Department of Computer Engineering

(01CE1702 – Artificial Intelligence)

Marwadi University
Marwadi Chandarana Group

NAAC A+

Unit-4

Game Playing & Planning

Unit-4 Game Playing & Planning

Marwadi University
Marwadi Chandarana Group
NAAC A+

Outlines:

- Overview
- MiniMax Procedures
- Alpha Beta Cut-Offs
- Refinement Theory
- Introduction to the Blocks World
- Components of a Planning System
- Goal Stack Planning
- Non-Linear Planning
- Hierarchical Planning
- Reactive Systems

Unit-4 Game Playing & Planning

Outlines:

- Overview
- MiniMax Procedures
- Alpha Beta Cut-Offs
- Refinement Theory
- Introduction to the Blocks World
- Components of a Planning System
- Goal Stack Planning
- Non-Linear Planning
- Hierarchical Planning
- Reactive Systems

Unit-4 Game Playing & Planning

- Game Playing is an important domain of Artificial Intelligence.

- There are two reasons that games appeared to be a good domain.
  1. They provide a structured task in which it is very easy to measure success or failure.
  2. They are easily solvable by a straightforward search from the starting state to a winning position.

- Games require only the domain knowledge such as the rules, legal moves and the conditions of winning or losing the game.

- In a two-player game, both the players try to win the game. So, both of them try to make the best move possible at each turn.

- To improve the effectiveness of a search based problem solving program two things can be done.
  1. Improve generate procedure so that only good moves are generated.
  2. Improve test procedure so that the best move will be recognized and explored first.

- The depth of the resulting tree or graph and its branching factor will be too large.
- Instead of legal-move generator we can use plausible-move generator in which only some small numbers of promising moves are generated.
- As the number of legal available moves increases it becomes increasingly important in applying heuristics to select only those moves that seem more promising.
- The performance of overall system can be improved by adding heuristic knowledge into both the generator and the tester.
- It is possible to search tree only ten or twenty moves deep then in order to choose the best move, the resulting board positions must be compared to discover which is most advantageous.
- The most common search technique in game playing is Minimax search procedure

Unit-4 Game Playing & Planning

Marwadi University
Marwadi Chandarana Group
NAAC A+

Outlines:

- Overview
- MiniMax Procedures
- Alpha Beta Cut-Offs
- Refinement Theory
- Introduction to the Blocks World
- Components of a Planning System
- Goal Stack Planning
- Non-Linear Planning
- Hierarchical Planning
- Reactive Systems

Unit-4 Minimax Search Procedure

- The Minimax search is a depth first and depth limited procedure.
- The idea is to start at the current position and use the plausible-move generator to generate the set of possible successor positions.
- Now we can apply the static evaluation function to those positions and simply choose the best one.
- After doing so, we can back that value up to the starting position.
- Here, we assume that static evaluation function returns larger values to indicate good situations for us.
- So our goal is to maximize the value of the static evaluation function of the next board position.
- The opponents' goal is to minimize the value of the static evaluation function.

Unit-4 Minimax Search Procedure

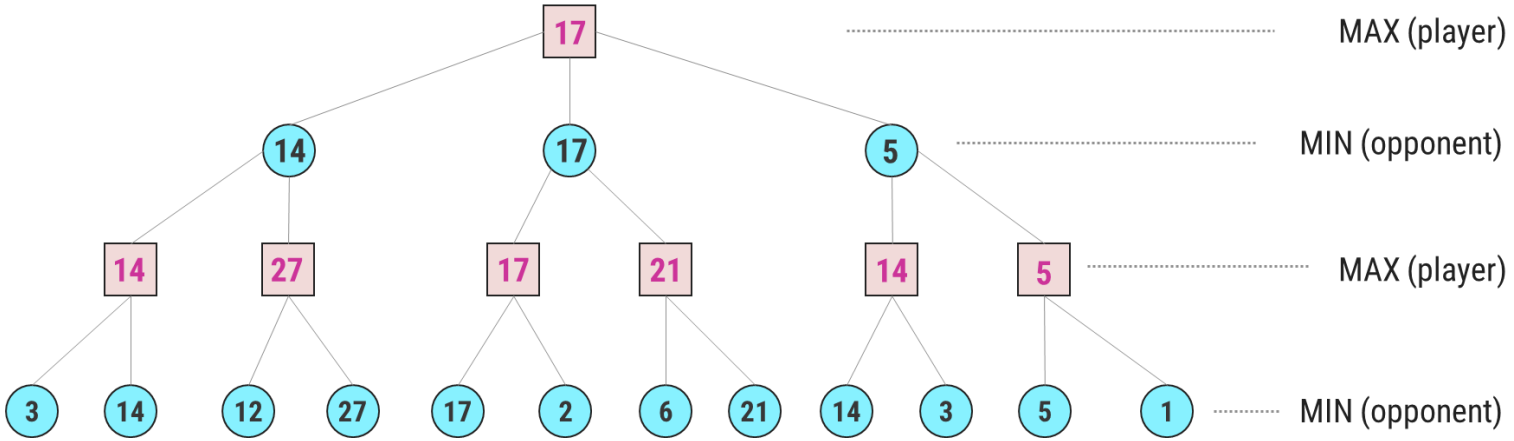
- The alternation of maximizing and minimizing at alternate ply when evaluations are to be pushed back up corresponds to the opposing strategies of the two players is called MINIMAX.

- It is recursive procedure that depends on two procedures :
    1. MOVEGEN(position, player)— The plausible-move generator, which returns a list of nodes representing the moves that can be made by Player in Position.
    2. STATIC(position, player) -- static evaluation function, which returns a number representing the goodness of Position from the standpoint of Player.

- To decide when recursive procedure should stop, variety of factors may influence the decision such as,
    - Has one side won?
    - How many ply have we already explored? Or how much time is left?
    - How stable is the configuration?

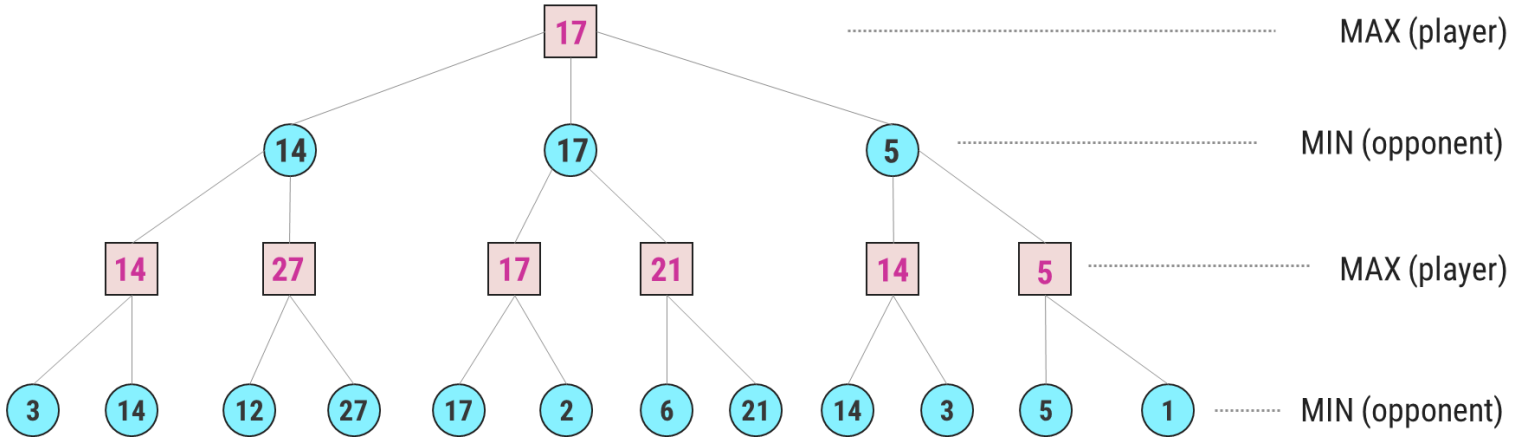
Unit-4 Minimax Search Procedure

- Given a game tree, the optimal strategy can be determined from the minimax value of each node, which we write as MINIMAX(n).
- The minimax algorithm computes the minimax decision from the current state.
- It uses a simple recursive computation of the minimax values of each successor state, directly implementing the defining equations.
- The recursion proceeds all the way down to the leaves of the tree, and then the minimax values are backed up through the tree as the recursion unwinds.



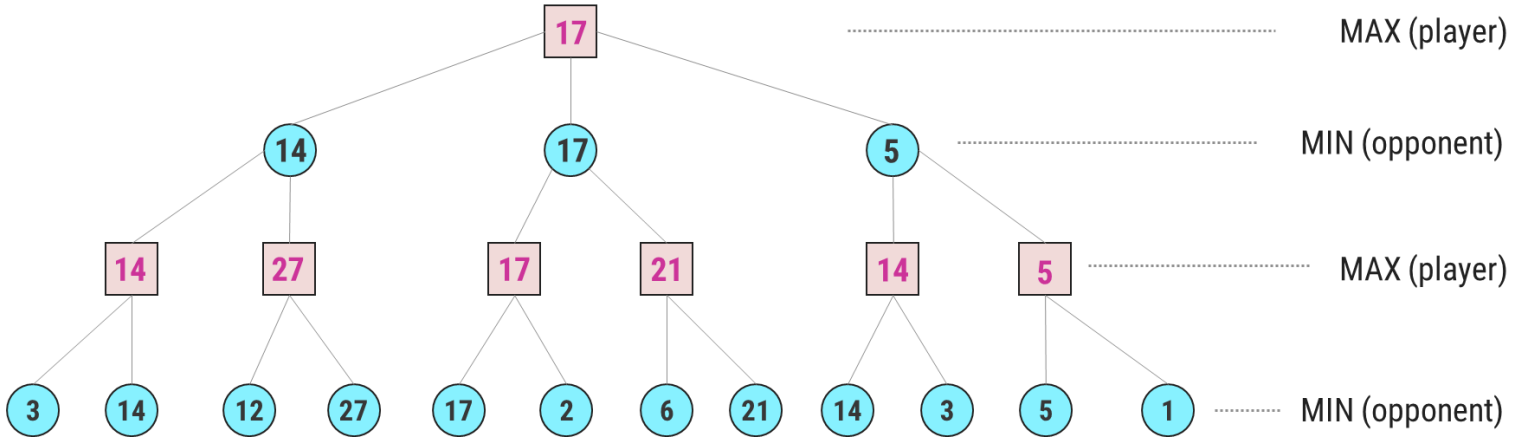
Unit-4 Minimax Search Procedure

- The minimax value of a node is the utility (for MAX) of being in the corresponding state, assuming that both players play optimally from there to the end of the game.
- The algorithm first recuses down to the three bottom left nodes and uses the UTILITY function on them to discover that their values are 3 and 14 respectively.
- Then it takes the maximum of these values, 14, and returns it as the backed up value of parent node.
- A similar process gives the backed-up values of 27, 17, 21, 14 and 5 respectively.



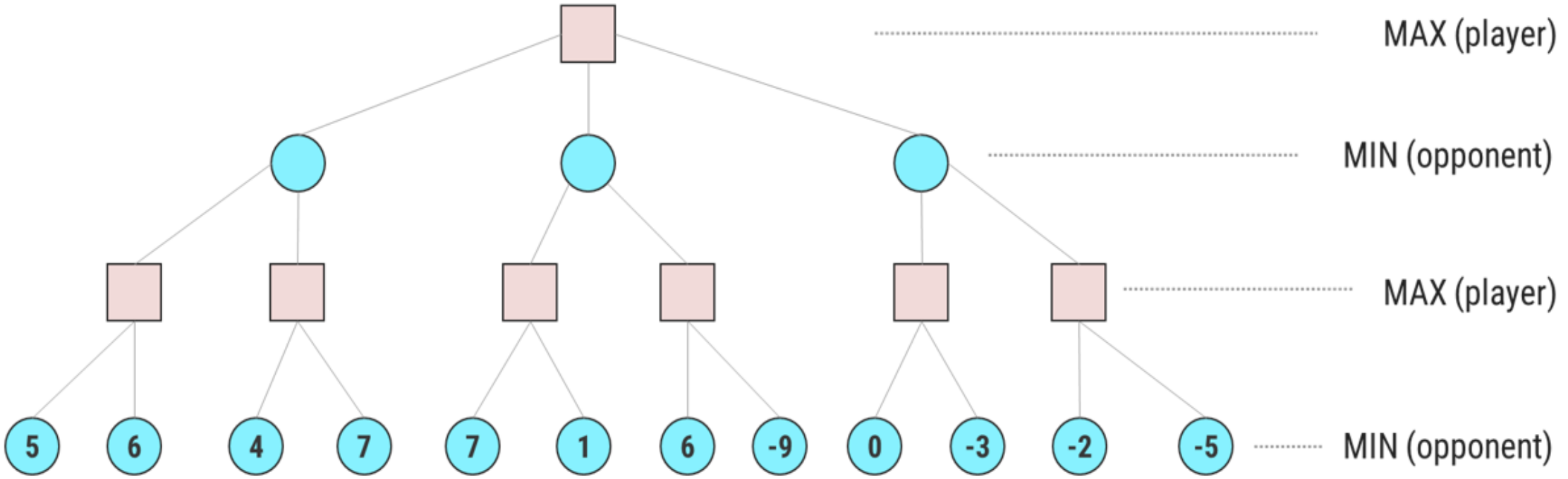
Unit-4 Minimax Search Procedure

- Then it takes the minimum of these values 14, 17 and 5 respectively.
- Finally, we take the maximum of 14, 17, and 5 to get the backed-up value of 17 for the root node.
- The minimax algorithm performs a complete depth-first exploration of the game tree.
- During every ply MAX prefers to move to a state of maximum value, whereas MIN prefers a state of minimum value.

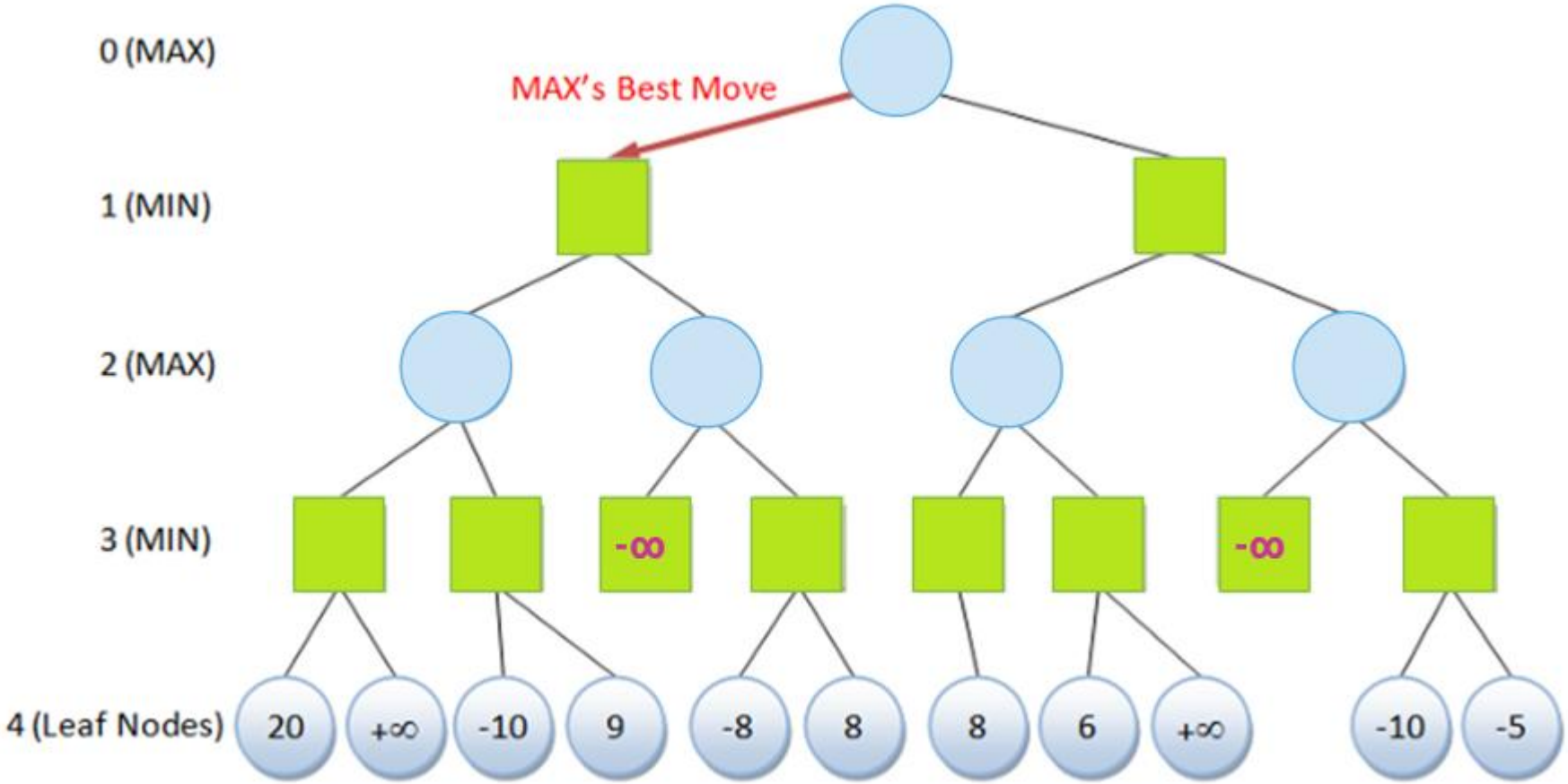


Unit-4 Minimax Search Procedure

Practice Problem - 1



Practice Problem - 2



Unit-4 Game Playing & Planning

Marwadi University
Marwadi Chandarana Group
NAAC A+

Outlines:

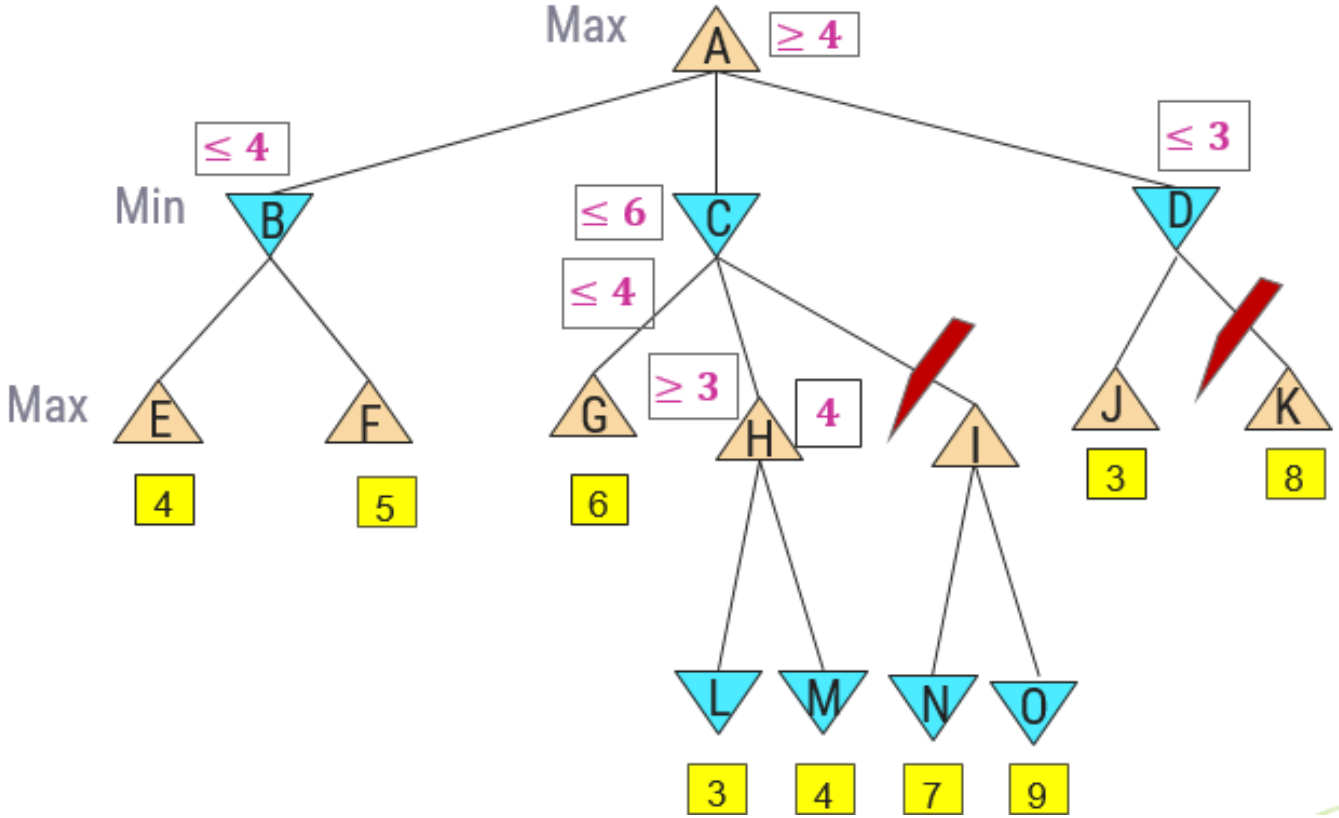
- Overview
- MiniMax Procedures
- Alpha Beta Cut-Offs
- Refinement Theory
- Introduction to the Blocks World
- Components of a Planning System
- Goal Stack Planning
- Non-Linear Planning
- Hierarchical Planning
- Reactive Systems

Unit-4 Alpha-Beta Pruning

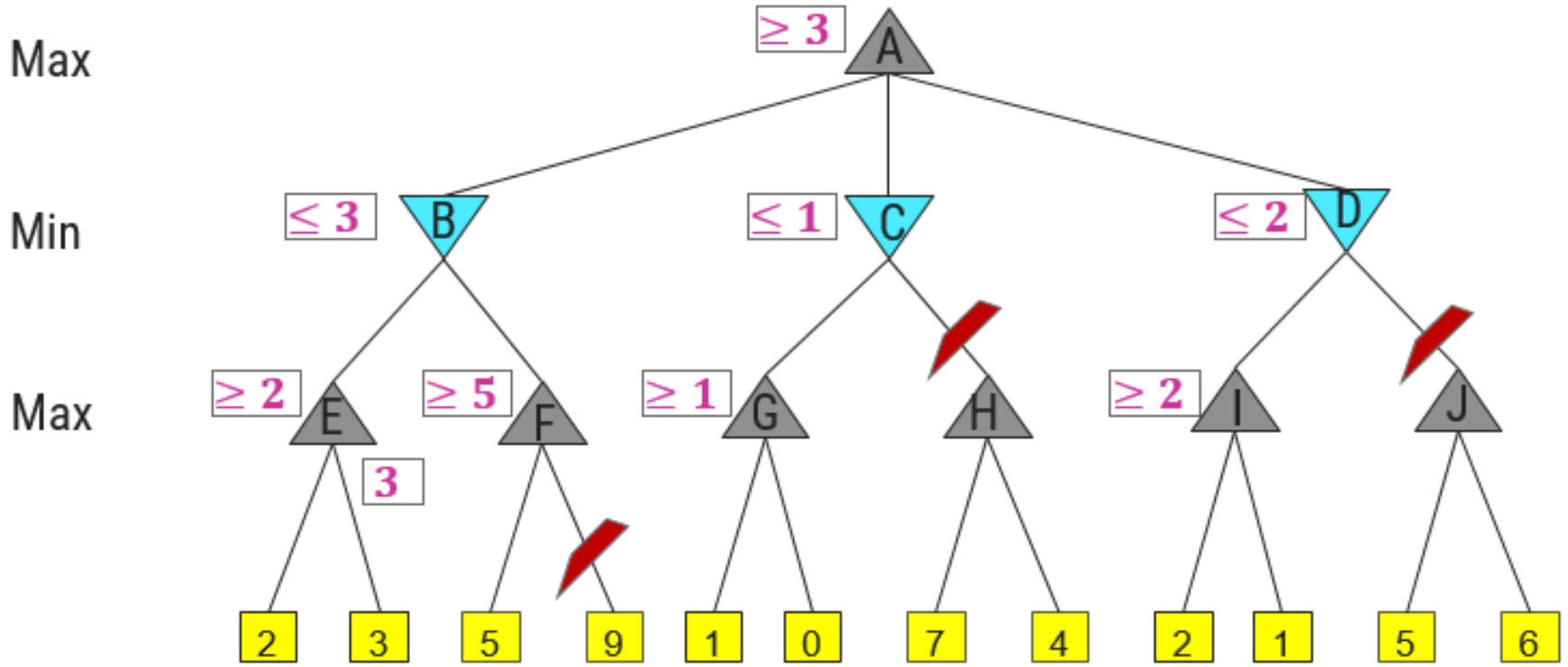
- Alpha-beta pruning is a modified version of the Minimax algorithm. It is an optimization technique for the Minimax algorithm.
- In the Minimax search algorithm, the number of game states to be examined can be exponential in the depth of a tree.
- Hence there is a technique by which without checking each node of the game tree we can compute the correct Minimax decision, and this technique is called pruning.
- This involves two threshold parameter Alpha and beta for future expansion, so it is called alpha-beta pruning. It is also called as Alpha-Beta Algorithm.
- Alpha-beta pruning can be applied at any depth of a tree, and sometimes not only it prunes the tree leaves but also entire sub-tree.

Unit-4 Alpha-Beta Pruning

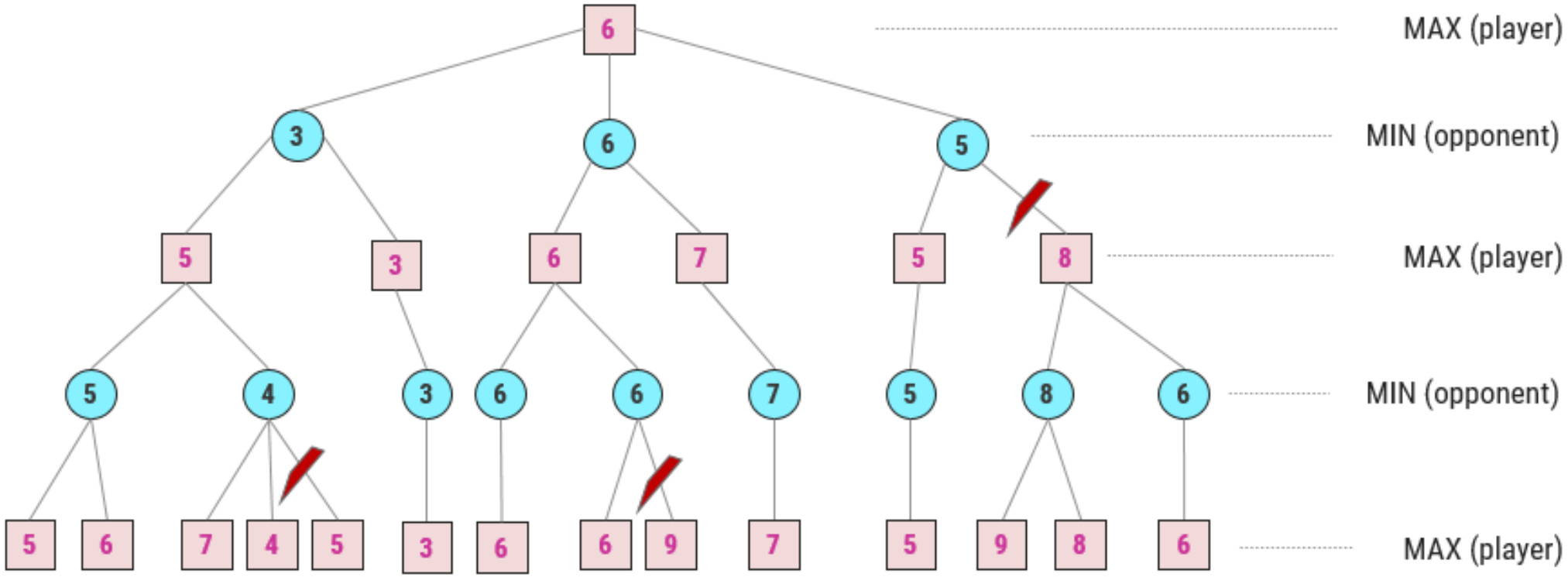
- Alpha-beta pruning technique maintains two bounds:
    1. **Alpha (α):** The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is -∞. A lower bound on best, i.e., Max
    2. **Beta (β):** The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is +∞. An upper bound on what the opponent can achieve



Alpha-Beta Pruning Example - 2



Alpha-Beta Pruning Example - 3



Unit-4 Game Playing & Planning

Marwadi University
Marwadi Chandarana Group
NAAC A+

Outlines:

- Overview
- MiniMax Procedures
- Alpha Beta Cut-Offs
- Refinement Theory
- Introduction to the Blocks World
- Components of a Planning System
- Goal Stack Planning
- Non-Linear Planning
- Hierarchical Planning
- Reactive Systems

Unit-4 Game Refinement Theory

- Game theory is a discipline which stands from the game player's point of view with a focus on how to win a game.
- However, game designers would consider another important aspect: how to make a game more attractive.
- With such motivation, a new game theory from the game designer's point of view, called game refinement theory was proposed in the early 2000s.
- Von Neumann was a pioneer who formed the foundation for the modern game theory, which has widely been applied in various fields.
- One direction with game theory was to find the best move in a game or to ensure the possibility of winning the game based on the understanding of current positions.
- Another direction with game refinement theory was to assess the attractiveness or sophistication of a game.

Unit-4 Game Refinement Theory

- In particular, game refinement theory gives a measure to quantify the sophistication of a game. This enables to obtain the deep insight into the current game and improve the quality of the game.
- The measure of game refinement can also be used to obtain the deep insight into the history of games.
- It also gives a reasonable look on the evolution of specific game variants.
- Game refinement theory has been widely applied to many different types of games with the promising results.
- We can extend the idea of game refinement into the other domains in human life such as sports games, video games, education or business.
- In many activities of human, the engagement is used as one of the important standards to evaluate the effectiveness of those activities.

Unit-4 Game Playing & Planning

Outlines:

- Overview
- MiniMax Procedures
- Alpha Beta Cut-Offs
- Refinement Theory
- Introduction to the Blocks World
- Components of a Planning System
- Goal Stack Planning
- Non-Linear Planning
- Hierarchical Planning
- Reactive Systems

Unit-4 Planning

- The planning in Artificial Intelligence is about the decision making tasks performed by the robots or computer programs to achieve a specific goal.
- The execution of planning is about choosing a sequence of actions with a high likelihood to complete the specific task.
- Planning is the task of finding a procedural course of action for a declaratively described system to reach its goals while optimizing overall performance measures.
- Automated planners find the transformations to apply in each given state out of the possible transformations for that state.
- In contrast to the classification problem, planners provide guarantees on the solution quality.

Unit-4 Planning

- Automation is an emerging trend that requires efficient automated planning.

- Many applications of planning in industry are, e.g.. robots and autonomous systems, cognitive assistants, cyber security, service composition, etc.

- Requirements of AI Planning Techniques
  - When explain ability is desired
  - When you want to be able to explain why a particular course of action was chosen
  - Assignment of responsibility is essential for automation of processes (e.g., autonomous driving, medical expert systems)

- In many real life applications, there is a structure of the problem that cannot be learned with Deep Learning (DL).

Unit-4 Planning

- Solving optimization problems with learning is hard, but integrating planning techniques with heuristic guidance learned by DL will result in the most famous success stories of AI to day.
  - GO player AlphaGO uses planning (monte-carlo tree search) with deep learning (heuristic guidance) to select the next move
  - Cognitive assistant (Samsung) uses knowledge graph, planning, and deep learning to answer complicated queries

Unit-4 Game Playing & Planning

Marwadi University
Marwadi Chandarana Group
NAAC A+

Outlines:

- Overview
- MiniMax Procedures
- Alpha Beta Cut-Offs
- Refinement Theory
- Introduction to the Blocks World
- Components of a Planning System
- Goal Stack Planning
- Non-Linear Planning
- Hierarchical Planning
- Reactive Systems

Unit-4 Components of Planning System

- Planning refers to the process of computing several steps of a problem-solving procedure before executing any of them.

- The planning consists of following important steps:

1. Choose the best rule for applying the next rule based on the best available heuristics.

2. The most widely used technique for selecting appropriate rules to apply is first to isolate a set of differences between desired goal state and then to identify those rules that are relevant to reduce those differences.

Unit-4 Components of Planning System

3. Detect when a solution has been found

A planning system has succeeded in finding a solution to a problem when it has found a sequence of operators that transform the initial problem state into the goal state.

How will it know when this has done?

In simple problem-solving systems, this question is easily answered by a straightforward match of the state descriptions.

One of the representative systems for planning systems is predicate logic. Suppose that as a part of our goal, we have the predicate P(x).

To see whether P(x) satisfied in some state, we ask whether we can prove P(x) given the assertions that describe that state and the axioms that define the world model.

Unit-4 Components of Planning System

4. Detect dead ends so that they can be abandoned and the system's effort is directed in more fruitful directions.

- As a planning system is searching for a sequence of operators to solve a particular problem, it must be able to detect when it is exploring a path that can never lead to a solution.
- The same reasoning mechanisms that can use to detect a solution can often use for detecting a dead end.
- If the search process is reasoning forward from the initial state. It can prune any path that leads to a state from which the goal state cannot reach.
- If search process reasoning backward from the goal state, it can also terminate a path either because it is sure that the initial state cannot reach or because little progress made.

5. Detect when an almost correct solution has been found.

- The kinds of techniques discussed are often useful in solving nearly decomposable problems.
- One good way of solving such problems is to assume that they are completely decomposable, proceed to solve the sub-problems separately. And then check that when the sub-solutions combined.
- They do in fact give a solution to the original problem.

Unit-4 Blocks World Problem

- Planning refers to the process of computing several steps of a problem-solving procedure before executing any of them.

- In order to compare different methods of planning, we should look at all of them in a single domain.

- The Block World Problem is described as,
  - There is a flat surface on which blocks can be placed.
  - There are a number of square blocks, all the same size.
  - The blocks are labeled with alphabets ‘A’, ‘B’, ‘C’, etc.
  - They can be stacked one upon the other.
  - There is robot arm that can manipulate the blocks.
  - The start state and goal state are given.

Unit-4 Blocks World Problem

- Actions of the robot arm (the robot arm can hold only one block at a time)
  - UNSTACK(A, B): Pick up block A from its current position on block B.
  - STACK(A, B): Place block A on block B.
  - PICKUP(A): Pick up block A from the table and hold it.
  - PUTDOWN(A): Put block A down on the table.

- Predicates : In order to specify both the conditions under which an operation may be performed and the results of performing it, we need the following predicates:
  1. ON(A, B): Block A is on Block B.
  2. ONTABLES(A): Block A is on the table.
  3. CLEAR(A): There is nothing on the top of Block A.
  4. HOLDING(A): The arm is holding Block A.
  5. ARMEMPTY: The arm is holding nothing.

Unit-4 STRIPS-Based Approach to Robot Control

- It uses the 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.
- Environment: office environment with specially colored and shaped objects.
- STRIPS planner: developed for this system to determine the actions of the robot should take to achieve goals.
- STRIPS (STanford Research Institute Problem Solver) is a restrictive way to express states, actions and goals, but leads to more efficiency.

- ADD List : List of new predicates that the operator causes to become true.
- DELETE List : List of old predicates that the operator causes to become false.
- PRECONDITIONS list contains those predicates that must be true for the operator to be applied.
- STRIPS style operators for BLOCK World problem are :

  - STACK(x, y)
  - P: CLEAR(y) Λ HOLDING(x)
  - D: CLEAR(y) Λ HOLDING(x)
  - A: ARMEMPTY Λ ON(x, y)
  - UNSTACK(x, y)
  - PICKUP(x)
  - P: CLEAR(x) Λ ONTABLE(x) Λ ARMEMPTY
  - D: ONTABLE(x) Λ ARMEMPTY
  - A: HOLDING(x)
  - PUTDOWN(x)

Unit-4 Game Playing & Planning

Marwadi University
Marwadi Chandarana Group
NAAC A+

Outlines:

- Overview
- MiniMax Procedures
- Alpha Beta Cut-Offs
- Refinement Theory
- Introduction to the Blocks World
- Components of a Planning System
- Goal Stack Planning
- Non-Linear Planning
- Hierarchical Planning
- Reactive Systems

Unit-4 Goal Stack Planning

Goal Stack Planning is the one of the simplest planning algorithms that is designed to handle problems which include compound goals.

It utilizes STRIPS as a formal language for specifying and manipulating the world with which it is working.

This approach uses a Stack for plan generation. The stack can contain Sub-goal and actions described using predicates. The Sub-goals can be solved one by one in any order.

It starts by pushing the unsatisfied goals into the stack.

Then it pushes the individual sub-goals into the stack and its pops an element out of the stack.

When popping an element out of the stack the element could be,

- either a predicate describing a situation about our world or
- it could be an action that can be applied to our world under consideration.

Unit-4 Goal Stack Planning

So, a decision has to be made based on the kind of element we are popping out from the stack.

If it is a Predicate, then compares it with the description of the current world, if it is satisfied or is already present in our current situation then there is nothing to do because already its true.

On the contrary if the Predicate is not true then we have to select and push relevant action satisfying the predicate to the Stack.

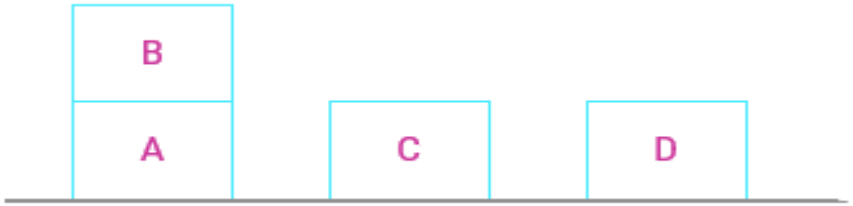
So after pushing the relevant action into the stack its precondition should also has to be pushed into the stack.

In order to apply an operation its precondition has to be satisfied, i.e., the present situation of the world should be suitable enough to apply an operation.

For that, the preconditions are pushed into the stack once after an action is pushed.

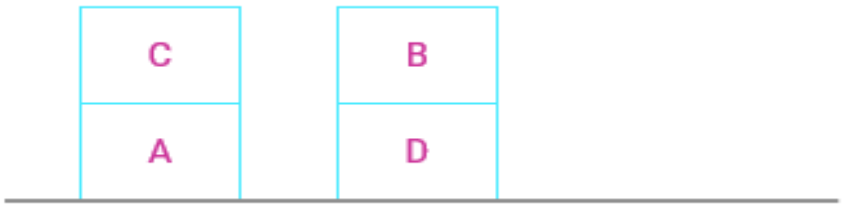
Unit-4 Goal Stack Planning

Lets start here with the BLOCK WORLD example, the initial state is our current description of our world. The Goal state is what we have to achieve.



Initial State

ON(B, A)ΛONTABLE(C) ΛONTABLE(A) ΛONTABLE(D) ΛCLEAR(D) ΛCLEAR(C) ΛCLEAR(B) Λ ARMEMPTY



Goal State

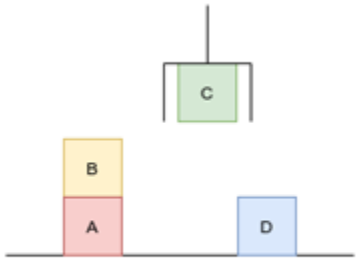
ON(C,A)ΛON(B,D) ONTABLE(A) ΛONTABLE(D) ΛCLEAR(C) ΛCLEAR(B) Λ ARMEMPTY

Unit-4 Goal Stack Planning

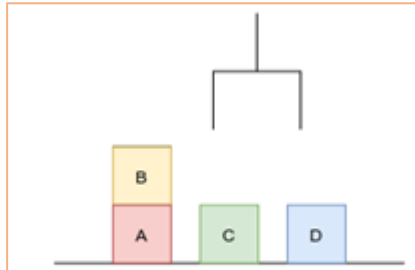
The following list of actions can be applied to the various situations in the problem.

OPERATORS	PRECONDITION	DELETE	ADD
STACK(A, B): Place block A on block B.	CLEAR(B) Λ HOLDING (A)	CLEAR(B) Λ HOLDING (A)	ARMEMPTY Λ ON(A,B)
UNSTACK(A, B): Pick up block A from its current position on block B.	ON(A,B) Λ CLEAR(A) Λ ARMEMPTY	ON(A,B) Λ ARMEMPTY	HOLDING (A) Λ CLEAR(B)
PICKUP(A): Pick up block A from the table and hold it.	CLEAR(A) Λ ONTABLE(A) Λ ARMEMPTY	ONTABLE(A) Λ ARMEMPTY	HOLDING (A)
PUTDOWN(A): Put block A down on the table.	HOLDING (A)	HOLDING (A)	ONTABLE(A) Λ ARMEMPTY

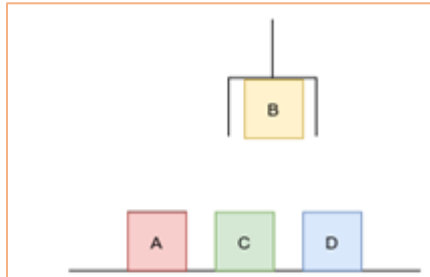
Unit-4 Goal Stack Planning



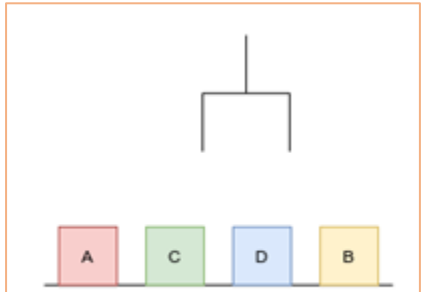
1 - PICKUP(C)



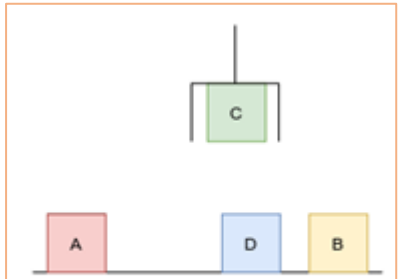
2 - PUTDOWN(C)



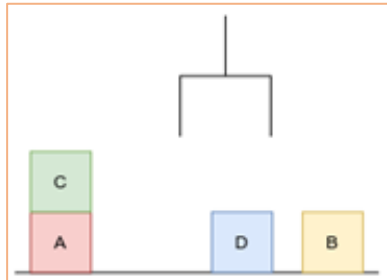
3 - UNSTACK(B,A)



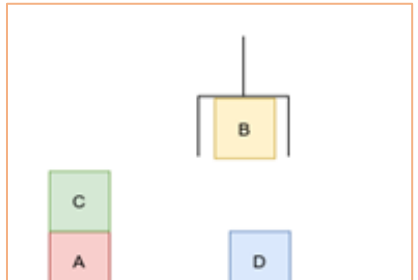
PUTDOWN(B)



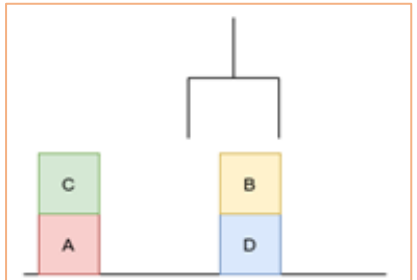
5 - PICKUP(C)



6 - STACK(C,A)



7 - PICKUP(B)



8 - STACK(B,D)

Unit-4 Constraint Posting

- The idea of constraint posting is to build up a plan by incrementally hypothesizing operators, partial orderings between operators, and binding of variables within operators.
- Constraint posting often comes with Non-Linear Planning. The idea of constraint posting is to build up a plan incrementally.
- At any given time in the problem-solving process, we may have a set of useful operators but perhaps no clear idea of how those operators should order with respect to each other.
- A solution is a partially ordered, partially instantiated set of operators to generate an actual plan. And we can convert the partial order into any number of total orders.

Unit-4 Game Playing & Planning

Marwadi University
Marwadi Chandarana Group
NAAC A+

Outlines:

- Overview
- MiniMax Procedures
- Alpha Beta Cut-Offs
- Refinement Theory
- Introduction to the Blocks World
- Components of a Planning System
- Goal Stack Planning
- Non-Linear Planning
- Hierarchical Planning
- Reactive Systems

Unit-4 Nonlinear Planning using Constraint Posting

- This planning is used to set a goal stack and is included in the search space of all possible sub-goal orderings. It handles the goal interactions by interleaving method.
- Many problems require an intertwined plan in which multiple sub-problems worked on simultaneously.
- Such a plan is called nonlinear plan because it is not composed of a linear sequence of complete sub-plans.
- Non-linear planning may be an optimal solution with respect to plan length (depending on search strategy used).
- It takes larger search space, since all possible goal orderings are taken into consideration.

Unit-4 Constraint Posting versus State Space Search

State Space Search	Constraint Posting Search
Moves in the space: Modify world state via operator	Moves in the space: Add operators, Oder Operators, Bind variables Or Otherwise constrain plan.
Model of time: Depth of node in search space.	Model of Time: Partially ordered set of operators.
Plan stored in Series of state transitions.	Plan stored in Single node.

Unit-4 Game Playing & Planning

Outlines:

- Overview
- MiniMax Procedures
- Alpha Beta Cut-Offs
- Refinement Theory
- Introduction to the Blocks World
- Components of a Planning System
- Goal Stack Planning
- Non-Linear Planning
- Hierarchical Planning
- Reactive Systems

Unit-4 Hierarchical Planning

- In order to solve hard problems, a problem solver may have to generate long plans.
- It is important to be able to eliminate some of the details of the problem until a solution that addresses the main issues is found.
- Then an attempt can be made to fill in the appropriate details.
- Early attempts to do this involved the use of macro operators, in which larger operators were built from smaller ones.
- Lower level activities would detail more precise steps for accomplishing the higher level tasks.
- Instead of having to try out a large number of possible plan ordering, plan hierarchies limit the ways in which an agent can select and order its primitive operators.

Unit-4 Hierarchical Planning

Hierarchical Planning describes Hierarchy of actions in terms of major action or minor action.

Planning for ”Going to Goa this Christmas”

1. Major Steps :
   - ❑ Hotel Booking
   - ❑ Ticket Booking
   - ❑ Reaching Goa
   - ❑ Staying and enjoying there
   - ❑ Coming Back

2. Minor Steps :
   - ❑ Take a taxi to reach station / airport
   - ❑ Have candle light dinner on beach
   - ❑ Take photos

Unit-4 Hierarchical Planning

- Hierarchical Planning Properties
  - ❑ Postpone attempts to solve mere details, until major steps are in place.
  - ❑ Identify a hierarchy of conditions
  - ❑ Construct a plan in levels, postponing details to the next level
  - ❑ Patch higher levels as details become visible

- Hierarchy of conditions reflect the intrinsic difficulty of achieving various conditions. Intrinsic difficulty is indicated by criticality value.

- A operation having minimum criticality can be trivially achievable, i.e., the operations having very less or no precondition. For example : Opening makemytrip.com

- Similarly operation having many preconditions to satisfy will have higher criticality.

- The assignment of appropriate criticality value is crucial to the success of this hierarchical planning method.

- Those preconditions that no operator can satisfy are clearly the most critical.

Unit-4 Game Playing & Planning

Marwadi University
Marwadi Chandarana Group
NAAC A+

Outlines:

- Overview
- MiniMax Procedures
- Alpha Beta Cut-Offs
- Refinement Theory
- Introduction to the Blocks World
- Components of a Planning System
- Goal Stack Planning
- Non-Linear Planning
- Hierarchical Planning
- Reactive Systems

Unit-4 Reactive Systems

- The idea of reactive systems is to avoid planning altogether, and instead, use the observable situation as a clue to which one can simply react.
- A reactive system must have an access to a knowledge base of some sort that describes what actions should be taken under what circumstances.
- A reactive system is very different from the other kinds of planning systems we have discussed. Because it chooses actions one at a time.
- It does not anticipate and select an entire action sequence before it does the first thing.
- The example is a Thermostat. The job of the thermostat is to keep the temperature constant inside a room.
- Reactive systems are capable of complex behaviors.
- The main advantage reactive systems have over traditional planners is that they operate robustly in domains that are difficult to model completely and accurately.

Unit-4 Reactive Systems

- Reactive systems dispense with modeling altogether and base their actions directly on their perception of the world.
- Another advantage of reactive systems is that they are extremely responsive since they avoid the combinatorial explosion involved in deliberative planning.
- This makes them attractive for real-time tasks such as driving and walking.

Reference Book

Artificial Intelligence – Kevin Knight and Elaine Rich

THANK YOU