NAME: DAKSHIT CHOPRA **DATE:** 04/03/2023

REG. NO.: RA2011003030243

BATCH: CSE K

AI Assignment 2

Unit 2

Assignment Questions

1. **What is Uniformed search and what are its different methods?**
   Ans:
   Uniformed search is a type of search algorithm used in artificial intelligence to find a solution to a problem without prior knowledge of the problem domain. This means that the algorithm does not have any information about the structure of the problem or the potential solutions, and must explore the search space systematically to find a solution.

   There are several methods of uniformed search, including:

- **Breadth-First Search (BFS):** BFS explores all the neighboring nodes at the current depth before moving on to the next level of nodes. It is guaranteed to find the shortest path to a solution if one exists.
- **Depth-First Search (DFS)**: DFS explores as far as possible along each branch before backtracking. It may find a solution faster than BFS, but it does not guarantee to find the shortest path.
- **Iterative Deepening Depth-First Search (IDDFS):** IDDFS is a combination of BFS and DFS. It repeatedly performs a DFS with a maximum depth limit, gradually increasing the depth limit until a solution is found.
- **Uniform-Cost Search (UCS):** UCS considers the cost of each path and explores the path with the lowest cost first. It is useful for finding the shortest path in a weighted graph.
- **Bidirectional Search:** Bidirectional search explores the search space from both the start and goal states, meeting in the middle to find a solution. It is more efficient than other methods if the search space is very large.
- **Depth-limited Search**
  Each of these methods has its own advantages and disadvantages, and the choice of which one to use depends on the characteristics of the problem being solved.

2. **How does the Breadth First search differ from the Depth First search?**
   Ans:
   Breadth-First Search (BFS) and Depth-First Search (DFS) are two different methods used for searching a graph or tree data structure.

   BFS explores all the **neighboring** nodes at the current depth before moving on to the next level of nodes. It starts at the root node and explores all the nodes at the first level before moving on

to the nodes at the second level, and so on. BFS guarantees to find the shortest path to a solution if one exists, but it requires more memory than DFS as it keeps track of all the nodes at each level.

On the other hand, DFS explores as far as possible along each branch before backtracking. It starts at the root node and explores the first adjacent node before moving to the next adjacent node. DFS can find a solution faster than BFS, but it does not guarantee to find the shortest path as it may get stuck in a deeper branch while a solution exists closer to the root. DFS requires less memory than BFS as it only keeps track of the nodes on the current path.

In summary, BFS is suitable for finding the shortest path in an unweighted graph, while DFS is suitable for finding a path in a large, complex graph or tree. The choice between BFS and DFS depends on the characteristics of the problem and the desired outcome.

3. **What is the purpose of Depth Limited search and what are its limitations?**
   Ans:
   Depth-Limited Search (DLS) is a variation of Depth-First Search (DFS) in which a maximum depth limit is set, and the search is terminated when that depth limit is reached. The purpose of DLS is to avoid the infinite loop problem in DFS, which can occur when a graph or tree has cycles.

   DLS is useful when the depth of the solution is known or can be estimated. It is also useful when the search space is very large, and a complete DFS would require too much time or memory. By setting a depth limit, the search algorithm can avoid exploring the entire search space, focusing only on the paths that are likely to lead to a solution.
   Limitations:
   - The main limitation of DLS is that it may not find a solution if the depth limit is set too low.
   - If the depth limit is set higher, the algorithm may take too long to terminate or may run out of memory.
   - The choice of the depth limit is critical, and it must be set carefully to balance between the time and memory constraints of the problem and the likelihood of finding a solution.
   - Another limitation of DLS is that it may find a suboptimal solution if the depth limit is set lower than the optimal depth. This is because the algorithm terminates when the depth limit is reached, even if a better solution exists at a deeper level.

4. **Explain the Iterative Deepening search and its advantages over the Depth First search.**
   Ans:
   Iterative Deepening Search (IDS) is an iterative graph searching strategy that takes advantage of the completeness of the Breadth-First Search (BFS) strategy but uses much less memory in each iteration (similar to Depth-First Search).
   IDS achieves the desired completeness by enforcing a depth-limit on DFS that mitigates the possibility of getting stuck in an infinite or a very long branch. It searches each branch of a node from left to right until it reaches the required depth. Once it has, IDS goes back to the root node and explores a different branch that is similar to DFS.

Its advantages over the Depth First Search:

1. DDFS gives us the hope to find the solution if it exists in the tree.
2. When the solutions are found at the lower depths say n, then the algorithm proves to be efficient and in time.
3. The great advantage of IDDFS is found in-game tree searching where the IDDFS search operation tries to improve the depth definition, heuristics, and scores of searching nodes so as to enable efficiency in the search algorithm.
4. Another major advantage of the IDDFS algorithm is its quick responsiveness. The early results indications are a plus point in this algorithm. This followed up with multiple refinements after the individual iteration is completed.
5. Though the work is done here is more yet the performance of IDDFS is better than single BFS and DFS operating exclusively.
6. Space and time complexities are expressed as: O(d) and here d is defined as goal depth.

5. What is Bi-directional search and how does it work?

Ans:

Searching a graph is quite famous problem and have a lot of practical use. We have already discussed here how to search for a goal vertex starting from a source vertex using BFS. In normal graph search using BFS/DFS we begin our search in one direction usually from source vertex toward the goal vertex, but what if we start search from both direction simultaneously. Bidirectional search is a graph search algorithm which find smallest path from source to goal vertex. It runs two simultaneous search –

1. Forward search from source/initial vertex toward goal vertex
2. Backward search from goal/target vertex toward source vertex

Bidirectional search replaces single search graph (which is likely to grow exponentially) with two smaller sub graphs – one starting from initial vertex and other starting from goal vertex. The search terminates when two graphs intersect.
Just like A* algorithm, bidirectional search can be guided by a heuristic estimate of remaining distance from source to goal and vice versa for finding shortest path possible.

6. What is Informed search and how is it different from Uniformed search?

Ans:

Informed search in AI is a type of search algorithm that uses additional information to guide the search process, allowing for more efficient problem-solving compared to uninformed search algorithms.

Parameters	Informed Search	Uninformed Search
Alternative name	It is also known as Heuristic Search.	It is also known as Blind Search.
Using Knowledge	It uses knowledge for the searching process.	It doesn't use knowledge for the searching process.
Performance	It finds a solution more quickly.	It finds solution slow as compared to an informed search.
Completion	It may or may not be complete.	It is always complete.
Cost Factor	Cost is low.	Cost is high.
Time	It consumes less time because of quick searching.	It consumes moderate time because of slow searching.
Direction	There is a direction given about the solution.	No suggestion is given regarding the solution in it.
Implementation	It is less lengthy while implemented.	It is more lengthy while implemented.
Efficiency	It is more efficient as efficiency takes into account cost and performance. The incurred cost is less and speed of finding solutions is quick.	It is comparatively less efficient as incurred cost is more and the speed of finding the Breadth-First solution is slow.
Computational requirements	Computational requirements are lessened.	Comparatively higher computational requirements.

7. **Explain the Generate and Test method and its limitations.**

Ans:

The Generate and Test method is a simple algorithmic approach used in Artificial Intelligence (AI) for solving problems by generating all possible solutions and testing them to see if they satisfy the given constraints.

The Generate and Test method involves two steps:

- Generate: First, it generates all possible solutions to the problem, which can be represented as a set of candidate solutions or hypotheses.
- Test: Next, it tests each hypothesis against the given constraints or criteria, and only the solutions that satisfy the constraints are considered valid.

The Generate and Test method can be applied to a wide range of problems, including logic puzzles, optimization problems, and machine learning tasks. It is a flexible and intuitive approach that can be easily implemented, and it can often find a valid solution even in the absence of a clear algorithmic approach.

Limitations:

1. It can be computationally expensive, especially when the search space is large or infinite, which can result in the algorithm taking too long to terminate or running out of memory.
2. It does not guarantee an optimal or complete solution to the problem. It only finds solutions that satisfy the given constraints, and there may be better solutions that are not considered.
3. It may not be suitable for problems with complex constraints or unknown criteria, as it requires a clear set of constraints or criteria to test the hypotheses against.
4. It may generate a large number of invalid hypotheses, which can be time-consuming to test and eliminate.

8. **What is Best First search and how is it different from A* algorithm?**

Ans:

Best First search and A* algorithm are both search algorithms that are used to find the shortest path from a starting point to a goal point in a graph or a network. However, they differ in the way they select the next node to expand in the search tree.

Best First search is a heuristic search algorithm that selects the node that is estimated to be closest to the goal based on some heuristic function. The heuristic function is used to estimate the distance from the current node to the goal. Best First search does not take into account the cost of the path to reach the current node, but only the estimated distance to the goal. As a result, Best First search may not always find the optimal path to the goal.

On the other hand, A* algorithm is also a heuristic search algorithm that selects the node based on the sum of the cost of the path to reach the node and the estimated distance from the node to the goal. This is done by using a heuristic function that estimates the total cost from the starting node to the goal through the current node. A* algorithm takes into account both the cost of the path and the estimated distance to the goal, and as a result, it is guaranteed to find

the optimal path from the starting node to the goal, provided that the heuristic function satisfies certain conditions (such as being admissible and consistent).

In summary, Best First search and A* algorithm are both heuristic search algorithms, but A* algorithm takes into account the cost of the path to reach the current node in addition to the estimated distance to the goal, while Best First search only considers the estimated distance to the goal. Therefore, A* algorithm is generally considered to be more efficient and effective in finding the optimal path.

9. **Explain the A* Algorithm and its importance in pathfinding.**

   Ans:

   A* Search Algorithm is a simple and efficient search algorithm that can be used to find the optimal path between two nodes in a graph. It will be used for the shortest path finding. It is an extension of Dijkstra's shortest path algorithm (Dijkstra's Algorithm).

   Like Dijkstra, A* works by making a lowest-cost path tree from the start node to the target node. What makes A* different and better for many searches is that for each node, A* uses a function f (n) f(n) f(n) that gives an estimate of the total cost of a path using that node.
   Path finding algorithms are important because they are used in applications like google maps, satellite navigation systems, routing packets over the internet. The usage of pathfinding algorithms isn't just limited to navigation systems. The overarching idea can be applied to other applications as well.
   This optimization results in shortest paths being found more quickly. The A* algorithm can be used to find shortest paths between single pairs of locations, where GPS coordinates are known.

10. **What is AO* algorithm and how does it work?**

    Ans:

    The AO* method divides any given difficult problem into a smaller group of problems that are then resolved using the AND-OR graph concept. AND OR graphs are specialized graphs that are used in problems that can be divided into smaller problems. The AND side of the graph represents a set of tasks that must be completed to achieve the main goal, while the OR side of the graph represents different methods for accomplishing the same main goal.

    Working of AO* Algorithm:

    The evaluation function in AO* looks like this:
    f(n) = g(n) + h(n)
    f(n) = Actual cost + Estimated cost
    here,
    f(n) = The actual cost of traversal.
    g(n) = the cost from the initial node to the current node.
    h(n) = estimated cost from the current node to the goal state.

11. **Explain the Hill Climbing algorithm and its advantages and disadvantages.**

    Ans:

    Hill Climbing Algorithm

    1.It is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.

2.Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of Hill climbing algorithm is Traveling- salesman Problem in which we need to minimize the distance traveled by the salesman.
3.It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that.
4.A node of hill climbing algorithm has two components which are state and value.
5.Hill Climbing is mostly used when a good heuristic is available.
6.In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state.

Advantages Of Hill Climbing:
1. Hill Climbing can be used in continuous as well as domains.
2. This technique is very useful in job shop scheduling, automatic programming, circuit designing, and vehicle routing.
3. It is also helpful to solve pure optimization problems where the objective is to find the best state according to the objective function.

Disadvantages of Hill Climbing:
Hill Climbing suffers from the following problems –
1.Local Maxima:
It is a state which is better than all of its neighbours but isn't better than some other states which are farther away. At local maxima, all moves appear to make things worse. They are sometimes frustrating also as they often occur almost within slight of a solution. So, it is also called Foot-Hills.
2.Plateau:
It is a flat area of the search space in which a whole set of neighbouring states(nodes) have the same order. On a plateau, it is not possible to determine the best direction in which to move by making local comparisons. A plateau is an area of the state space landscape where the evaluation function is flat. It can be a flat local maximum or a shoulder from which it is possible to make progress.
3.Ridge:
It is a special kind of local maximum. it is an area of the search space which is higher than the surrounding areas and that itself has a slope. We can't travel the ridge by single moves as the orientation of the high region compared to the set of available moves makes it impossible.

12. What is Simulated Annealing and how is it different from Hill Climbing algorithm?
Ans:
A Simulated annealing algorithm is a method to solve bound-constrained and unconstrained optimization parameters models. The method is based on physical annealing and is used to minimize system energy.

In every simulated annealing example, a random new point is generated. The distance between the current point and the new point has a basis of the probability distribution on the scale of the proportion of temperature. The algorithm aims at all those points that minimize the objective

with certain constraints and probabilities. Those points that raise the objective are also accepted to explore all the possible solutions instead of concentrating only on local minima.
Optimization by simulated annealing is performed by systematically decreasing the temperature and minimising the search's extent.
Whereas Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbour has a higher value.

13. What is Local Beam search and how does it differ from other local search algorithms?
Ans:
Beam search is an optimization of best-first search that reduces its memory requirements. Best-first search is a graph search which orders all partial solutions (states) according to some heuristic. But in beam search, only a predetermined number of best partial solutions are kept as candidates.
In the context of a local search, we call local beam search a specific algorithm that begins selecting β generated states. Then, for each level of the search tree, it always considers β new states among all the possible successors of the current ones until it reaches a goal.
Since local beam search often ends up on local maxima, a standard solution is to choose the next β states in a random way, with a probability dependent on the heuristic evaluation of the states. This kind of search is called stochastic beam search.

14. Explain Genetic Algorithms and how do they mimic the process of natural selection?
Ans:
A genetic algorithm is a search heuristic 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.

Notion of Natural Selection:

The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found.
This notion can be applied for a search problem. We consider a set of solutions for a problem and select the set of best ones out of them.

Five phases are considered in a genetic algorithm -

1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

15. **What are Adversarial search methods and how do they relate to game playing?**

Ans:

Adversarial search is a search, where we examine the problem which arises when we try to plan ahead of the world and other agents are planning against us.

1.In previous topics, we have studied the search strategies which are only associated with a single agent that aims to find the solution which often expressed in the form of a sequence of actions.

2.But, there might be some situations where more than one agent is searching for the solution in the same search space, and this situation usually occurs in game playing.

3.The environment with more than one agent is termed as multi-agent environment, in which each agent is an opponent of other agent and playing against each other. Each agent needs to consider the action of other agent and effect of that action on their performance.

4.So, Searches in which two or more players with conflicting goals are trying to explore the same search space for the solution, are called adversarial searches, often known as Games.

5.Games are modeled as a Search problem and heuristic evaluation function, and these are the two main factors which help to model and solve games in AI.

Types of Games in AI:

Perfect information:

A game with the perfect information is that in which agents can look into the complete board. Agents have all the information about the game, and they can see each other moves also. Examples are Chess, Checkers, Go, etc.

Imperfect information:

If in a game agents do not have all information about the game and not aware with what's going on, such type of games are called the game with imperfect information, such as tic-tac-toe, Battleship, blind, Bridge, etc.

Deterministic games:

Deterministic games are those games which follow a strict pattern and set of rules for the games, and there is no randomness associated with them. Examples are chess, Checkers, Go, tic-tac-toe, etc.

Non-deterministic games:

Non-deterministic are those games which have various unpredictable events and has a factor of chance or luck. This factor of chance or luck is introduced by either dice or cards. These are random, and each action response is not fixed. Such games are also called as stochastic games. Example: Backgammon, Monopoly, Poker, etc.

16. **How is game playing modeled as a search problem?**

Ans:

A game can be defined as a type of search in AI which can be formalized of the following elements:

1.Initial state: It specifies how the game is set up at the start.
2.Player(s): It specifies which player has moved in the state space.
3.Action(s): It returns the set of legal moves in state space.
4.Result(s, a): It is the transition model, which specifies the result of moves in the state space.
5.Terminal-Test(s): Terminal test is true if the game is over, else it is false at any case. The state where the game ends is called terminal states.
6.Utility(s, p): A utility function gives the final numeric value for a game that ends in terminal states s for player p. It is also called payoff function. For Chess, the outcomes are a win, loss, or draw and its payoff values are +1, 0, ½. And for tic-tac-toe, utility values are +1, -1, and 0.

17. Explain the Mini max approach and its significance in game playing.

Ans:

1.Mini-max algorithm is a recursive or backtracking algorithm which is used in decision- making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
2.Mini-Max algorithm uses recursion to search through the game-tree.
3.Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac- toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.
4.In this algorithm two players play the game, one is called MAX and other is called MIN.
5.Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
6.Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
7.The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
8.The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

18. What is the Mini max algorithm and how does it work?

Ans:

The minimax algorithm helps find the best move, by working backwards from the end of the game. At each step it assumes that player A is trying to maximize the chances of A winning, while on the next turn player B is trying to minimize the chances of A winning (i.e., to maximize B's own chances of winning).
A minimax algorithm is a recursive program written to find the best gameplay that minimizes any tendency to lose a game while maximizing any opportunity to win the game. Graphically, we can represent minimax as an exploration of a game tree's nodes to discover the best game move to make.

19. Explain the Alpha beta pruning technique and how it optimizes the Mini max algorithm.

Ans:

Alpha-Beta Pruning:

1.Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.

2.As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. 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.

3.Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.

4.The two-parameter can be defined as:

a. Alpha: The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is -∞.

b. Beta: The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is +∞.

5.The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

20. What are some game theory problems that can be solved using search algorithms?

Ans:

Search algorithms can be used to solve various game theory problems, including:

1. **Two-player games:** Games like Tic-Tac-Toe, Connect Four, and Chess can be modeled as search problems, where the players take turns making moves, and the goal is to reach a winning position. Search algorithms such as Minimax or Alpha-Beta pruning can be used to find the optimal move for each player, assuming that both players play optimally.
2. **Optimal strategies:** Search algorithms can be used to find the optimal strategies for players in games such as poker or blackjack, where the optimal strategy depends on the cards that have been dealt and the actions taken by other players.
3. **Social dilemmas:** Social dilemmas are situations where individual rationality leads to a suboptimal outcome for the group. Search algorithms can be used to find solutions to social dilemmas such as the prisoner's dilemma or the tragedy of the commons.
4. **Auctions:** Auctions are a common mechanism for allocating resources in various contexts. Search algorithms can be used to find the optimal bidding strategies for players in various auction formats, such as first-price sealed-bid or ascending-price auctions.

Overall, search algorithms can be a useful tool for solving various game theory problems and finding optimal solutions in strategic interactions.