The first topic, Approach to AI, is the most important and is frequently asked in exams. It covers topics like Heuristic Search and A*. It is essential to thoroughly understand these topics for better performance in exams.

Algorithm Overview

Let's start by discussing the best first algorithm. Then, we will move on to game playing algorithms, specifically the minimax algorithm and alpha beta cut off. These are the main topics that we will cover.

Constraint Satisfaction

In addition to these topics, we will also cover constraint satisfaction where we use crypt analysis.

DFS and BFS

Lastly, we will touch upon DFS and BFS algorithms. Although these are typically discussed in data structure, they are important to understand in the context of these other algorithms, such as A*.

When preparing for exams, it is important to focus on certain topics that are likely to appear in the test. For instance, questions related to theory and graphs are common, as well as those related to fuzzy sets, alpha cuts, and set operations such as union, intersection, and minus. These questions are logical and can be easily solved with practice.

- Priority topics: Fuzzy sets and set operations
- Neural network: This is an important topic, but the number of questions asked on it may not be as many as the previous topics.
- Other topics: Genetic algorithms and Single.

Types of Neural Networks

- Multi-layer feedforward network
- Recurrent hope field network

Machine Learning

In machine learning, we generally talk about supervised and unsupervised learning. It's important to understand these learning strategies.

Theory Topics

These learning strategies are mainly related to theory only.

Knowledge Representation

Knowledge representation is an important concept in machine learning.

NLP Multi Agent Exam Review

When reviewing for the NLP Multiagent exam, I would give 2 stars to the multiagent section. This section covers the types of agents and their properties, including how they use current and past history. While the questions may be simple, they test your knowledge on these topics.

For knowledge representation, NLP, and planning, I would give them one star. While there may be fewer questions on these topics, they still require studying. Specifically, in knowledge representation, you should focus on the approaches to representing knowledge, such as predicate logic. I would give predicate logic 2 stars internally since it is also a concept in mathematics and artificial intelligence.

Using Heuristics to Solve a Puzzle

If we move the puzzle piece down, the space will go down and 7 will come up. We can only move the piece right, so the third move will be to the right. This will create three states in blind search. Informed search or the best first method works by calculating the heuristic value, and exploring the state with the minimum value. We calculate the heuristic value of each state by counting how many tiles are misplaced based on the goal state.

- State 1: 1 is misplaced, 2 is not, 3 is not, 4 has 2 misplaced, 5 is misplaced, 6 is in the right position, 7 is in the right position, and 8 is misplaced. The heuristic value is 4.
- State 2: 1, 2, and 8 are misplaced, 3 and 6 are in the right position, 4 is not, 5 is in the right position, and 7 is not there. The heuristic value is 4.
- State 3: 1, 2, 3, 4, and 8 are in the right position, 5 is not, 6 is in the right position, and 7 is in the right position. The heuristic value is 2.

The minimum heuristic value is 2, so we will explorer this state next.

The concept of heuristic is to find a quick answer by exploring the minimum value. In this case, the minimum value of the state is 2, which is the heuristic value, indicating that it will lead to the goal state quickly. Therefore, there is no need to explore other paths. We only need to explore the path with the minimum heuristic value.

There are four choices to explore, up, down, left, and right. Moving right results in the state 1 2 3 4 6 empty 7 5 8. Moving left, however, will not benefit as it will only lead back to the parent state. Moving up results in the state 1 2 3 4 6 7 5 8. Moving down results in the state 1 2 3 4 5 6 7 8. The heuristic value of the resulting state after moving down is 1, with only one tile not in its correct position. The heuristic value of the resulting state after moving up is 2, with two tiles not in their correct positions. The heuristic value of the resulting state after moving right is 3, with three tiles not in their correct positions. Therefore, moving down has the minimum heuristic value and is the best path to explore.

Using Heuristics in Problem Solving

When solving a problem using heuristics, it is essential to find the misplaced tile. In a blind search, you explore all the depths, but with heuristics, you only explore a state that is most likely to lead to the solution.

How Heuristics Work

Heuristics work by finding the minimum misplaced tile and exploring the state that is most likely to lead to the solution. For instance, if we have a puzzle with tiles numbered from 1 to 8, and we want to find the misplaced tile, we look for the minimum value. If the minimum value is 3, we explore that state and stop exploring other states.

Exploring States

When exploring states, we must keep in mind the goal state. If we find the goal state, there is no need to explore further. For instance, if we have a puzzle with a goal state of 1 2 3 4 5 6 7 8 and we explore the state 1 2 3 4 5 6 7 8, we have achieved the goal state, and there is no need to explore further.

Uninformed vs. Informed Heuristics

In uninformed heuristics, we explore all the depths with a branch factor of b and a depth of d, while in informed heuristics, we only explore the most likely state to lead to the solution. In both cases, the space complexity is the same, but with informed heuristics, we find the solution quickly. However, there is no guarantee that we will get an optimal solution.

Max, on the other hand, will try to maximize their utility by selecting the best possible move. It's crucial to note that what may be the best move for Max may not be the best move for you.
Let's take a look at a game tree starting from point A:
A
From this point, Max will choose the best move, while Min will choose the worst move for Max, which is the best move for Min. This is a critical aspect of game theory that players need to keep in mind.
Game Strategy
We are currently on the root level and Max has two choices: B and C. He must choose one of those options. If he chooses B, it will be Min's turn to make a move. We will play this game in a typical manner, where we take turns selecting our moves. However, we need to keep an important point in mind: once we have made our move, it will be the last move at this level.

- Max's options: B or C
- If Max chooses B, it will be Min's turn
- We will take turns making moves
- Our move will be the last at this level

When playing a game, it is important to consider the utility or potential profit of each move. For example, if a move has a utility of +8, choosing that move will result in the maximum reward. However, it is important to remember that the opponent's moves cannot be predicted, so one must choose their own strategy carefully.
After choosing a move, the next decision will depend on the opponent's move. It is important to choose the maximum value between available options. This process of choosing and backtracking helps determine the optimal move for winning the game.
Ultimately, the goal is to reach the terminal with the highest utility possible.
Let's analyze the game tree:

- Node D: Max's turn to move. The two possible values are -1 and 8. Max will choose the maximum value, which is 8.
- Node E: Min's turn to move. The two possible values are -3 and -1. Min will choose the minimum value, which is -1.

Therefore, Max will choose the left side of Node D to get the maximum utility/profit.

Alpha beta pruning

Going to discuss all the important points about Alpha-Beta Pruning, which are beneficial for your competitive exams as well as for your college or university exams.
What is Alpha-Beta Pruning?
Alpha-Beta Pruning is the advanced version of the minimax algorithm. The time complexity and performance of the minimax algorithm are calculated in B raised to power D, where B is the branching factor and D is the depth of the game tree. In other words, we explore all the nodes to find the best path for the Max player, which is the player we consider at the root level. The time complexity we get is the order of B raised to power D.

Alpha-Beta Pruning is used to cut off the search by exploring fewer nodes. If we have already found the best path, we prune all the remaining paths. Alpha and Beta are two values that we generally consider. We consider Alpha for the Max nodes and Beta for the Min nodes.

How Does Alpha-Beta Pruning Work?

To understand how Alpha-Beta Pruning works, let's take an example game tree:

- Root node: Max
- Next node: Min
- Next node: Max
- Next node: Min
- Leaf nodes: Terminals

We explore this game tree to find the best path for the Max player. If we have already found a path, we prune all the remaining paths. This way, we explore fewer nodes and get faster performance.

When playing a game where players choose paths, the Alpha-Beta Pruning method can help determine the best choices without exploring all nodes. To start, use Depth-first search to reach a terminal node with a value of 3. Above it is a Min node where the beta value will be calculated. Alpha is considered on Max nodes and Beta on Min nodes.

- Max chooses path
- Min chooses path
- Max chooses path
- Min chooses path
- Terminal nodes have utility values

To find the values of alpha and beta, use the given values and avoid exploring unnecessary nodes. This process is called Alpha-Beta Pruning.

Alpha on Max

You can write these things before:

- As on the left most side at terminal
- We have value 3
- We will fix beta's value 3

But Min value of this node, we don't know that. But by this, we got an idea that if this node's beta value is 3 so Min value will be equal or less than 3. Obviously, if we get 3 values of this beta then the min value of this node either will be 3 (we have 3 already) or will be less than 3.

We moved to this node as we moved to this node. It has a value of 4. Obviously, it is a Min node then Beta's value will remain 3 because this value is larger. Now we know that this node's min value is 3. It is confirmed now. Now this node has a value of 3. How did we consider it? We saw this value.

Fixing the Value of Beta

We need to fix the value of beta, which can either be 3 or less than 3. We take the upper bound in beta, which means plus infinity, and then decrease its value by choosing the minimum. Similarly, we take alpha's value in an increasing way, but we don't have to use minus infinity or plus infinity. Now, considering a simple example, if the node's value is fixed at 3, then either Min or Max value gets fixed accordingly. We can take the Max value of Alpha as 3 and assume the Max value of this node to be 3 as well.

Alpha-Beta Pruning

Declarative knowledge represents the non-facts objects about the world, while procedural knowledge is more complex and deals with how the objects behave and act. Meta knowledge refers to the knowledge regarding the different types of knowledge, and heuristic knowledge is obtained only from experts of that particular domain. Lastly, structural knowledge establishes the relationship between objects and provides solutions to real-life problems.
In the next note, we will discuss knowledge representation schemes such as semantic networks, frames, production rules, first logic, and proposition logic.

Propositional Logic

Going to discuss Propositional Logic and its important points. This topic is beneficial for both competitive exams and college or university level exams. So, let's dive in and learn more about Propositional Logic.
What is Propositional Logic?
Propositional logic is a simple topic in Knowledge Representation. But why do we need Knowledge Representation? The answer is simple: we want to make our machines intelligent, and knowledge is one of the key factors in building intelligence.
Therefore, it is important to represent knowledge in the correct way. Propositional Logic is a tool that helps us do that.
Knowledge Representation in Artificial Intelligence
When it comes to Artificial Intelligence, knowledge representation is crucial for the machine to be able to interpret and make decisions based on that information. Human beings acquire knowledge from childhood and store it in our minds, allowing us to use it when needed.
Similarly, in AI, there are various methods of knowledge representation, with Propositional Logic being one of the simplest and earliest methods.
Propositional Logic
Propositional Logic involves representing knowledge through sentences or propositions. This allows for the machine to understand the relationship between different propositions and make logical deductions based on that knowledge.
If we were to talk about a statement in English, "The sky is blue" would be a simple statement. In programming, a statement could be "print 'hello'". However, statements can also be written in mathematical form such as "1 + 1 = 2". These statements are also known as propositions.
Propositional Logic
Propositional logic involves reasoning and making arguments based on statements. Each statement can only have two possible outcomes: true or false. A proposition cannot be both true and false at the same time. For example, "1+1=2" is a true proposition, while "2+1=4" is a false one. Similarly, "New Delhi is the capital of India" is a true proposition.
Propositional Logic in Programming
In propositional logic, every statement has a truth value of either true or false. For example, the statement "Some students are intelligent" can be true or false depending on the context. However, if we write a statement that can only have one output, such as "2+2=4", that is a part of propositional logic. When it comes to programming, languages like C use functions like print f to output data.

However, we aim to enhance this technique by providing an additional functionality to our intelligent agent.

Planning for a Bike Trip to Leh & Ladakh

Planning is essential to reach a goal state. Let's take an example of planning a tour to Leh & Ladakh on a bike. The end goal is reaching Ladakh, starting from Delhi.

- Research and plan the route to take.
- Ensure the bike is in good condition and serviced.
- Pack necessary items such as clothes, food, and water.
- Plan for stops and accommodations along the way.

Whether it's the current situation or the past, chances are you've visited a hill station before. Maybe you got there by bike, bus, or some other means of transportation. Regardless, you have prior knowledge about your past experiences.

Utility Based Agents

Focus on Actions, Not the Goal State

It's easy to assume that the goal state is the ultimate focus of any task. However, when it comes to studying, it's important to shift your focus on the actions you take instead. This can help you avoid confusion and distractions.

Goal State and Utility Agent

When we perform actions, our main motive is typically to reach a goal state. However, it is important to note that the goal state exists in the mind of the utility agent. The utility agent determines whether or not a particular state is desirable based on its own preferences and priorities.

When it comes to reaching a destination, the focus should be on utility. This refers to the state that the agent reaches after an action is performed. The utility function is used to measure whether the agent is in a happy or unhappy state. Essentially, the function is used to gauge the level of satisfaction or dissatisfaction with the end result. This is similar to how happiness or unhappiness is measured in human beings.

The concept used here is similar to that used in GPS systems or tracking devices. Let's take a real-life example where we are traveling from source to destination in a car, covering a distance of 300-400 kilometers. We often use the GPS system which displays 2 or 3 possible routes. However, we tend to follow the shortest path by default.

In a normal or happy state, we follow the shortest path to reach the goal state. However, unforeseen events like traffic or protests can cause delays. GPS may not provide real-time data, leading to an unhappy state when we fail to reach our destination on time. In such a scenario, the agent will take action to address the situation.

The specific action taken by the agent will depend on the circumstances. For example:

- It may reroute to an alternate path to avoid traffic
- It may adjust the speed to ensure safe and efficient travel
- It may seek alternative means of transportation, such as public transit or ride-sharing

Ultimately, the goal is to reach the destination as quickly and safely as possible, even in the face of unexpected challenges.

The domain refers to a specific set of values or inputs for a particular function or system. In this case, the domain is represented by the color red.
In graph theory, we often need to assign colors to vertices. For example, suppose we have a graph with vertices colored green and blue. We need to assign a color to each vertex, but the color can be any of these 3 colors:

- Red
- Green
- Blue

This is known as a 3-coloring of the vertices. We can represent this mathematically using a function that maps each vertex to one of the 3 colors:

```
f: V → {Red, Green, Blue}
```

Branch & Bound Algorithm

Branch & Bound Algorithm is a popular method for finding optimal solutions to a problem. It involves dividing the problem into smaller subproblems, or branches, and then exploring each branch individually to find the best solution.
The algorithm works by first creating an initial solution, or node, which represents the entire problem. This node is then divided into smaller subproblems, or child nodes, which are explored one at a time.
As each child node is explored, the algorithm evaluates its potential for finding a better solution than the current best solution. If the child node has the potential for a better solution, it is added to a priority queue. The algorithm then explores the child node with the highest potential next.
This process continues until all possible branches have been explored and a final solution is found.
Branch and Bound Algorithm
The Branch and Bound algorithm involves finding sub-solutions, which is represented by the "Branch" in the name.
Problem Generation
When generating a problem, we also generate its subprograms and sub-solutions. This means that for a given problem, there can be multiple solutions.
Calculation of Minimum Value
After conducting research, we have determined that the minimum value of the given equation is:
INSERT FORMULA HERE
Once we have calculated this value, we can then proceed to find further sub-solutions based on it.
Sub-Solutions

- Sub-solution 1: INSERT FORMULA HERE
- Sub-solution 2: INSERT FORMULA HERE
- Sub-solution 3: INSERT FORMULA HERE

These sub-solutions can provide additional insights and applications for the original equation.
Code Example
INSERT CODE EXAMPLE HERE