SEARCH STRATEGIES: INFORMED SEARCH STRATEGIES

Abstract

Search Algorithms in Artificial Intelligence are widely in use in present day user-based applications. These strategies actually gives resolution to every problem or search command given by the user by understanding command, evaluating existing condition, developing possible alternatives and assisting AI agents to provide best possible resolution to the concern. Thus, by performing search functions and providing resolution, AI agents make artificial intelligence easy and useful. Informed Search Algorithm applies Heuristic application to find the most appropriate route or solution. In this chapter, we have discussed applicability of Search algorithms and then discussed in detail Informed Search Algorithms, its type, utility, examples and its application in present day time.

Key Words:- Search Algorithms, Optimal Solution, Uninformed (Blind) Search, Informed (Heuristic) Search algorithms, greedy search, Best First Search Algorithm, A* Search Algorithm

Authors

Prof. (Dr.) Sangeeta
Professor
Department of Political Science
Shaheed Bhagat Singh College
University of Delhi
misrasangeeta@yahoo.co.in

Mr. Vinayak Rai
(Doing M.Com)
University of Delhi
vinayakrai219@gmail.com

Dr. Nitish Pathak
Bhagwan Parshuram Institute of Technology
New Delhi, India
nitishpathak@bpitindia.com

Dr. Neelam Sharma
Department of Artificial Intelligence & Machine Learning,
Maharaja Agrasen Institute of Technology
New Delhi, India.
neelamsharma@mait.ac.in

I. INTRODUCTION

Search Strategies or Search Algorithms in Artificial Intelligence are algorithms that help to resolve search problems that generally include search space, start and goal state. Whenever any command is given, search algorithms in AI evaluates scenarios and possible alternatives and assists AI agents to provide best possible resolution to the problem. We can thus say the search algorithms helps AI agents to reach the goal state through the assessment and evaluation of the existing situations and possible alternatives by providing search solutions through a sequence of actions that helps in transforming the initial problem state to goal state. Without utilizing these algorithms, it is not possible for AI machines and applications to initiate search functions and suggest appropriate solutions. Thus, by performing search functions and providing resolution, AI agents make artificial intelligence easy and useful.

Building agents with rational behaviour are the main problem-solving agents in artificial intelligence study and research work. These agents generally utilise search algorithm in the background application to do the task. These techniques are now most commonly used and universally applied problem-solving approach in Artificial Intelligence particularly to resolve a problem and propose the best result.

Some generally used Search Algorithms terms are explained as under:

1. **Search:** That refers to solving search problem stepwise in a given space. It can be influenced by:

    - A search space which is a collection of possible resolutions of a problem a system might be having.
    - Start State that refers to the jurisdiction where the agent are directed or instructed to start the search.
    - Goal Test which is about the operation or function that studies and examines the current state and returns irrespective of whether goal state has been attained or not.

2. **Search Tree:** It refers to statement of search issue in tree representation. The node present at the bottom or start point of the search tree displays the preliminary or initial condition.

3. **Actions:** It refers to explaining all the stages, actions, or operations accessible to the agent.

4. **Transition Model:** It is generally utilized to convey a portrayal of what change or impact each step or action brings.

5. **Path Cost:** It refers to the AI function that gives a statement of cost applicable to each path.

6. **Solution:** It is generally a statement of step-by step action from the start node till it reaches the target node.

7. **Optimal Solution:** It refers to the low-cost solution suggested by AI among all the possible solutions.

II. FEATURES OF SEARCH ALGORITHMS

The imperative features of search algorithms in AI are listed as under:

1. **Completeness:** In AI, a search algorithm is considered to be complete only when there is assurance of reaching appropriate solution for any random problem or input in a situation where there exists not less than one solution for the given problem or input.
2. **Optimality:** Generally in AI, a solution selected for an algorithm is taken as optimal only if it has capability to give best resolution of the search question at lowest path cost amid all other possible solutions.
3. **Time Complexity:** It refers to the time taken by an algorithm to complete the job and this depends greatly on the complexity of task entered.
4. **Space Complexity:** It is about the maximum memory or storage space required to run the search application which generally depends upon the complexity involved in resolving the search problem.

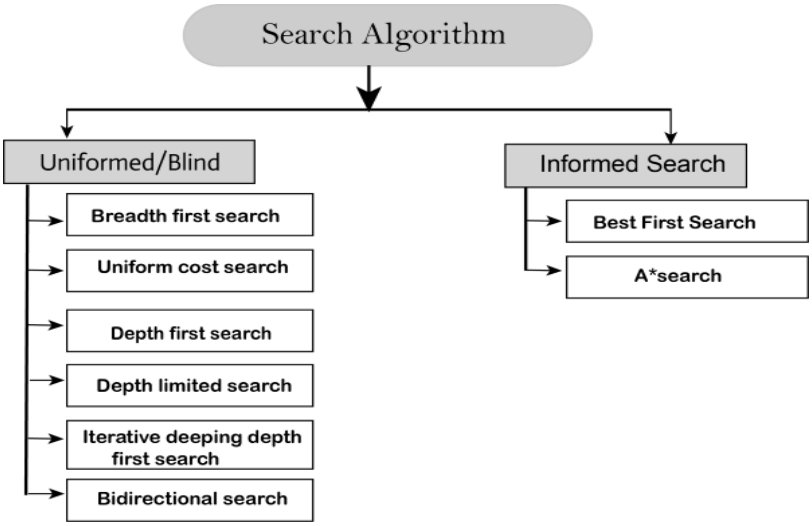
Importance of Search Algorithms in AI

- **Solving Problems:** Search algorithms uses its logical search mechanisms like "**problem description, actions, and search space"** and enhances its problem-solving capability in artificial intelligence. This way it helps user to use user-friendly applications especially for route planning like Google Maps. Such applications uses search algorithms to provide route chart with clear provision for both quickest and also shortest path between two geographical locations.
- **Search Programming:** Most of the Artificial Intelligence task can be coded in terms of searching mechanisms, that boosts the devising of applicable resolution to a given search problem.
- **Goal-Based Agents:** In AI, Search Algorithms are used to improve and enhancing the efficiency of goal-based agents. These agents develop most optimal series of action to resolve problems in a given situation.
- **Support Production Systems:** Search algorithms in AI applications helps in running production systems by referring to and applying stated rules and following defined procedures for putting them into application. In the process, production system uses search algorithms to apply prescribed rules that can help adopt required or suitable action.
- **Neutral Network System:** The 'neutral network systems' are the computing systems that comprises of a 'hidden layer', 'an input layer', 'an output layer' and 'coupled nodes'. They are generally utilized to execute many tasks or jobs in AI. This way search algorithms improves the finding the 'connection weights' which results in developing the requisite 'input-output mapping'.

Types of Search Algorithms in AI

Search Algorithms in AI can be divided into below stated algorithms:

- Uninformed (Blind) Search algorithms, and
- Informed (Heuristic) Search algorithms on the bases of search issue.



Uninformed or Blind Search Algorithms

The uninformed or blind search algorithms requires certain basic information like goal location as it does not have any domain information. It operates in a 'brute-force way'. This is because it only has the information on way to 'traverse the tree and identify leaf and goal nodes' only. This search strategy applies only a way or method by which search tree can be found without any information regarding the search space like initial state operators and the command about required test for the goal. As such, this strategy has also been named as the Blind Search. In this type, all the nodes of the tree is examined till the goal node is found or achieved. These algorithms are further categorized into below mentioned algorithms:

- Breadth-first Search
- Depth-first Search
- Uniform Cost Search
- Iterative Deepening depth-first Search
- Bidirectional Search

Informed Search Algorithms or Heuristic Search

"Heuristics means norms, procedures or principles that helps in deciding which amongst the several available alternative choice of action promises to be the most effective mean to achieve some predetermined goal".

Informed Search Algorithm uses Heuristic function to find the most appropriate route. While doing so, it considers the present state of the agent as the starting point and produces an approximation of gap between agent and the goal. The Solution hence suggested by this method may or may not always be the best, but it definitely tries to suggest good or we may say decent solution in comparatively much reasonable time. Heuristic function examines the gap between the state and the goal and the outcome is denoted by h(n). It computes the cost

of an optimal path between the given pair of states. The value of the heuristic function hence computed is always positive.

Admissibility of the heuristic function can be depicted as:

h(n) <= h*(n)

Where,

h(n) = heuristic cost
h*(n) = estimated cost.

In this, heuristic cost must be either less than or equivalent to the projected cost.

Pure Heuristic Search

This is the simplest form of heuristic search algorithms that works by expanding nodes on the bases of their heuristic value i.e. h(n). In this, two lists - Open and Closed are maintained wherein already expanded nodes are kept in Closed list whereas such nodes which have yet not been expanded are kept in Open list.

On each iteration, node 'n' having lowest heuristic value is expanded. After generating all its successors, node 'n' is placed in the closed list. The algorithm continues to proceed in this way till goal state is found. Here we can take the example of searching the shortest route between Kolkata and Guwahati. The heuristic in this case will be straight-line between Kolkata and Guwahati. The result hence searched will be:

$h(Kolkata) = \boldsymbol{euclideanDistance}(\boldsymbol{Kolkata},\boldsymbol{Guwahati})$
The informed search strategies uses three main algorithms. These are:

- Best First Search Algorithm (Greedy search)
- A* Search Algorithm
- Hill Climbing

Now we will discuss in detail these three main algorithms.

III. BEST-FIRST SEARCH ALGORITHM

This is also known as GREEDY SEARCH. This search algorithm picks such path that appears best at that moment. This algorithm is a blend of both - depth-first search and breadth-first search algorithms and in its operations, it hence take the benefits of both the algorithms. This algorithm is preferred as it helps choose the most promising node at each step. Here, the node which is closest to the goal node is expanded and the closest cost is estimated by heuristic function, i.e.

h(n)= g(n).

Where,
h(n) = estimated cost from node n to the goal.

The greedy best first algorithm is applied by the priority queue.

Steps followed in Best first search algorithm is stated as under:

- **Step 1:** The process begins once the start node is placed into the OPEN list.
- **Step 2:** In case the OPEN list is empty, Stop and return failure.
- **Step 3:** Remove the node n (that has the lowest value h(n)) from the OPEN list, and place it in the CLOSED list.
- **Step 4:** Expand the node n, and generate the successors of node n.
- **Step 5:** Check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node, then return success and terminate the search, else proceed to Step 6.
- **Step 6:** For each successor node, algorithm checks for evaluation function f(n), and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both lists, then add it to the OPEN list.
- **Step 7:** Return to Step 2.

Advantages

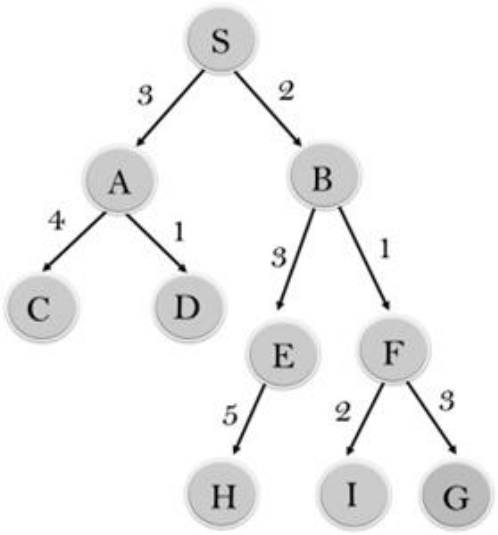
- Since this algorithm is a combination of both- Breadth-First Search and Depth-First Search, the benefit of using this method lies in the fact that it can switch between both Breadth-First Search and Depth-First Search by gaining the advantages of both the algorithms.
- This algorithm is more effective and capable than Breadth-First Search and Depth-First Search algorithms.

Drawbacks

- In worst scenario, this algorithm can operate as an unguided Depth-First Search.
- It sometimes gets stuck in a loop as Depth-First Search.
- This algorithm is not best and optimal.
- Its disadvantage can also be explained by imagining a situation in a simple game where the goal can be reached by reaching a specific location on the board. Player has been allowed to move in any direction but there are hurdles that has created by putting walls and blocking some paths. By following greedy search approach, player will always select shortest path without considering without taking into account the potential obstacles that has been placed or the fact that dead ends has been placed on some paths. In case, chosen path takes the player to dead end or a loop, algorithm will instead of re-routing the path, will keep on moving back and forth between the same nodes. It will also not try other available options. This way solution can never be reached.

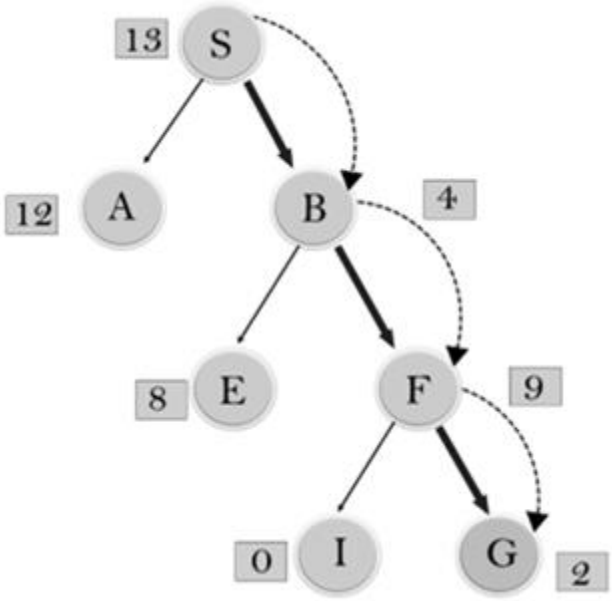
Example

Here we can take below stated search problem in example. In this case, it has been traversed by using greedy best-first search. At each iteration, each node is expanded using evaluation function f(n)=h(n), which is given in the below table[1].



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

In this search example, two lists - OPEN and CLOSED Lists has been used. Following are the iteration for traversing the above example.[2]



[1] This example has been taken from https://rcet.org.in
[2] Ibid.

Expand the nodes of S and put in the CLOSED list
Initialization: Open [A, B], Closed [S]
Iteration 1: Open [A], Closed [S, B]
Iteration 2: Open [E, F, A], Closed [S, B] : Open [E, A], Closed [S, B, F]
Iteration 3: Open [I, G, E, A], Closed [S, B, F] : Open [I, E, A], Closed [S, B, F, G]
Hence the final solution path will be: S----> B----->F----> G

Time Complexity: In the worst situation, the time complexity of this Greedy best first search is O(bm).

Space Complexity: In the worst situation, the space complexity of Greedy best first search is O(bm). Where, m is the maximum depth of the search space.

Complete: Generally, the Greedy best-first search is incomplete even when the given state space is finite.

Optimal: Greedy best first search algorithm is also not optimal.

IV. A* SEARCH ALGORITHM

This Search Algorithm is the most commonly identified form of best-first search. This method applies heuristic function h(n), and cost to reach the node n from the start state g(n). It has joint features of both UCS and greedy best-first search and this helps it in solving the problem effectively. This search algorithm searches the shortest available path through the search space by utilising the heuristic function, expands only few search tree and hence provides optimal result much faster. This algorithm is very much similar to UCS except the calculation part where it uses g(n)+h(n) instead of g(n).

Here since both search heuristic as well as the cost to reach the node is utilized, we can take a combination both the costs (as stated below), and the sum hence arrived is called fitness number f(n).

$$\mathbf{f(n) = g(n) + h(n)}$$

Where,

f(n) is the estimated cost of the most optimal solution,
g(n) represents the cost calculated to reach node n from start state, and
h(n) re[resents the cost calculated to reach from node n to goal node.

Here it has to be noted that at each step in the search space, only the node having lowest value f(n) is expanded, and once the goal node is reached, the algorithm gets terminated.

Algorithm used in A* search[3]

- **Step1:** The process starts once the starting node is placed in the OPEN list.

[3] Ibid.

- **Step 2:** Check if the OPEN list is empty or not, if the list is empty then return failure and stops.
- **Step 3:** Select the node from the OPEN list which has the smallest value of evaluation function (g+h), if node n is goal node then return success and stop, otherwise
- **Step 4:** Expand node n and generate all of its successors, and put n into the closed list. For each successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute evaluation function for n' and place into Open list.
- **Step 5:** Else if node n' is already in OPEN and CLOSED, then it should be attached to the back pointer which reflects the lowest g(n') value.
- **Step 6:** Return to Step 2.

Advantages of applying this Search Algorithm

- This algorithm is the best search algorithm than other search algorithms.
- A* search algorithm is optimal and complete.
- This algorithm is capable to solve very complex problems also.

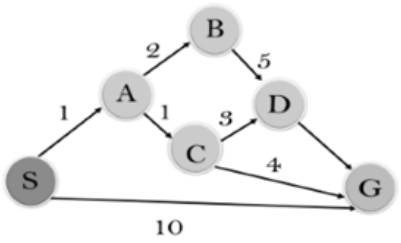
Disadvantages attached with this Search Algorithm

- This search algorithm does not always gives the shortest possible path as it is mostly based on heuristics and approximation.
- This search algorithm has some complexity issues.
- The main drawback of applying this algorithm is the memory required for the task as it keeps all generated nodes in the memory that occupies lots of space. This way it is not preferred to resolve large-scale problems.

Example:[4]

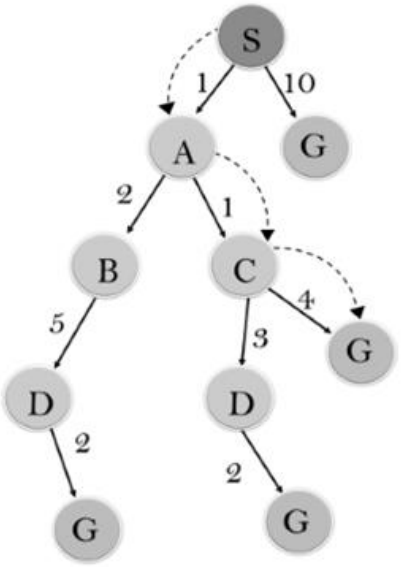
In this example, the given graph has been traversed utilising the A* search algorithm. The heuristic value hence calculated for all the states is stated in the table (given below) using which f(n) of each state can be calculated by applying the formula f(n)= g(n) + h(n), where g(n) is the cost to reach any node from start state. In this example, OPEN and CLOSED list has been used.

[4] https://rcet.org.in/



State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

Solution



Initialization: {(S, 5)}
Iteration1: {(S--> A, 4), (S-->G, 10)}
Iteration2: {(S--> A-->C, 4), (S--> A-->B, 7), (S-->G, 10)}
Iteration3: {(S--> A-->C--->G, 6), (S--> A-->C--->D, 11), (S--> A-->B, 7), (S-->G, 10)}
Iteration 4 will give the final result, as S--->A--->C--->G it provides the optimal path with cost 6.

Points to Remember

- A* algorithm always returns following the path which occurred first, and it does not search for all remaining paths.
- The efficiency and capability of A* algorithm depends on the quality of heuristic.
- A* algorithm expands all nodes that satisfies the condition f(n)<="" li="">

Complete: A* algorithm is complete as long as[5]:

- Branching factor is limited.
- Cost at every step is fixed.

Optimal: A* search algorithm is optimal if it follows below two conditions[6]:

- Admissibility: The first condition required for optimality is that h(n) should be an admissible heuristic for A* tree search. An admissible heuristic is optimistic in nature.
- Consistency: Second required condition is consistency for only A* graph-search.

If the heuristic function is admissible, then A* tree search will always find the least cost path.

Time Complexity: The time complexity of A* search algorithm depends a lot on the heuristic function, and the number of nodes expanded is exponential to the depth of solution d. So the time complexity is O(b^d), where b is the branching factor.[7]

Space Complexity: The space complexity of A* search algorithm is O(b^d).[8]

V. HILL CLIMBING

Belonging to the family of local search algorithms, hill climbing is a simple optimization algorithm generally used in AI to identify and give the optimal solution to a certain problem where objective is to choose best solution out of a set of possible solutions. This algorithm constantly moves in the direction of elevated terrain or improved value. It does so till it reaches a peak value where none of its neighbors have a greater value. Once it reaches there, the search ends.

In this method, search algorithm generally starts with an initial solution. Since its goal is to find the best possible solution out of the set of possible available solutions, it analyses the situation, makes small changes and improves the solution. This function is based on heuristic functions that enables assessment and evaluation of quality of solution at each step. This process of making small changes continues till it reaches a local maximum i.e. the point from where no further improvement is possible. This search is also known as greedy local search as it only conducts searches in its favorable immediate neighbor state only and does not go beyond. The two components generally used by this algorithm node are- State and Value.

[1] [5] https://www.geeksforgeeks.org/

[6] Ibid.
[7] Ibid.
[8] Ibid.

Types of Hill Climbing

- **Steepest Ascent Hill Climbing:** This algorithm first evaluates all the existing neighbouring nodes. It then chooses the one that is closest to the preferred solution state and leads to the best improvement from the existing state.
- **First-Choice Hill Climbing:** In this the search algorithm randomly chooses a move that may lead to an improvement from the current one irrespective of the fact whether it leads to the selection of the best one or not.
- **Stimulated Annealing:** This is a probabilistic algorithm that sometimes selects even the worst one to avoid getting struck in local maxima and moves ahead to find goal.

Features of Hill Climbing Algorithm

- **Greedy Strategy:** Since this algorithm directs its search to select the solution where the cost is optimized, it is sometimes called Greedy Strategy,
- **No Backtracking:** In this, the search algorithm makes improvement and moves ahead, it cannot recall its past states. As such, in this method it is not possible to move back into the previous or any other search space.
- **Generate and Test Variant:** The Generate and Test method is based on this method only. Feedback received from this Test approach helps in selecting which way to move through the search space.

Advantages of Hill Climbing

- Simple, intuitive and straight forward: This search algorithm is preferred for its being simple and straight-forward that is easy to understand and implement and its accuracy in solving problem and giving optimal result.
- Memory efficiency: This algorithm is memory-efficient as it has to maintain only current state’s date.
- Rapid and Swift Convergence: to solution makes this search algorithm useful especially when time is crucial.
- Easily Modifiable: This algorithm may easily be changed and extended to contain additional heuristics or constraints.

Weaknesses in using Hill Climbing

- **Susceptibility to Local Optima:** Sometimes these search algorithms get stuck at locally optimal solutions which is always not good and preferred.
- **Limited exploration:** This feature restricts search algorithm to improvise according to neighbouring or available options and provides best solution from that range only which limits its search capability to local optima only and limits its exploration which is not always desirable and helpful.
- **Dependence on Initial State:** In this method, the initial step begins with a randomly selected solution. It has often been found that quality and effectiveness of the solution depends a lot on the first value chosen.

Advantages of Using Informed Search Strategies

This strategy is preferred over Blind Search strategies because of the speed and accuracy of the result it delivers. Some of the advantages of this strategy is stated as under:

- **Faster convergence:** This search strategy expands nodes with lowest heuristic value only. This way informed search strategy avoids or prunes large unproductive areas of the state space and leads to goal at much shorter span of time.
- **Optimal Solutions:** These algorithms like A* are programmed in such a way that they deliver optimal result within given admissible heuristics. These two features i.e. speed and accuracy makes the strategy beneficial especially for mission-critical applications.
- **Efficiency:** expansion of limited number of nodes reduces the task of the of the search strategy and helps them utilise computational resources very economically. This allows application on resource-constrained devices.
- **Scalability:** The ability of these strategies to leverage heuristics helps informed search resolve much complicated problems with large search spaces and complexity.
- **Directed probing:** Heuristics or informed search strategy enable searching in the most relevant areas of the space. This qualitative enhancement in search strategies has made them invaluable and unavoidable in resolving complex real-world problems.
- **Tractability:** Its capability to resolve complex problems with speed and accuracy has made AI application much easier particularly in game playing, route mapping, etc. We can hence say here ‘Heuristics tame complexity.

Disadvantages of using Informed Search Strategies

Informed search strategies has certain disadvantages and these are listed as under:

- **Sub-optimal Heuristics:** For optimal result it is essential that heuristic should be perfect. Poor heuristics can grossly misrepresent actual distances and may also mislead search. This vitiates performance of these search strategies greatly.
- **Design Complexity:** Developing high-quality heuristics requires indepth and extensive domain knowledge, expertise and insight. This makes scaling informed search to a new problem much tougher.
- **Overhead Cost:** The process requires computation of complex heuristic values at each node only that adds overhead cost to the search process and to some degree slows it down. As such we can say that ‘Simpler heuristics are more efficient’.
- **No guarantee of Optimality**: Unlike other algorithms like A*, Greedy Searches often settles for suboptimal solutions because they often get trapped using misleading heuristics. As such no guarantee can be given for their optimality.
- **Lack of Exploration:** These days we finds over-reliance or we can say over-dependence of people on heuristics. This attitude is actually preventing discovery of novel solutions. Hence we can say- ‘Balancing exploration is key’.
- **Problem Specificity**: Heuristics are problem specific only. They are usually designed to resolve one problem only and cannot be transferred over to new domains. Here we can cay- ‘Reusability is limited’.

Comparison between Informed Search and Uninformed Search

The comparison between the two is stated in the table below:

Aspect	**Informed Search Algorithm**	**Uninformed Search Algorithm**
Search Strategy	These are directed by heuristic information and knowledge	This strategy begins its search without any domain information or we may say blindly and lacks heuristic assistance
Heuristic Function	This search algorithm utilises heuristic functions to calculate the cost from state to reach the goal	This search method does not use heuristic functions
Efficiency	Being informed, this search strategy is much more effective in exploring fewer states	This search strategy may explore a greater number of states also
Completeness	Following this strategy there is always no guaranteed assurance that a desired solution can be reached or not	This strategy will always find the best of the available solution that exists
Optimality	This strategy are programmed to find and suggest optimal solutions from the available options with the application of heuristics.	This strategy does not help in searching optimal solutions
Example algorithm	• A*, • Best-first search, and • Hill climbing	• Breadth-first search, • Depth-first search, and • Uniform cost search
Use Cases	This strategy is generally preferred for resolving complex problems with heuristic information	This strategy is helpful when there is simple problem or when the heuristic function is not available

Application of Informed Search Strategies in Real World

This Search Algorithm has been used by many different domains. They utilise heuristic application and data to begin their search to give speedy and accurate solutions. The application now-a-days is very much in use and some such commonly used applications based on heuristic method are listed as under:

- **Navigating by Pathfinding:** Daily use applications like ‘GPS systems’ and ‘mapping applications’ often uses informed search algorithms in finding shortest and swiftest route between two given points by analysing situation on road along with current traffic condition and hence assists in route planning. Now-a-days this strategy has become unavoidable. Even a layman who has no idea of AI technicalities are using this strategy while doing route planning.

- **Playing a Game:** Now-a-days every mobile and computer system has different games. Besides these can be played on internet also. Game developers use informed search strategies in developing their games. These games (for example board games like chess, ludo, checkers, etc.) has playing agents that applies informed search algorithms to achieve the objective. In this they generally utilises search applications like minimax with alpha-beta pruning and heuristic-based evaluation functions to decide next action and plan further actions.

- **Vehicle Autonomy and Robotics: Present day vehicles and robots are using this search strategy** for operations like path following or for avoiding obstacles on the route or for doing motion planning. It helps robots efficiently navigate difficult conditions. At present this strategy is used in route navigation, cloud services, healthcare, virtual assistance and many others. Besides applications like self-driving car or auto-mode in vehicles uses A* search algorithms with spatial heuristics for navigation and route planning,

- **Timetabling and Scheduling:** This strategy also helps in planning work and placing staff at different tasks. As such we can say that this strategy is helpful in office planning and class planning (in bigger coaching institutes) also. The strategy has been utilised in scheduling applications like airline scheduling by examining weather condition and air route traffic, staff scheduling, vehicle scheduling (used extensively by cab services like ola, etc.), personnel scheduling and also in class scheduling. Shorter make-span has made these applications cost-effective and service-effective. The strategy hence has been proved helpful in enhancing resource provision and allocation and reduce clashes in planning applications.

- **Routing on a Network:** The strategy has been used widely in computer networks in selecting the best out of the possible paths for data packet while accounting for network latency and congestion.

- **Bioinformatics:** Informed search strategy is utilised in protein folding and drug discovery applications in Physics and Chemistry- based heuristics to model molecular interactions.

- **Machine Translation:** This strategy is also used in doing higher quality language translations also by utilising beam search with linguistic heuristics.

VI. CONCLUSION

After studying features and applications of Informed Search Algorithms it can be said that in present times these algorithms has become vital in Artificial Intelligence. By using heuristic driven guidance they have not only enhanced the sufficiency of goal-oriented searches but also have eased their application. In current day daily life we are seeing its application and utilization everywhere whether it is about tour planning, class fixing or organising, data collection, and many more. The usage of heuristic function has helped in resolving these problems much quicker and with accuracy.

REFERENCES

[1] Elaine Rich and Kevin Knight, “Artificial Intelligence”, McGraw-Hill.
[2] E Charniak and D McDermott, “Introduction to Artificial Intelligence”, Pearson.
[3] https://www.analyticsvidhya.com/blog
[4] https://www.egyankosh.ac.in/bitstream/
[5] https://www.databasetown.com/informed-search-strategies-in-artificial-intelligence/
[6] https://www.geeksforgeeks.org/
[7] https://www.javatpoint.com/ai-informed-search-algorithms
[8] https://www.geeksforgeeks.org/best-first-search-informed-search/
[9] https://intellipaat.com/blog/
[10] https://udrc.lkouniv.ac.in/
[11] INFORMED SEARCH ALGORITHMS (HEURISTIC SEARCH), https://rcet.org.in /uploads/academics/regulation2021/rohini_22798218887.pdf
[12] Search Algorithms in Artificial Intelligence, https://www.javatpoint.com
[13] Stuart Russell, Peter Norvig, “Artificial Intelligence-A Modern Approach”, Pearson.

