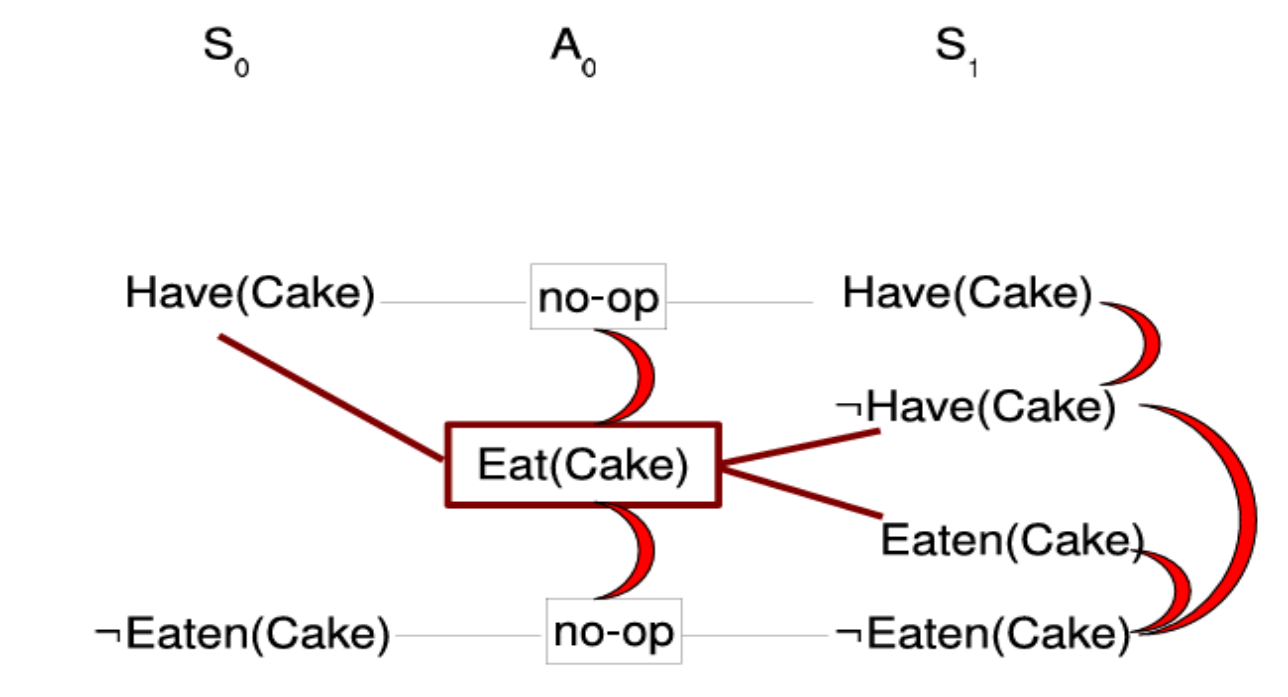
- In addition to ‘normal’ action, *persistence action or no-op* (no operation): one for each fluent, preserves the Fluent's truth.
- ***Mutex or mutual exclusion links*** depicted by red semicircles mean that ***actions cannot occur together.***
- Similarly, there are *mutex links between fluents(literals / effects).*
- *Build the graph until two consecutive levels are identical; then the graph levels off.*

Planning Graph – Example (Contd..)



Add persistence actions (inaction = no-ops) to map all literals in the state S_i to state S_{i+1}

Planning Graph – Example (Contd..)



Identify mutual exclusions between actions and literals based on potential conflicts.

Mutex between actions

Mutex relation holds between two actions at the same level if any of the following conditions holds:

- **Inconsistent effects**:
  - *Preconditions of one action conflict with the effects of another.*
  - **Example**: If Eat(Cake) requires Have(Cake), but another action negates it, they cannot co-occur.

    i.e.,

    ✓ you need to have the cake to eat it.

    ✓ If something else happens that makes you not have the cake, then eating it isn't possible.

    That is why these situations can’t happen at the same time.

Mutex between actions

- **Interference**:
  - *One action negates the effect of the other.*
  - Example: If the action Eat(Cake) has the effect ¬Have(Cake), it conflicts with a no-op action preserving Have(Cake).
    - ✓i.e., Eating the cake means you don't have it anymore.
    - ✓Doing nothing means you still have the cake.

    These two outcomes can't happen together because they are opposites.

Mutex between fluents (effects)

Mutex holds between fluents if:

- *One literal is the negation of the other.*
  - like Have(Cake) and ¬Have(Cake).
- *There are no actions that can simultaneously achieve both literals(inconsistent support).*
  - for example, Eaten(Cake) and ¬Eaten(Cake) cannot both be supported because achieving one invalidates the other.

Why Mutexes Are Important ?

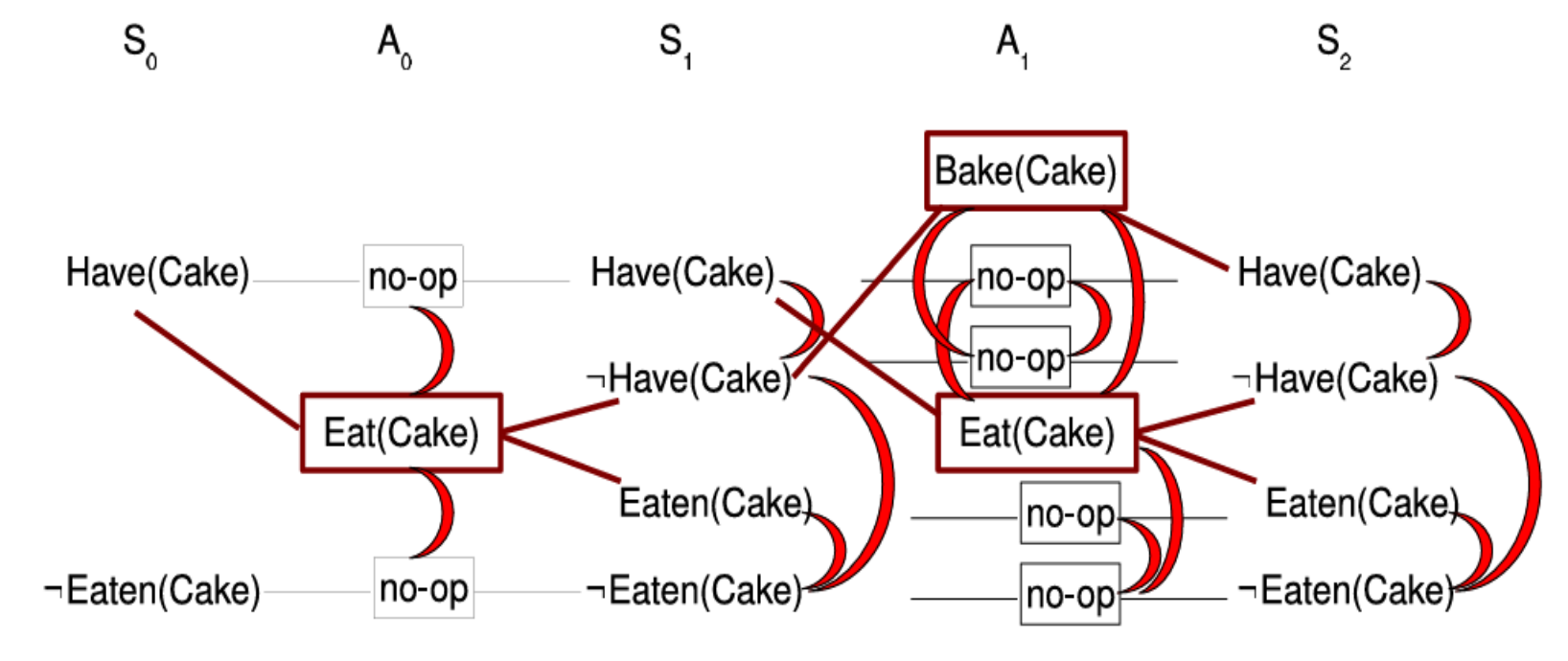
Mutex relationships simplify planning graphs by:

Pruning Infeasible Paths: They help to identify and eliminate incompatible actions or states early.

Focusing on Feasibility: They ensure that only logically consistent actions and states are considered during the planning process.

Improving Efficiency: By excluding invalid combinations, mutexes make graph-based planning computationally manageable.

Planning graph example



Repeat the process until graph levels off:

- Two consecutive levels are identical, or
- Contain the same number of literals.

References

- Stuart Russell and Peter Norvig, “Artificial Intelligence”, 2nd edition, Pearson Education, 2003.
- Saroj Koushik, “Artificial intelligence”.
- NPTEL

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Thank You

IARE
INSTITUTE OF AERONAUTICAL ENGINEERING

Course Title - Logic Programming for Artificial Intelligence

Topic Title – Planning and Acting in the real world

Presenter's Name – Ms. Bidyutlata Sahoo
Presenter's ID – IARE11028
Department Name – CSE (AI & ML)
Lecture Number - 04
Presentation Date – 28/11/2024

Course Outcome

At the end of the course, students should be able to:

CO5: Develop a comprehensive understanding of advanced AI planning strategies and diverse learning paradigms to solve complex problems.

Topic Learning Outcome

Understand the concept of planning and acting in the real world for making AI systems functional and effective in practical applications.

Planning and Acting in the real world

Planning and Acting in the real world

Planning and acting in the real world is a central problem in AI, as it *involves building systems* that can ***make decisions and execute actions under real-world constraints and uncertainty***.

Challenges in Real-World Planning

- **Uncertainty**: *The real world is filled with unpredictable factors*, such as incomplete or noisy sensor data, and uncontrollable external events.
- **Dynamic Environments**: *The environment can change while the agent is executing its plan*, requiring the ability to react and adapt.
- **Limited Time and Resources**: *Agents often have limited computational resources, energy, or time to make decisions* and act.
- **Partial Observability**: *An agent may not have access to complete information about the state of the world*, requiring it to infer or estimate the missing details.

Key Approaches to Real-World Planning

1. Conditional Planning
2. Execution Monitoring
3. Replanning
4. Hierarchical Planning
5. Interleaved Planning and Execution

1. Conditional Planning

- Conditional planning *is a way of creating a plan that prepares for different possible situations*. It *includes "if-then" branches to handle uncertainties or unexpected things* that might happen during the task.
- **For example**, imagine a robot navigating a building. Its goal is to get to a specific room. Along the way, it might face situations like:
  - ✓**If a door is open**, it goes through it.
  - ✓**If a door is locked**, it tries a key or looks for another route.
  - ✓**If the path is blocked**, it finds an alternative way.

2. Execution Monitoring

- Execution monitoring is like *keeping an eye on things while doing a task to make sure everything is going as planned. If something unexpected happens, you can fix it or change the plan.*
- **For example,** *think of a drone flying to deliver a package*. As it flies:
  - ✓**It checks if the wind is pushing** it off course.
  - ✓**If everything is fine**, it keeps going as planned.
  - ✓**If the wind is too strong**, it changes its path to stay on track**.**

3. Replanning

- *If the agent's original plan becomes impossible due to environmental changes or execution failures, it must be able to generate a new plan that adapts to the current state.*
- **Example**: *A self-driving car might replan its route if a road suddenly becomes blocked by an accident.*

4. Hierarchical Planning

- *Hierarchical planning decomposes a complex problem into smaller, more manageable sub-problems. This allows the agent to plan and act at different levels of abstraction, simplifying the overall process.*
- **Example**: *A robot might break down a high-level goal like "prepare dinner"* into sub-goals such as "*gather ingredients*" and "*cook meal*."

5. Interleaved Planning and Execution

- In dynamic environments, it can be beneficial to interleave planning and execution, *allowing the agent to begin acting while still refining or expanding its plan.*
- **Example**: *A robot might begin navigating to a goal location while still computing a more detailed plan for handling obstacles along the way.*

Acting in the Real World

1. Real-Time Decision Making

- Real-time decision-making means *an agent has to quickly decide what to do while it's doing the task, especially when time is limited or the situation changes fast.*
- **For example**, in autonomous driving:
  - ✓ The car needs to decide whether to slow down, stop, or change lanes based on traffic, pedestrians, or signals, all while driving at high speed.
  - ✓ The car can’t spend too much time thinking about each decision, or it might miss important events, but it also needs to make smart choices to stay safe.

Acting in the Real World

2. Continuous vs. Discrete Actions

- In real life, actions can be continuous or discrete.
  - ✓ Discrete actions are *like step-by-step moves with clear choices.* For example, a robot can either move forward, backward, left, or right—these are fixed options.
  - ✓ Continuous actions are *smoother and more flexible*, like controlling a robot's movement speed or steering angle, where the changes can be as small or as precise as needed.
- Planning for **continuous actions** is harder because there are endless possibilities to choose from, but it allows for smoother and more natural actions.

Acting in the Real World

3. Learning from the Environment

- *Agents often need to learn from their environment to improve their planning and acting capabilities over time.* This can be done through techniques such as **reinforcement learning**, where agents learn optimal actions *based on trial and error*.
- **For example**, a robot learning to walk might:
  - ✓ Try moving with one leg. If it stays balanced, it gets a reward.
  - ✓ If it falls, it learns to adjust and try a different movement next time.

Applications in the Real World

- **Robotics**: Robots must plan paths, manipulate objects, and interact with dynamic environments like homes or warehouses.
- **Autonomous Vehicles**: Self-driving cars plan routes, avoid obstacles, and handle uncertainties like weather, traffic, and road conditions.
- **Game AI**: In video games, agents must plan strategies and react to player actions in real time, often under incomplete information.
- **Healthcare**: AI planning is used in treatment scheduling, surgery planning, and managing medical resources.

References

- Stuart Russell and Peter Norvig, “Artificial Intelligence”, 2nd edition, Pearson Education, 2003.
- Saroj Koushik, “Artificial intelligence”.
- NPTEL

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Thank You

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Course Title - Logic Programming for Artificial Intelligence

Topic Title –Plan Generation Systems

Presenter's Name – Ms. Bidyutlata Sahoo
Presenter's ID – IARE11028
Department Name – CSE (AI & ML)
Lecture Number - 05
Presentation Date – 21/11/2024

Course Outcome

At the end of the course, students should be able to:

CO5: Develop a comprehensive understanding of advanced AI planning strategies and diverse learning paradigms to solve complex problems.

Topic Learning Outcome

Choose plan generation system for the auto-generation of detailed plan when manual planning is impractical to achieve specific goals.

Plan Generation Systems

Plan Generation Systems

- Plan generation systems in AI are *designed to create a sequence of actions or steps to achieve a specific goal or set of goals*.
- They *analyze the current state, goals, and available actions to produce a feasible plan* that an intelligent agent can execute, often *within constraints such as time, resources, or environmental uncertainties.*
- These systems ***are widely used in applications like robotics, logistics, game AI, and autonomous vehicles***.

Components of Plan Generation Systems

1. Initial State: *Represents the current state of the world or the situation from which planning begins*. This is usually based on sensor data, input parameters, or an environment model.

2. Goal State: *Defines the desired end state the agent aims to reach.* Goals can be singular or multiple and may involve constraints, such as minimizing time or resource usage.

3. Actions (Operators): *A set of actions the agent can perform to move from one state to another*. Each action has ***preconditions*** (conditions required to execute it) and ***effects*** (the results of performing the action).

Components of Plan Generation Systems

4. Planner (Algorithm): The core of the plan generation system, the planner selects and sequences actions that transform the initial state into the goal state. It *uses search strategies, heuristics, or logic-based techniques to find optimal or feasible plans.*

5. Constraints and Resources: *Constraints like time, energy, or spatial limits* restrict possible plans, while *resources, such as available energy or access to specific tools,* influence the agent's choice of actions.

6. State Space: *Represents all possible states* that can result from performing different sequences of actions. The planner navigates this space to find a path from the initial state to the goal state.

How Plan Generation Systems Work ?

Plan generation systems operate by analysing the current state and goals, then exploring possible actions and state transitions to create a sequence of actions. The system typically follows these **steps:**

- **Define the Problem and Constraints**: The planner *begins by defining the initial state, goal state, available actions*, and constraints (such as deadlines or resource limitations).
- **Search the State Space**: The *system uses a search algorithm* to explore the state space, considering possible states and actions to transit from one state to another. Each action applied to a state generates a new state, expanding the search space.

How Plan Generation Systems Work ?

- **Evaluate Feasibility**: The planner *checks that each action in a potential plan is feasible*, i.e., *it meets all preconditions and adheres to constraints.*
- **Select an Optimal or Satisfactory Plan**: Using a *cost function or heuristic*, the planner *evaluates different plans to select one that optimally balanced constraints, goals, and resources*, or at least meets minimum criteria for feasibility.
- **Return the Plan**: *Once a feasible plan is found, it is passed to the agent for execution*, often with built-in contingency steps if unexpected conditions arise.

Common Algorithms and Techniques in Plan Generation Systems

1. Heuristic Search Algorithms

- **A***: A* search uses a heuristic to estimate the cost to reach the goal from a given state, balancing the cost of reaching the current state and the estimated cost to reach the goal. It's widely used due to its efficiency in finding the shortest path.
- **Greedy Best-First Search**: This method expands nodes based on the heuristic alone, focusing on the most promising paths first but potentially ignoring longer-term costs, which may be less optimal.

Common Algorithms and Techniques in Plan Generation Systems (Contd..)

2. Graph-Based Algorithms

- **Planning Graphs**: Planning graphs represent states and actions in a layered, graphical format, with each layer containing possible actions or states. **GraphPlan** is a popular technique that uses these graphs to quickly find valid plans.
- **Constraint Satisfaction**: Some systems model the planning problem as a constraint satisfaction problem, using methods like backtracking or constraint propagation to explore feasible action sequences.

Common Algorithms and Techniques in Plan Generation Systems (Contd..)

3. Hierarchical Task Network (HTN) Planning

- **HTN Planning**: In HTN planning, the problem is decomposed into hierarchically structured tasks or subtasks. Each high-level task is broken down into more specific tasks until primitive actions (actions executable by the agent) are reached. This method allows for modular and efficient planning.
- **Example**: An HTN planner for a robot might break down the task of “making coffee” into steps like “get cup,” “brew coffee,” and “serve coffee,” each of which is further divided into actionable steps.

Common Algorithms and Techniques in Plan Generation Systems (Contd..)

4. Logic-Based Techniques

- **STRIPS (Stanford Research Institute Problem Solver)**: STRIPS represents actions in terms of preconditions and effects, allowing the planner to build a plan by sequentially applying actions to transit from the initial to the goal state.

Common Algorithms and Techniques in Plan Generation Systems (Contd..)

5. Reinforcement Learning (RL) Techniques

- **Q-Learning and Policy Gradient Methods**: For environments that require agents to learn from interactions, RL techniques can be used to generate plans by rewarding successful actions and penalizing failures. Over time, the system learns an optimal policy (plan) by maximizing cumulative rewards.

Plan Generation System in Action - Example

Imagine a ***robot in a warehouse tasked with moving items to designated locations***. The robot's plan generation system would:

I. Define the initial position of items (initial state) and where they need to go (goal state).

II. Use sensors to gather real-time information about obstacles or new tasks that arise.

III. Apply a search algorithm (e.g., A*) to navigate through possible actions like "move forward," "turn left," and "pick up item."

IV. Evaluate each action sequence for feasibility, updating the plan if obstacles are detected or if priorities change.

V. Execute the plan while continuing to monitor for any unexpected changes, adapting as needed.

References

- Stuart Russell and Peter Norvig, “Artificial Intelligence”, 2nd edition, Pearson Education, 2003.
- Saroj Koushik, “Artificial intelligence”.
- NPTEL

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Thank You

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Course Title - Logic Programming for Artificial Intelligence

Topic Title –Learning and its Types

Presenter's Name – Ms. Bidyutlata Sahoo
Presenter's ID – IARE11028
Department Name – CSE (AI & ML)
Lecture Number - 06
Presentation Date – 26/11/2024

Course Outcome

At the end of the course, students should be able to:

CO5: Develop a comprehensive understanding of advanced AI planning strategies and diverse learning paradigms to solve complex problems.

Topic Learning Outcome

Compare different types of learning suited for different problem domains.

Introduction to Learning

Introduction to Learning

- Denotes *changes in the system* that *enable a system to do the same task more efficiently next time.*
- Learning is *constructing or modifying representations* of what is being experienced.
- Learning *is making useful changes in our minds.*
- Learning *improves understanding and efficiency.*
- *Discover new things or structures which were previously unknown* (data mining, scientific discovery)

Introduction to Learning

- **Learning** is fundamental in AI because ***it enables systems to*** *improve their performance on a given task over time, based on data, experience, or interactions with the environment. Learning enables AI models to generalize patterns, make predictions, and adapt to new situations* ***without being explicitly programmed for each specific task.***

Learning Systems

- Machine learning systems perform the following iteratively:
  - ***Produce a result***
  - ***Evaluate it against expected result***
  - ***Tweak a system***
- Machine learning systems *also discover patterns* without prior expected results.

Why Learning is Crucial in AI?

- Improves Performance Over Time
- Adaptability to Changing Environments
- Reduces Need for Manual Programming
- Enables Personalization
- Handles Complex and Unstructured Data
- Supports Decision-Making and Predictions
- Enhances Automation of Routine Tasks
- Drives Innovation Across Industries

Learner Architecture

Learning systems has four main components:

- ✓**Knowledge Base (KB):**
  - What is being learnt
  - Representation of domain
  - Description and representation of problem space
- ✓**Performer:** does something with the knowledge base to produce results.
- ✓**Critic:** evaluates results produced against expected results.
- ✓**Learner:** takes output from the critic and modifies something in the KB or the performer.

Types of Learning based on the availability of data and the learning feedback

Based on **the type and availability of data and the learning feedback provided to the model, *learning is divided into 4 types:***

1. ***Supervised Learning***
2. ***Unsupervised Learning***
3. ***Semi-Supervised Learning***
4. ***Reinforcement Learning***

1. Supervised Learning

- In supervised learning, the *data is labelled*, *meaning each input has a corresponding desired output* (label).
- The *model learns by making predictions* ***and adjusting*** based on the labelled examples, improving its accuracy over time.
- This is particularly *useful when there is a large dataset with known outputs*, like classifying images or predicting language translations.

2. Unsupervised Learning

- Unsupervised learning *deals with unlabelled data*, where the *model does not have any predefined output labels*. Instead, it *looks for patterns, clusters, or structures* within the data on its own.
- This approach is beneficial when we want to explore the data, group similar items, or reduce data dimensions without needing specific output labels.

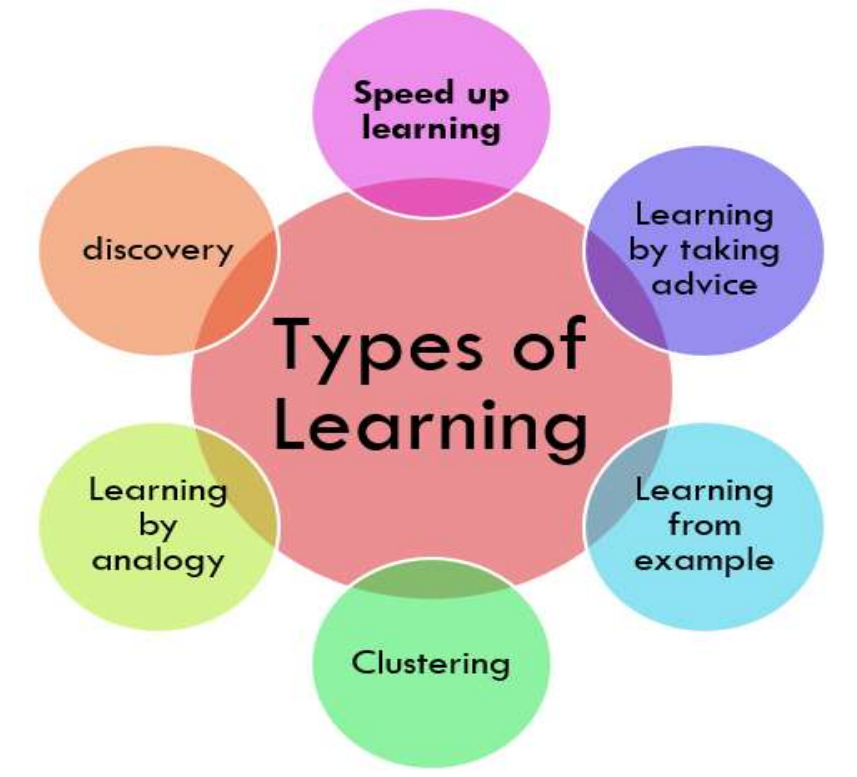
3. Semi supervised Learning

- Semi-supervised learning is a *combination of supervised and unsupervised approaches.*
- It *involves a small amount of labelled data and a larger amount of unlabelled data*.
- The labelled data helps guide the learning, while the unlabelled data provides additional information, allowing the model to generalize better.
- **Example:** *Image Classification*: Training on a few labelled images and many unlabelled images to classify objects in photos.

4. Reinforcement Learning

- Unlike the other three, reinforcement learning is *based on a feedback mechanism* that *rewards or penalizes the agent based on its actions*, rather than providing explicit labels.
- The *agent learns to maximize cumulative rewards over time by interacting with its environment.*
- This approach is *suitable for scenarios where learning is based on actions and outcomes*, such as in robotics or game AI.

Types of learning commonly studied and applied in artificial intelligence



1. Speedup Learning

- A type of ***deductive learning*** that *requires no additional input, but improves the agent's performance over time.*
- ***Speed Up Learning*** refers to techniques and strategies *used to improve the efficiency and speed of the learning process in AI, enabling models to learn faster and with fewer resources.*
- The **goal** is *to reduce training time, computational cost, or data requirements while maintaining or improving model accuracy*. This is *particularly important in fields where large datasets or complex computations are involved, such as deep learning* and *reinforcement learning*.

2. Learning by taking advice

- This type *involves using guidance or advice from human experts or other sources* to improve an AI system's performance.
- *It is akin to supervised learning*, where labelled examples or feedback are provided, helping the AI learn desired behaviours or patterns more effectively.
- **McCarthy** proposed the "*advice taker*" which was such a system, and TEIRESIAS [Davis, 1976] was the first such system.

3. Learning from example

- Often *known as supervised learning*, this is one of the most common types of learning in AI.
- The *model is trained on labelled data* (examples with correct outputs) *and learns to predict outputs for new inputs*.
- **Examples** include *image classification, language translation, and Email Spam Classification.*

4. Clustering

- This *represents unsupervised learning*, where an *AI system groups data points based on similarity without predefined labels*.
- Clustering is widely *used in data analysis* and *pattern recognition*, such as market segmentation or image grouping.
- **Scenario:** *Customer Segmentation for Marketing*

5. Learning by Analogy

- **Analogy** means *transfer of knowledge.*
- Analogy-based learning in AI involves *applying knowledge or patterns learned from one domain to a similar domain.*
- This can help AI systems solve problems in new but related situations by *drawing analogies with past experiences.*
- **Scenario:** Imagine you're *teaching a child how to ride a bicycle.* The child has never ridden a bike before, but they are already proficient in riding a scooter.

6. Discovery

- ***Both inductive and deductive learning*** in which an agent learns without help from a teacher.
- Discovery learning in AI is about *exploring and identifying new patterns, relationships, or structures within data.*
- It is often *applied in data mining and exploratory analysis*, where the AI autonomously finds new insights or hidden patterns in large datasets.
- **Example** : In the field of ***drug discovery***, AI systems are increasingly being used to identify new drug compounds that could potentially treat diseases.

References

- Stuart Russell and Peter Norvig, “Artificial Intelligence”, 2nd edition, Pearson Education, 2003.
- Saroj Koushik, “Artificial intelligence”.
- NPTEL

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Thank You

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Course Title - Logic Programming for Artificial Intelligence

Topic Title – Discovery, Clustering and Analogy

Presenter's Name – Ms. Bidyutlata Sahoo
Presenter's ID – IARE11028
Department Name – CSE (AI & ML)
Lecture Number - 07
Presentation Date – 26/11/2024

Course Outcome

At the end of the course, students should be able to:

CO5: Develop a comprehensive understanding of advanced AI planning strategies and diverse learning paradigms to solve complex problems.

Topic Learning Outcome

1. **Make use of** discovery techniques to develop more powerful and efficient AI systems.
2. **Define** clustering and differentiate it from other machine learning tasks to segregate the data points based on their similarity and dissimilarity.
3. **Apply** analogical reasoning to solve new problems by relating them to previously solved problems.

Discovery

Discovery

- Learning is the process by which one entity acquires knowledge.
- Discovery is a *restricted form of learning* in which *one entity acquires knowledge without the help of a teacher.*
- There are three types of automated discovery systems like:
    1. ***AM: Theory-Driven Discovery***
    2. ***BACON: Data-Driven Discovery***
    3. ***Clustering***

Theory Driven Discovery - AM (1976)

- AM is a program that *discovers concepts in elementary mathematics and set theory.*
- AM has 2 inputs:
  - *A description of some concepts of set theory* (in LISP form). E.g. set union, intersection, the empty set.
  - *Information on how to perform mathematics*. E.g. functions.

How does AM work?

AM *employs many general-purpose AI techniques:*

- A frame-based representation of mathematical concepts.
- AM can create new concepts (slots) and fill in their values.
- Heuristic search employed
  - ✓250 heuristics represent hints about activities that might lead to interesting discoveries.
  - ✓How to employ functions, create new concepts, generalisation etc.
- Hypothesis and test-based search.
- Agenda control of discovery process.

Learning by Discovery

AM discovered:

- **Integers**-- it is possible to count the elements of this set and this is the image of this counting function -- the integers -- interesting set in its own right.
- **Addition**-- The union of two disjoint sets and their counting function.
- **Multiplication**-- Having discovered addition and multiplication as laborious set-theoretic operations more effective descriptions were supplied by hand.
- **Prime Numbers--** factorisation of numbers and numbers with only one factor were discovered.

Learning by Discovery

- **Golbach's Conjecture--** Even numbers can be written as the sum of 2 primes. E.g. 28 = 17 + 11.
- **Maximally Divisible Numbers--** numbers with as many factors as possible. A number k is maximally divisible is k has more factors than any integer less than k. E.g. 12 has six divisors 1,2,3,4,6,12

Data Driven Discovery -- BACON (1981)

- Many discoveries are made from *observing data obtained from the world and making sense of it .*
- E.g. *Astrophysics* - discovery of planets, *Quantum mechanics* - discovery of sub-atomic particles.
- BACON is an attempt at providing such an AI system.

Data Driven Discovery -- BACON (1981)

BACON system features:

- Unlike theory-driven systems like AM, *BACON worked primarily with empirical data.* It discovered relationships (e.g., linear, proportional) by analyzing numerical datasets and identifying patterns.
- BACON was able to *rediscover famous scientific laws*, such as: Ohm's Law, **Kepler's Laws** of planetary motion, Boyle's Law.
- BACON *used heuristics* (simple decision rules) to guide its search for patterns and hypotheses in data.
- It *modelled the inductive reasoning process* scientists use to infer general laws from specific observations.

Clustering

Clustering

- Clustering involves *grouping data into several new classes.*
- It is a common descriptive task where one seeks to identify a finite set of categories or clusters to describe the data. For example, we may want to cluster houses to find distribution patterns.
- Clustering is the process of grouping a set of physical or abstract objects into classes of similar objects.
- *A cluster is a collection of data objects that are similar to one another within the same cluster* and *are dissimilar to the objects in other clusters.* Clustering analysis helps to construct meaningful partitioning of a large set of objects.

Clustering - Applications

- The task of clustering is *to maximize the intra-class similarity and minimize the interclass similarity.*
- **Customer Segmentation**: Group customers based on purchasing behaviour.
- **Image Segmentation**: Identify and separate different objects or regions in an image.
- **Document Clustering**: Organize text documents by topics.
- **Anomaly Detection**: Identify outliers in datasets for fraud detection or fault diagnosis.
- **Biology and Medicine**: Group genes or cells with similar characteristics in genomics or medical research.

Clustering – Types

K-Means Clustering:

- Divides data into k predefined clusters by minimizing intra-cluster variance.
- Fast and simple but sensitive to the number of clusters (k) and initial centroids.

Hierarchical Clustering:

- Builds a tree-like structure (dendrogram) of clusters.
- Two types:
  - ✓ **Agglomerative**: Bottom-up merging of clusters.
  - ✓ **Divisive**: Top-down splitting of clusters.

Clustering – Types

DBSCAN (Density-Based Spatial Clustering of Applications with Noise):

- Groups points based on density, identifying outliers as noise.
- Suitable for datasets with arbitrary shapes and noise.

K-Modes Clustering

- **K-Modes Clustering** is an extension of the K-Means algorithm specifically designed for clustering **categorical data**.
- Unlike K-Means, which relies on mean values and Euclidean distance, K-Modes uses **mode** (the most frequent category) and a distance measure suitable for categorical variables.

Analogy

Analogy

- Analogy is a powerful ***inference tool.***
- A comparison between one thing and another, typically for the purpose of explanation or clarification.
- **Consider the sentences to understand how our language and reasoning are made with examples:**
  - *The analogy between the heart and a pump.*
  - *Bill is like a fire engine.*
  - *Last month, the stock market was a roller coaster.*

Learning by Analogy

- The form of learning *applies existing knowledge to a new problem instance on the basis of similarities between them.*
- It *involves developing a set of mappings between features of two instances*.
- **Example:** *solving problem in exam where previously learned similar examples serve as a guide.*

Learning by Analogy

- **Example:** *when we learn to drive truck using our knowledge of car driving.*

Analogical Reasoning Steps

- **Retrieve:**- Retrieve cases from memory that are relevant to solving it.
- **Reuse**:- Map the solution from previous case to the target problem. This involves adapting the solution to fit new solution.
- **Revise**:- Test the new solution to real world and, if necessary, revise.
- **Retain**:- After the solution has been successfully adapted to target problem, store the resulting experience as the new case in memory.

Learning by Analogy - Methods

- As *mapping between two dissimilar concepts* (e.g., mapping between heart and pump) *are difficult*, so there are *two ways to solve the problem.*
  - **Transformational and**
  - **Derivational**

Transformational Analogy

- Suppose you are asked to prove a theorem in plane geometry.
- You might *look for a previous theorem that is very similar and copy its proof, making substitutions when necessary.*
- The **idea** is *to transform a solution to a previous problem in to solution for the current problem.*
- Transformational Analogy *does not look at how problem was solved. It only look at the final solution*.

Fig: Transformational analogy

Derivational Analogy

- In this *it replays a previous derivation and modify if it is necessary.*
- *If initial assumption and steps are same then copy it otherwise alternative needs to be found.*
- The detailed history of problem-solving episode is called derivation.
- Analogical reasoning that takes these histories into account is called derivational analogy.

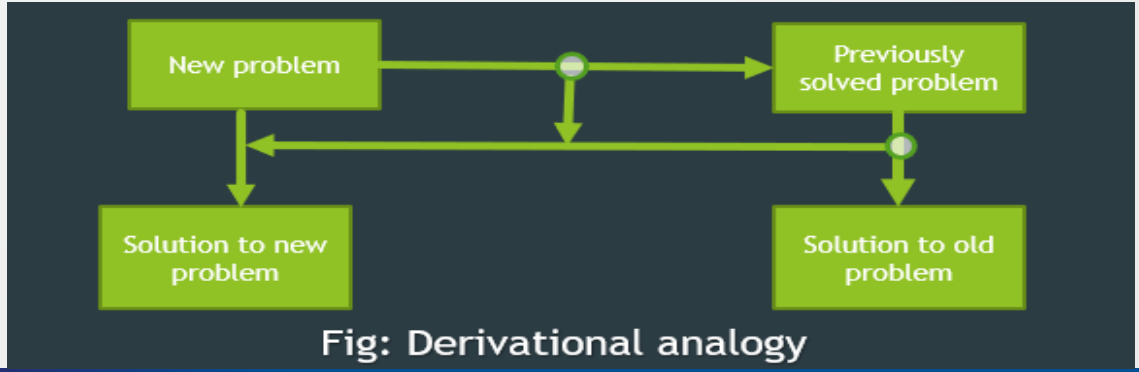


Fig: Derivational analogy

References

- Stuart Russell and Peter Norvig, “Artificial Intelligence”, 2nd edition, Pearson Education, 2003.
- Saroj Koushik, “Artificial intelligence”.
- NPTEL

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Thank You

IARE
INSTITUTE OF AERONAUTICAL ENGINEERING

Course Title - Logic Programming for Artificial Intelligence

Topic Title – Neural Networks and Genetic Learning

Presenter's Name – Ms. Bidyutlata Sahoo
Presenter's ID – IARE11028
Department Name – CSE (AI & ML)
Lecture Number - 08
Presentation Date – 28/11/2024

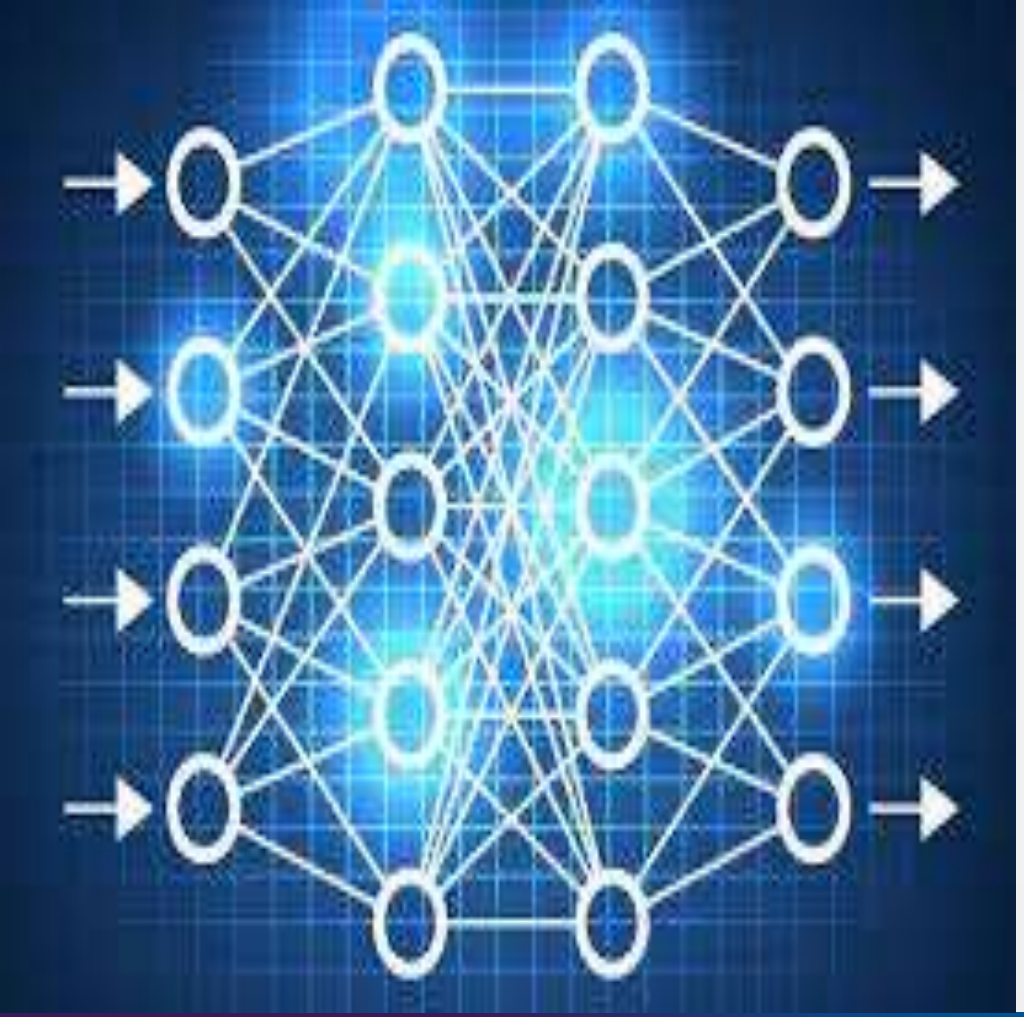
Course Outcome

At the end of the course, students should be able to:

CO5: Develop a comprehensive understanding of advanced AI planning strategies and diverse learning paradigms to solve complex problems.

Topic Learning Outcome

1. ***Identify and explain*** the role of various activation functions and their impact on the network's performance to solve complex problems.
2. ***Choose*** genetic algorithms and genetic programming systems to solve optimization and search problems.



Neural Networks

Neural Networks

- Neural networks are a *subset of machine learning*, ***inspired by the structure and function of the human brain.***
- They *consist of layers of interconnected nodes* (neurons) that can learn *to recognize patterns in data* and *make decisions*.

A Single Node In Neural Network

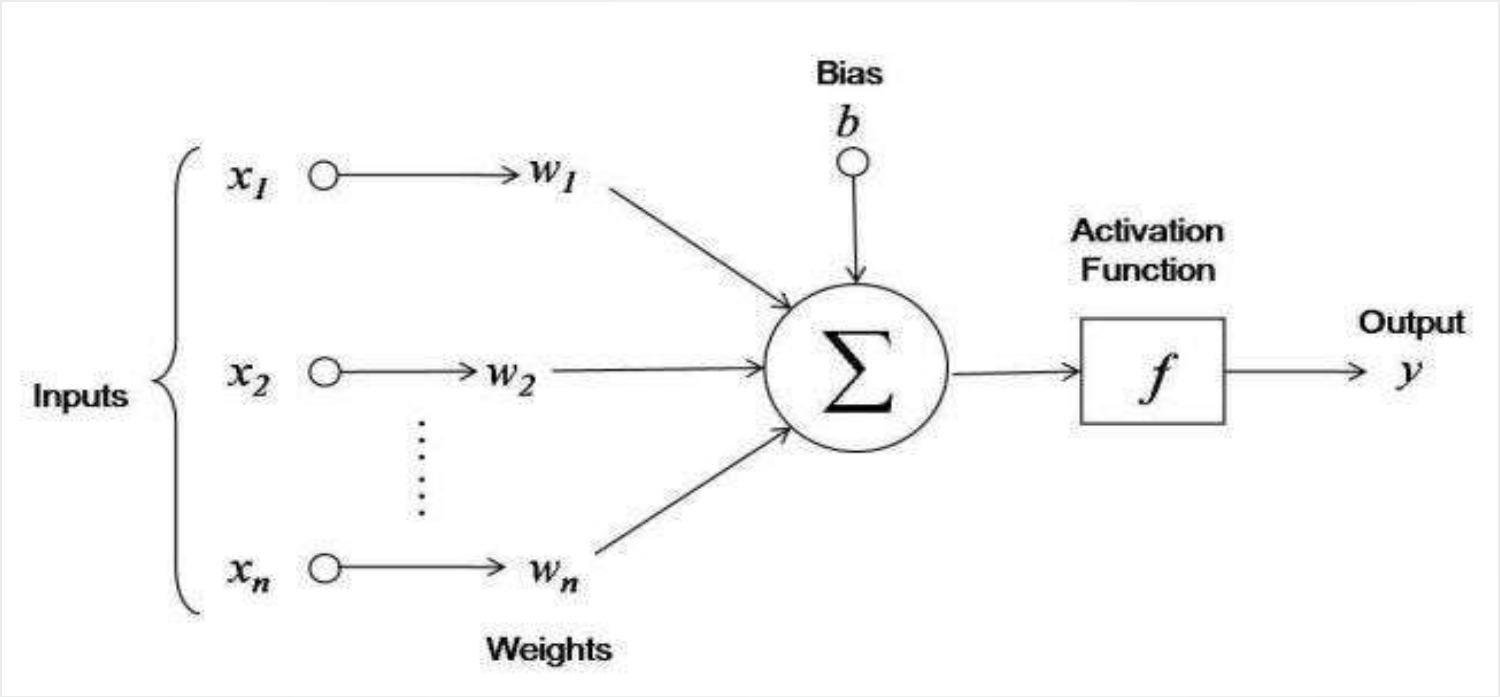
- Every *node in neural network perform 3 actions:*

✓ ***Takes input***

✓ ***Using some function, it process***

✓ ***Then generates the output***

A Single Node In Neural Network



(A Single Node in Neural Network)

A Single Node In Neural Network

- Neural networks *mimic the basic functioning of the human brain* and are inspired by how the human brain interprets information.
- They *solve various real-time tasks because of its ability to perform computations quickly and its fast responses.*
- Artificial Neural Network has a huge number of interconnected processing elements, also known as *Nodes.*
- These nodes are ***connected with other nodes using a connection link***. The *connection link contains weights*, ***these weights contain the information about the input signal***. Each iteration and input in turn leads to updation of these weights.

A Single Node In Neural Network

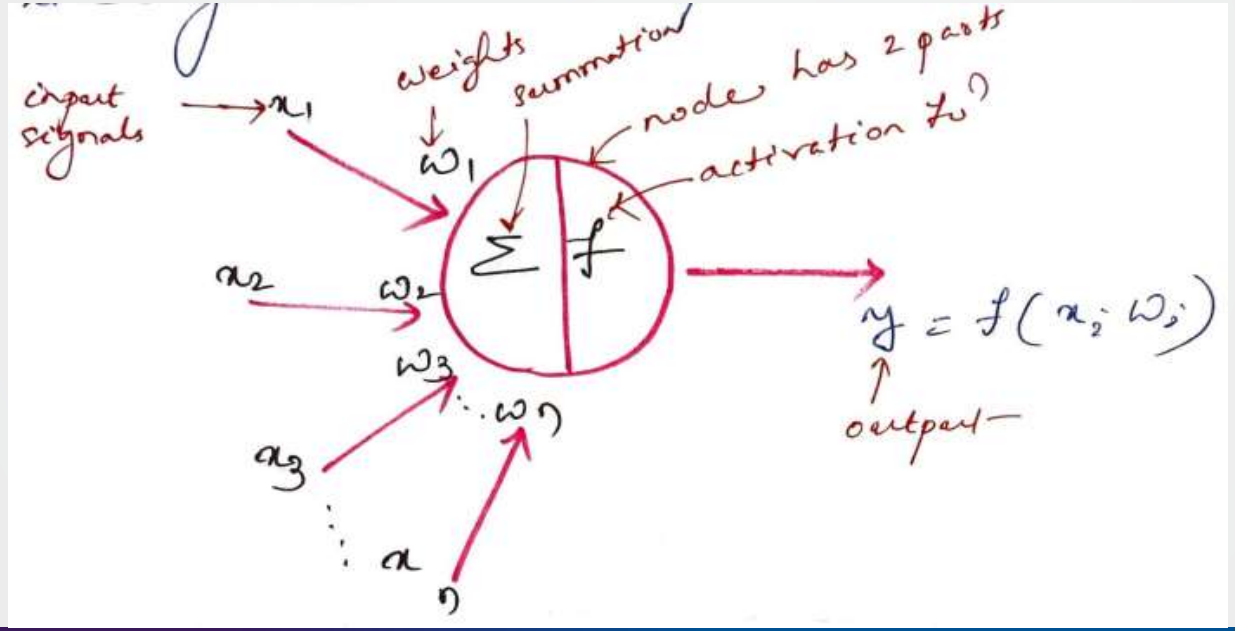
- *Types of tasks that can be solved using an artificial neural network includes:*
  - ***Classification problems***
  - ***Pattern Matching***
  - ***Data Clustering, etc.***

Basic Components Of Neural Networks

1. Neurons
2. Layers
3. Weights and Biases
4. Activation Function
5. Forward Propagation
6. Loss Function
7. Backpropagation

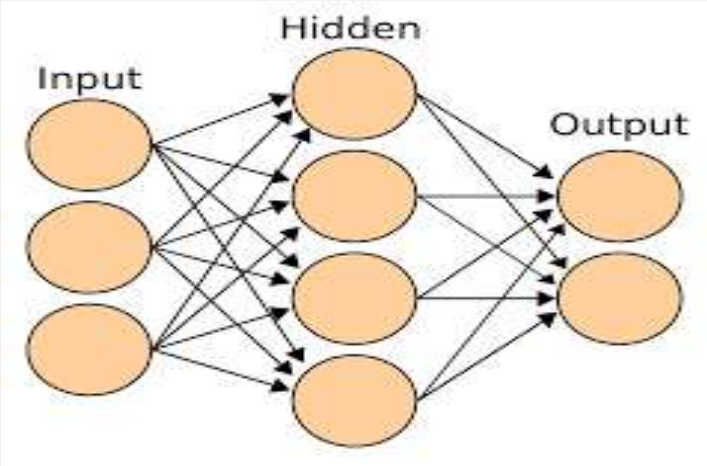
1. Neurons

- The *fundamental unit of a neural network.*
- *Takes inputs, processes them through an activation function, and produces an output.*



2. Layers

- *Input Layer:* The first layer that receives the input data.
- *Hidden Layers*: Intermediate layers that process inputs received from the previous layer. They can be one or more layers deep.
- *Output Layer*: The final layer that produces the output.



3. Weights and Biases

- *Weights:* Parameters that adjust the *input signal strength.*
- *Biases: Additional parameters added to the neuron's input* to *adjust the output along with the weights.*

4. Activation Function

- A *function applied to the weighted sum of inputs to introduce non-linearity into the model.*
- Common activation functions include *ReLU, Sigmoid, and Tanh*.

5. Forward Propagation

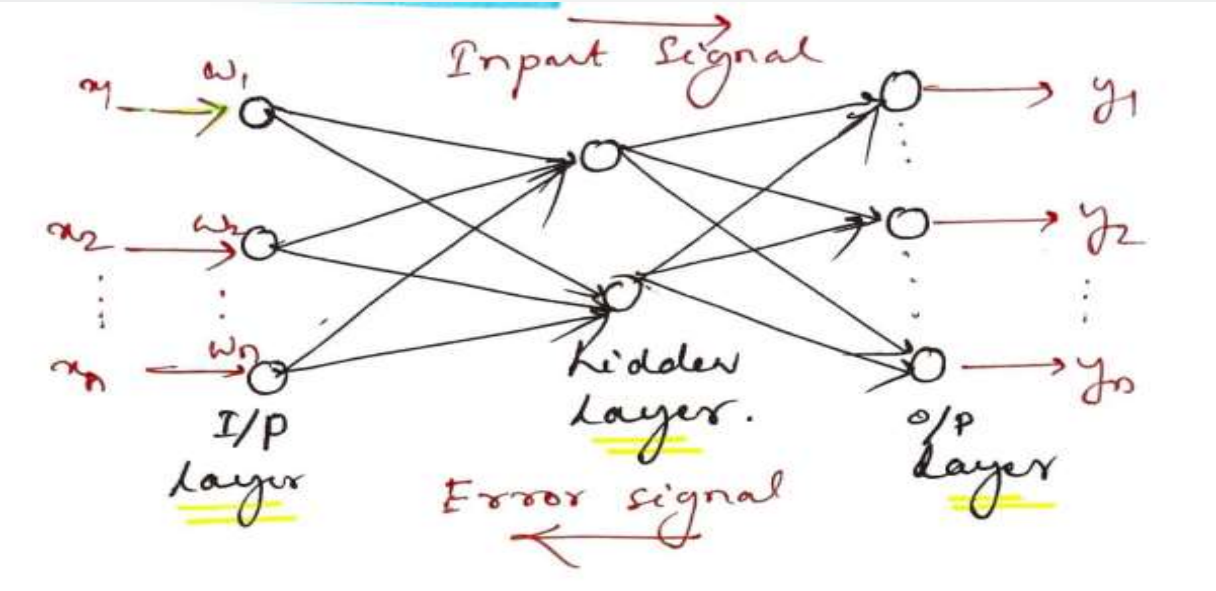
- The *process of passing inputs through the network to get the output.*

6. Loss Function

- A *function that measures the difference between the network's output and the actual target value.*
- Common loss functions include *Mean Squared Error (MSE) and Cross-Entropy Loss..*

7. Backpropagation

- The *process of updating the network's weights and biases to minimize the loss function.*
- Uses gradients to propagate the error back through the network



Advantages of Neural Networks

- *Adaptability*. They can learn and make independent decisions.
- *Parallel processing*. Large networks can process multiple inputs simultaneously.
- *Fault tolerance*. Even if a part of the network fails, the entire network can still function.

Limitations of Neural Networks

- *Data dependency:* They require a large amount of data to function effectively.
- *Opaque nature:* Often termed as "***black boxes***" because it's challenging to understand how they derive specific decisions.
- *Overfitting:* They can sometimes memorize data rather than learning from it.

Applications of Neural Networks

Neural networks are employed across various domains for:

- *Identifying objects, faces, and understanding spoken language* in applications like *self-driving cars and voice assistants.*
- *Analysing and understanding human language, enabling sentiment analysis, chatbots, language translation, and text generation.*
- *Diagnosing diseases from medical images, predicting patient outcomes, and drug discovery.*
- *Predicting stock prices, credit risk assessment, fraud detection, and algorithmic trading*.

Applications of Neural Networks

- *Personalizing content* and *recommendations in e-commerce*, streaming platforms, and social media.
- *Enhancing game AI, generating realistic graphics, and creating immersive virtual environments*.
- Monitoring and optimizing manufacturing processes, predictive maintenance, and quality control.
- Analysing complex datasets, simulating scientific phenomena, and aiding in research across disciplines.
- Generating music, art, and other creative content.

Genetic Learning

Genetic Learning

- ***Genetic learning,*** inspired by the principles of natural evolution, is a key concept in Artificial Intelligence (AI) ***that falls under the broader category of evolutionary algorithms***.
- A *genetic algorithm* is a *heuristic search* algorithm that is *inspired by Charles Darwin's theory of natural evolution.*
- This algorithm ***reflects the process of*** *natural selection* where the *fittest individuals are selected for reproduction in order to produce offspring of the next generation*.
- It is particularly *useful for solving optimization and search problems* where traditional algorithms might struggle.

Genetic Learning

- Nature has always been a great source of inspiration to all mankind. Genetic Algorithms (GAs) are *search based algorithms based on the concepts of natural selection and genetics.*
- GAs are a subset of a much larger branch of computation.

What Are Evolutionary Algorithms?

- **Evolutionary Algorithms (EAs)** are a *subset of artificial intelligence* techniques *inspired by the process of natural selection and biological evolution.*
- These algorithms are *used to solve complex optimization and search problems* by *mimicking the principles of evolution*, such as *reproduction, mutation, recombination, and selection*.

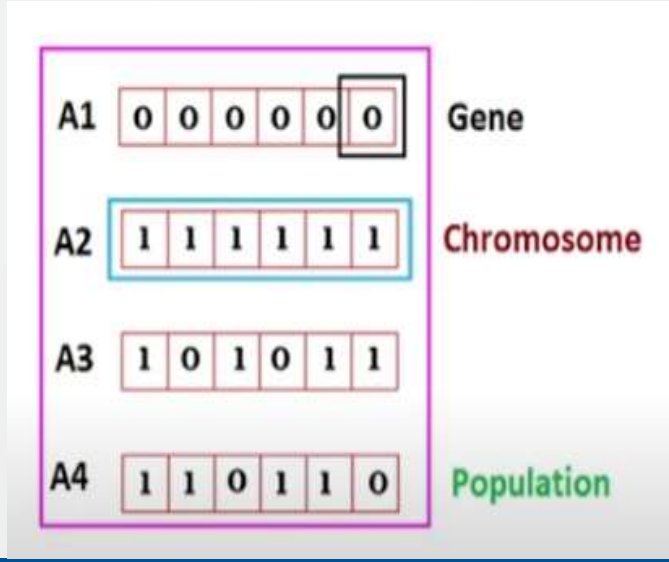
How genetic learning works?

- Here's a step-by-step explanation of how genetic learning works:

1. ***Initialization***
2. ***Evaluation (Fitness Assignment)***
3. ***Selection***
4. ***Crossover (Recombination / Reproduction)***
5. ***Mutation***
6. ***Termination***

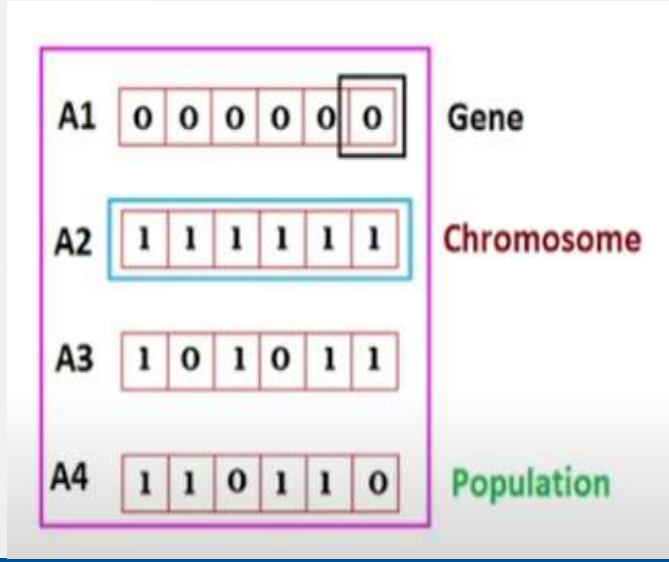
1. Initialization

- The process begins with a *set of individuals* which is called a *population.*
- *Each individual is a solution to the problem* you want to solve known as *Chromosome.*
- *An individual is characterized by a set of parameters* (variables) known as *Genes.*
- *Genes are joined into a string* to form a *Chromosome.*



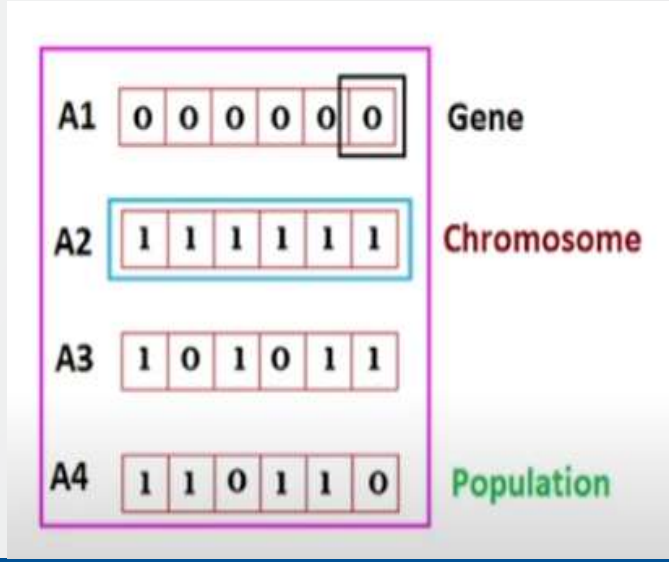
2. Evaluation (Fitness Assignment)

- Each individual in the population is evaluated using a *fitness function* that *measures how well it solves the given problem*. (i.e., *the ability of an individual to compete with other individuals*.)
- It gives a fitness score to each individual.
- *The probability that an individual will be selected for* reproduction is based on its *fitness score.*



3. Selection

- The **idea** of selection phase is *to select the fittest individuals and let them pass their genes to the next generation. (which are called as parents)*
- *Two pairs of individuals (parents) are selected based on their highest fitness scores.*



3. Selection

Various **selection methods** are used, such as:

- Roulette Wheel Selection: Individuals are selected based on their proportion of the total fitness.
- Tournament Selection: Randomly selects a group of individuals and picks the best among them.
- Rank Selection: Individuals are ranked by fitness, and selection is based on their rank.

Example:

If an individual has a fitness score of 90 and another has 70, the one with a score of 90 has a higher chance of being selected for reproduction.

Roulette Wheel Selection – Working process

Individuals are selected based on their proportion of the total fitness.

- ✓Each *individual is assigned a probability based on its fitness relative to the total fitness of the population..*
- ✓The probabilities are visualized as segments on a roulette wheel.
- ✓A *random number is generated, and the individual corresponding to that segment is selected.*

Roulette Wheel Selection - Example

Individual	Fitness	Proportion of Total Fitness
A	10	10 / 50 = 0.20 (20%)
B	15	15 / 50 = 0.30 (30%)
C	5	5 / 50 = 0.10 (10%)
D	20	20 / 50 = 0.40 (40%)

Total Fitness = 10 + 15 + 5 + 20 = 50

The roulette wheel is divided into segments: A (20%), B (30%), C (10%), D (40%).

The algorithm randomly selects a point on the wheel. If the random number falls within the first 20%, Individual A is selected. If it falls between 20% and 50%, Individual B is chosen, and so on.

Tournament Selection – Working process

Involves selecting a group of individuals at random and then picking the best one among them.

- ✓ Choose a small subset (usually 2 to 4) of individuals randomly from the population.
- ✓ Compare their fitness values.
- ✓ The *individual with the highest fitness in this subset is selected for reproduction.*

Tournament Selection – Example

Individual	Fitness
A	10
B	15
C	5
D	20

Suppose we randomly select a subset of 2 individuals: B and D. Among them, D has the highest fitness (20), so D is selected.

Rank Selection – Working process

Individuals are ranked by fitness, and selection is based on their rank.

- ✓Sort the population by fitness from highest to lowest.
- ✓Assign ranks to individuals (e.g., rank 1 for the highest fitness).
- ✓The *probability of selection is based on the individual's rank* rather than the fitness score.
- ✓Often, a linear or exponential ranking system is used to assign probabilities.

Rank Selection – Example

Individual	Fitness	Rank	Selection Probability (based on rank)
A	20	1	4/10 = 0.40 (40%)
B	15	2	3/10 = 0.30 (30%)
C	10	3	2/10 = 0.20 (20%)
D	5	4	1/10 = 0.10 (10%)

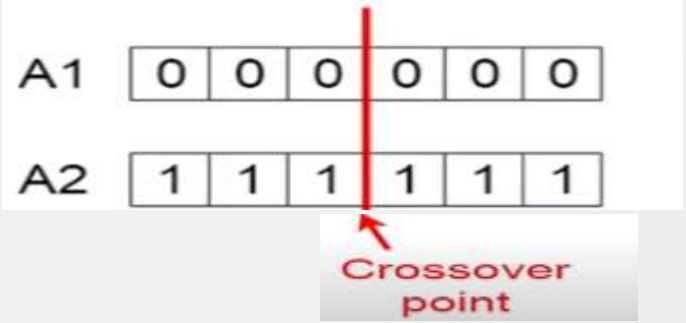
Total Ranks = 1 + 2 + 3 + 4 = 10

Individuals are selected based on their rank:

- Individual A has a 40% chance of being selected.
- Individual B has a 30% chance, and so on.

4. Crossover (Recombination / Reproduction)

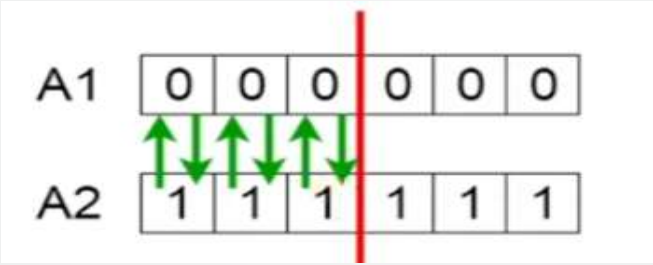
- Crossover is the most significant phase in a genetic algorithm.
- *Two parent individuals are selected to create one or more* ***offspring*** *through a process called crossover.*
- The *crossover point is chosen randomly*, and the *genetic material* (*bits or values*) from each parent is combined to produce new individuals.
- Example: Consider the crossover point to be 3 here.



4. Crossover (Recombination / Reproduction)

Offspring:

- *Offsprings are created* by *exchanging the genes of parents among themselves* until the crossover point is reached.



- The new offspring are added to the population.

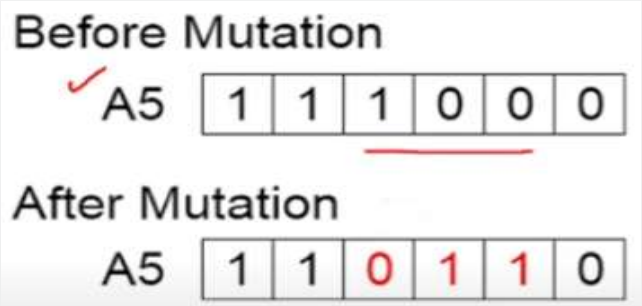
4. Crossover (Recombination / Reproduction)

Types of Crossover:

- Single-point Crossover: A single crossover point is selected, and the genetic material is swapped between parents at that point.
- Two-point Crossover: Two points are chosen, and the middle segment is swapped.
- Uniform Crossover: Each gene from the parent has an equal chance of being passed to the offspring.

5. Mutation

- *To maintain diversity within the population and avoid premature convergence*, a **mutation** is applied randomly to some of the offspring.
- This *involves flipping bits* (in binary representation) or *making small changes* (in real-number representation).
- The *mutation rate is typically low* (e.g., 1% to 5%), meaning only a *few genes are altered per individual*.

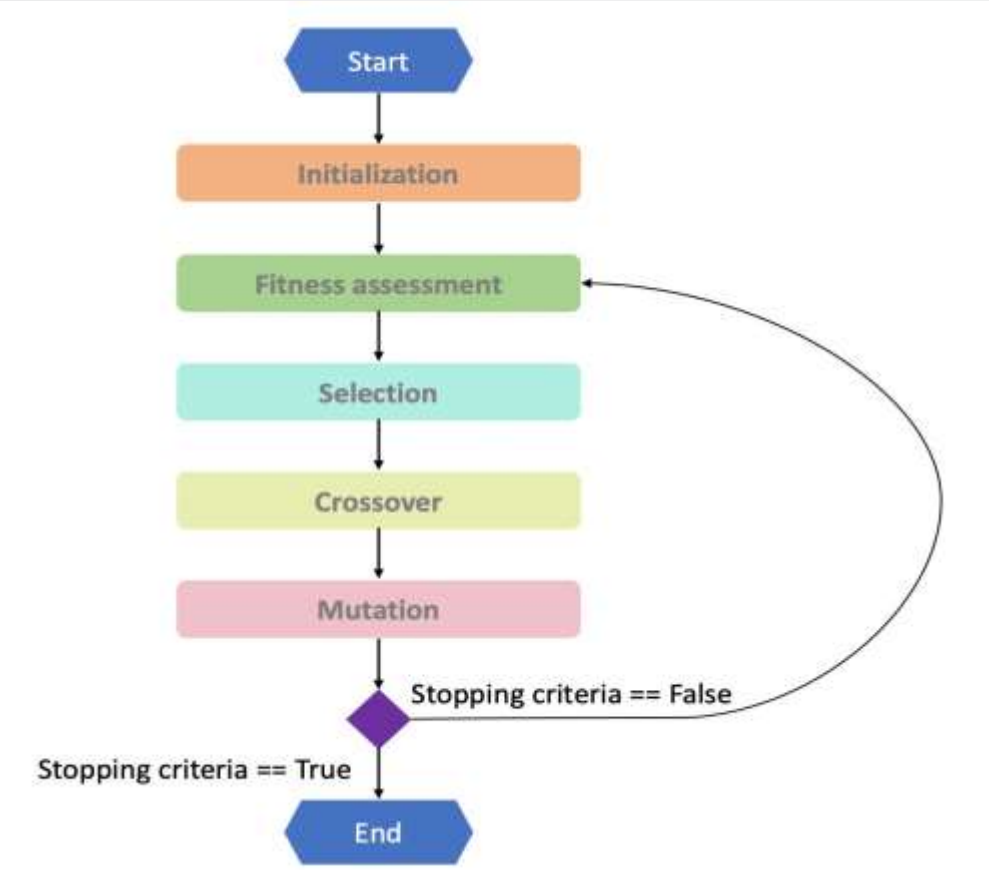


6. Termination

Stopping Criteria:

- The algorithm repeats the cycle of evaluation, selection, crossover and mutation for multiple generations.
- The process continues until a stopping criterion is met, such as:
  - *A maximum number of generations.*
  - *A satisfactory fitness level.*
  - *No improvement in the best solution over a specified number of generations.*
- That means *the algorithm terminates if the population has converged* (does not produce offspring which are significantly different from previous generation.)

Flow Chart for Genetic Algorithm



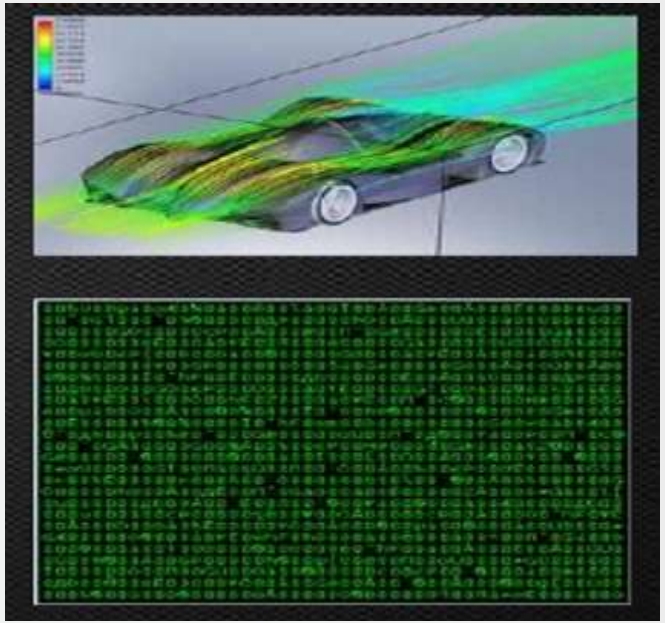
Advantages of Genetic Learning

Here's a step-by-step explanation of how genetic learning works:

- **Exploration and Exploitation**: Balances exploring new solutions (diversity) and exploiting the best-known solutions (intensification).
- **No Gradient Needed**: Does not require gradient information, making it suitable for non-differentiable and complex problem spaces.
- **Robust and Flexible**: Can *handle diverse and noisy search spaces*, making it useful for real-world applications.

Applications of Genetic Learning

- Travelling Salesman Problem
- Artificial Life(A-Life)
- Robotics
- Automotive Design
- Evolvable Hardware
- Computer Gaming
- Encryption and Code Breaking
- Optimizing Chemical Kinetic Analysis



Why we use Genetic Algorithms ?

- They are ***Robust.***
- ***Provide optimization over large space state.***
- Unlike traditional AI, ***they do not break on slight change in input or presence of noise.***

References

- Stuart Russell and Peter Norvig, “Artificial Intelligence”, 2nd edition, Pearson Education, 2003.
- Saroj Koushik, “Artificial intelligence”.
- NPTEL

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Thank You

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Course Title - Logic Programming for Artificial Intelligence

Topic Title –Reinforcement Learning

Presenter's Name – Ms. Bidyutlata Sahoo
Presenter's ID – IARE11028
Department Name – CSE (AI & ML)
Lecture Number - 09
Presentation Date – 28/11/2024

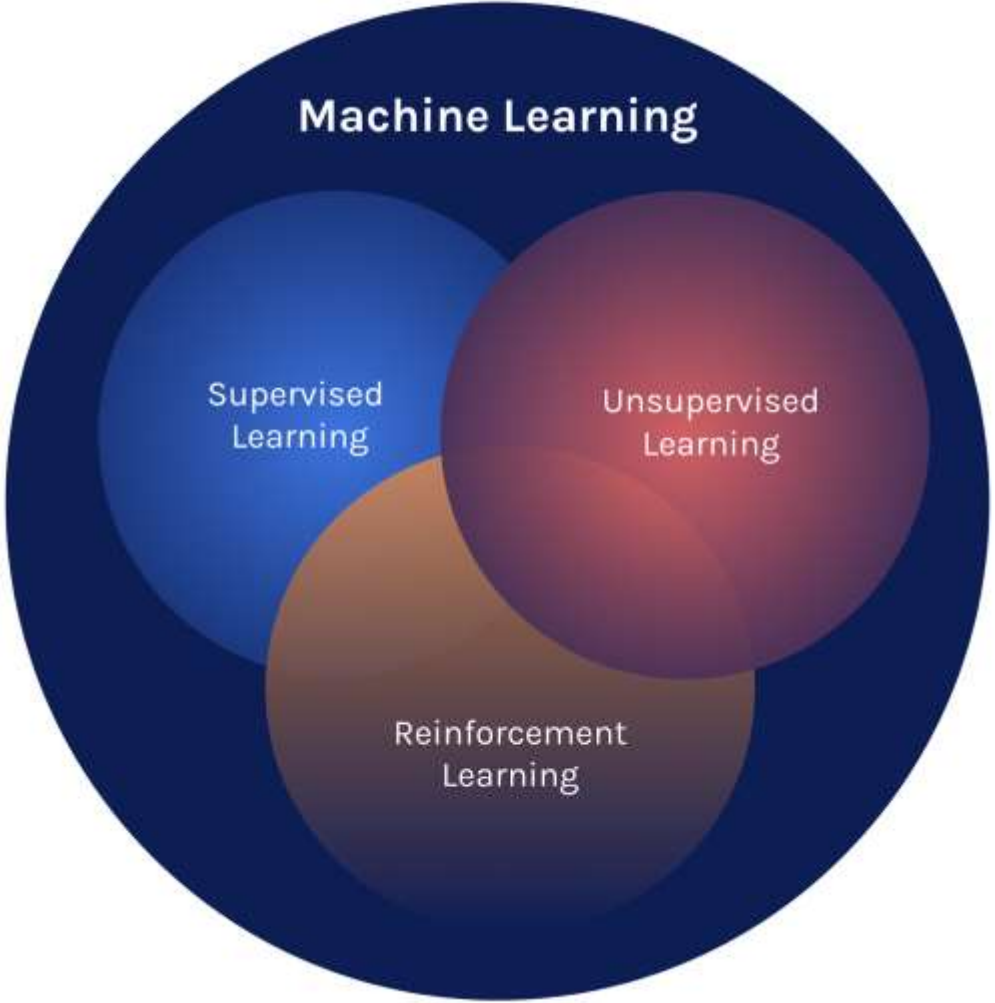
Course Outcome

At the end of the course, students should be able to:

CO5: Develop a comprehensive understanding of advanced AI planning strategies and diverse learning paradigms to solve complex problems.

Topic Learning Outcome

Explain the core concepts of reinforcement learning for decision making problems.



Reinforcement Learning

Machine Learning

- *Machine learning* is a scientific discipline that is *concerned with the design and development of algorithms* that allow computers to learn based on data, such as from sensor data or databases.
- A major *focus of machine learning* research is *to automatically learn to recognize complex patterns and make intelligent decisions based on data .*

Machine Learning Types

With respect to the feedback type to learner:

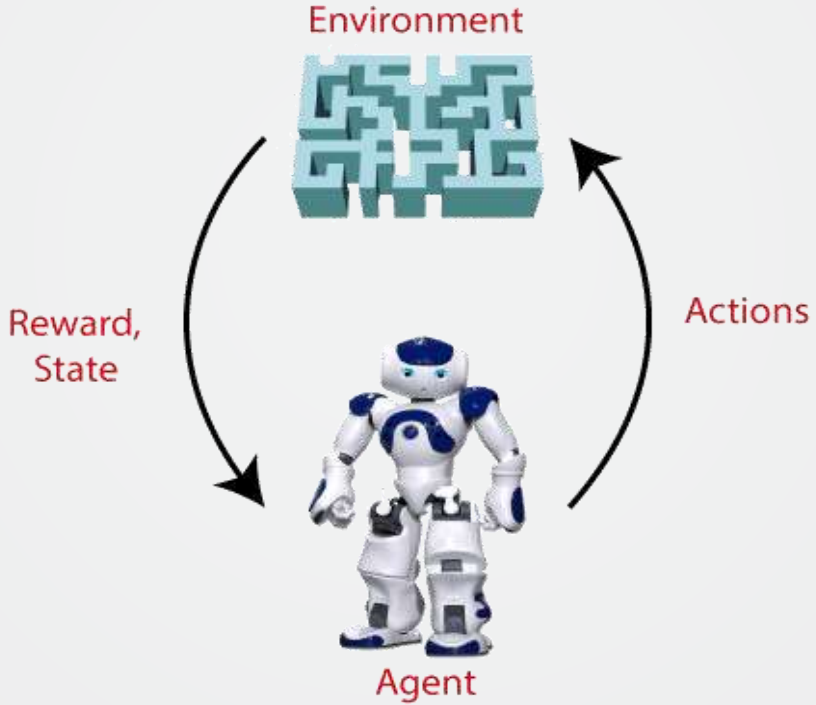
- **Supervised learning** : Task Driven (Prediction & Classification)
- **Unsupervised learning** : Data Driven (Clustering & Association)
- **Reinforcement learning** —
  - ✓Closer to human learning.
  - ✓Algorithm learns a policy of how to act in a given environment.
  - ✓Every action has some impact in the environment, and the environment provides rewards that guides the learning algorithm.

Reinforcement Learning

- Reinforcement learning *involves training an agent to interact with an environment in order to achieve a specific goal or maximize a cumulative reward.*
- The *agent learns through trial and error*, *receiving feedback from the environment in the form of rewards or penalties*.
- i.e., is *a feedback-based learning method*, in which a learning *agent gets a reward for each right action* and *gets a penalty for each wrong action.*
- It is commonly used in scenarios such as *game playing, robotics, and autonomous systems.*

Reinforcement Learning

- The goal of an agent is *to get the most reward points, and hence, it improves its performance.*



Reinforcement Learning

Example:

You can see a dog and a master. Let's imagine you are training your dog to get the stick. Each time the dog gets a stick successfully, you offered him a feast (a bone let's say). Eventually, the dog understands the pattern, that whenever the master throws a stick, it should get it as early as it can to gain a reward (a bone) from a master in a lesser time.

Reinforcement Learning

Real time applications:

- Game Playing
- Robot Navigation
- Self driving cars
- Healthcare
- Finance
- Natural Language Processing (NLP)

References

- Stuart Russell and Peter Norvig, “Artificial Intelligence”, 2nd edition, Pearson Education, 2003.
- Saroj Koushik, “Artificial intelligence”.
- NPTEL

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Thank You