**Chapter 7**

**Search Strategies: Informed Search Strategies**

Prof. (Dr.) Sangeeta, Professor,

Department of Political Science,

Shaheed Bhagat Singh College,

University of Delhi

Mr. Vinayak Rai (Doing M.Com),

University of Delhi

**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 uses Heuristic function to find the most appropriate route or solution. In this chapter, we have discussed applicability of Search algorithms and than 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

**Introduction**

Search Strategies or Search Algorithms in Artificial Intelligence are algorithms that helps in resolution of search problems that generally includes 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 concern. We can thus say the search algorithms helps AI agents to attain the goal state through the assessment of scenarios and alternatives by providing search solutions through a sequence of actions that transform the initial state to goal state. Without these algorithms it is not possible for AI machines and applications to implement search functions and find 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 research. These agents often use search algorithm in the background to do the task. These techniques are now most commonly used and universally applied problem-solving approach in Artificial Intelligence to solve a particularly problem and provide the best result.

Some Generally Used Search Algorithms Terminologies:

1. **Search-** That refers to solving search issue in a given space step-by-step. It can be influenced by:

* A search space which is a collection of potential solutions a system might be having.
* Start State that refers to the jurisdiction where the agent starts the search.
* Goal Test which is about the operation or function that examines the current state and returns whether or not the goal state has been attained.

1. **Search Tree-** that refers to statement of search issue in tree representation. The node present at the bottom of the search tree corresponds to the initial condition.
2. **Actions-** It refers to explaining all the steps, activities, or operations accessible to the agent.
3. **Transition model-** that is generally used to convey a portrayal of what each action does.
4. **Path Cost-** It is a function that gives a statement of cost to each path.
5. **Solution-** which is generally an action sequence that connects the start node to the target node.
6. **Optimal Solution-** refers to the solution at the lowest cost among all the solutions.

**Properties of Search Algorithms**

There are four imperative properties of search algorithms in artificial intelligence. These are:

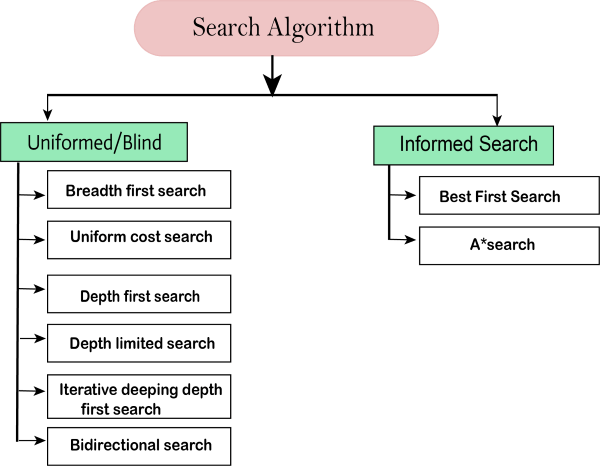
1. **Completeness**- A search algorithm is said to be complete only if there is assurance of reaching appropriate solution for any random input when there exists at least one solution for the given input.
2. **Optimality**- A solution selected for an algorithm is considered optimal only if it has capability to give best solution at lowest path cost among all other 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**- which is about the maximum memory or storage space essential during the search as is determined by the problem’s complexity.

**Importance of Search Algorithms in AI**

* **Solving Problems**: Search algorithms uses its logical search mechanisms like problem description or definition, 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 or programmed in terms of searching mechanisms, which boosts the devising of applicable solution to a given problem.
* **Goal-based agents:** Search Algorithms in AI helps in improving 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 using rules and methods for putting them into practice. In the process, production system uses search algorithms to find the rules that can lead to the required 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. These networks are used to perform many tasks in AI. This way search algorithms improves the searching of connection weights that results in developing the requisite input-output mapping.

**Types of Search Algorithms in AI**

Search Algorithms in AI can be divided into: Uninformed (Blind) Search and Informed (Heuristic) Search algorithms on the bases of search issue.



**Uninformed or Blind Search Algorithms**

The uninformed or blind search needs domain information like goal location as it does not contain any domain knowledge. It operates in a brute-force way as it only has the information on way to traverse the tree and identify leaf and goal nodes. Since uninformed search applies only a way in which search tree is searched without any information about the search space like initial state operators and test for the goal, it is hence also named as the blind search. In this type, each node of the tree is examined until the goal node is 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 are criteria, methods or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal”.

Informed Search Algorithm uses Heuristic function to find the most appropriate route. While doing so, it takes the present state of the agent as its input and produces the approximation of how close agent is from the goal. The Solution hence suggested by this method, however, might not always be the best, but it definitely tries to suggest good solution in reasonable time. Heuristic function estimates how close a state is to the goal and the outcome is represented by h(n). It calculates the cost of an optimal path between the pair of states. The value of the heuristic function is always positive.

Admissibility of the heuristic function can be depicted as:

**h(n) <= h\*(n)**

Where h(n) is heuristic cost, and

h\*(n) is the estimated cost.

In this, heuristic cost should be less than or equal to the estimated cost.

**Pure Heuristic Search:**

This is the simplest form of heuristic search algorithms that works by expanding nodes based on their heuristic value 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, each node n with the lowest heuristic value is expanded and generates all its successors and n is placed to the closed list. The algorithm continues till goal state is found. Here we can take the example of searching the shortest path from Kolkata to Guwahati, then heuristic for Guwahati may be straight-line distance between Kolkata and Guwahati, that is

ℎ(𝐾𝑜𝑙𝑘𝑎𝑡𝑎) = 𝒆𝒖𝒄𝒍𝒊𝒅𝒆𝒂𝒏𝑫𝒊𝒔𝒕𝒂𝒏𝒄𝒆(𝑲𝒐𝒍𝒌𝒂𝒕𝒂,𝑮𝒖𝒘𝒂𝒉𝒂𝒕𝒊)

In the informed search we will discuss two main algorithms which are given below:

* **Best First Search Algorithm (Greedy search)**
* **A\* Search Algorithm**

**BEST-FIRST SEARCH ALGORITHM**: This is also known as GREEDY SEARCH. This search algorithm selects such path which seems best at that moment. This algorithm is a combination of both - depth-first search and breadth-first search algorithms and in its operations, it hence take the advantages of both algorithms. This algorithm is preferred as it helps choose the most promising node at each step. Here, we expand the node which is closest to the goal node 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 implemented by the priority queue.

Best first search algorithm:

* Step 1: Place the starting node into the OPEN list.
* Step 2: If the OPEN list is empty, Stop and return failure.
* Step 3: Remove the node n, from the OPEN list which has the lowest value of h(n), and places 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:**

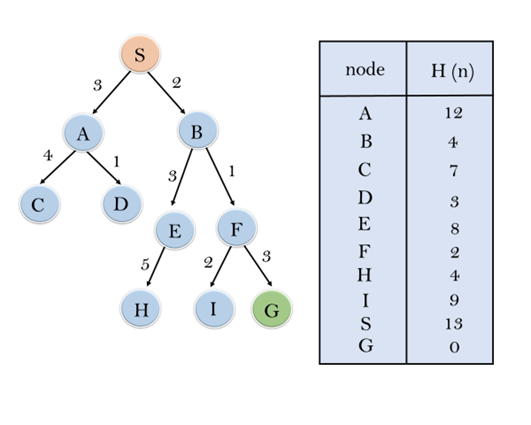
* Best first search can switch between Breadth-First Search and Depth-First Search by gaining the advantages of both the algorithms.
* This algorithm is more efficient than Breadth-First Search and Depth-First Search algorithms.

**Disadvantages:**

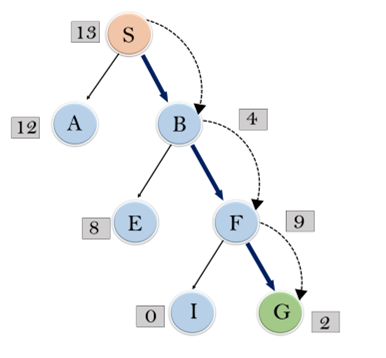
* It can behave as an unguided depth-first search in the worst case scenario.
* It can get stuck in a loop as Depth-First Search.
* This algorithm is not optimal.
* Its disadvantage can also be explained by imagining a situation in a simple game where the goal can be reached by reaching a specific location 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 choose shortest path without considering without taking into account the potential obstacles that has been placed or the fact that some paths may lead to dead ends. In case, chosen path leads to dead end or a loop, algorithm will keep moving back and forth between the same nodes without re-routing the path or trying other available options. This way solution can never be reached.

Example:

Here we can take below stated search problem in example (taken from https://rcet.org.in/). 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.



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



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**: The worst case time complexity of Greedy best first search is O(bm).

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

**Complete:** Greedy best-first search is also incomplete, even if the given state space is finite.

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

**A\* SEARCH ALGORITHM:** A\* Search Algorithm is the most commonly identified form of best-first search. This method uses heuristic function h(n), and cost to reach the node n from the start state g(n). It has joint features of UCS and greedy best-first search which helps it in solving the problem efficiently. This search algorithm finds the shortest path through the search space using the heuristic function. In the process, it expands less search tree and provides optimal result much faster. A\* algorithm is similar to UCS except that 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 combine both the costs as following, and this sum is called as a fitness number f(n).

**f(n) = g(n) + h(n)**

where,

f(n) is the estimated cost of the cheapest solution,

g(n) is the cost to reach node n from start state, and

h(n) is the cost to reach from node n to goal node.

**Here it has to be noted that at each point in the search space, only that node is expanded which have the lowest value of f(n), and the algorithm gets terminated once the goal node is found.**

Algorithm of A\* search:

* Step1: Place the starting node in the OPEN list.
* 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:**

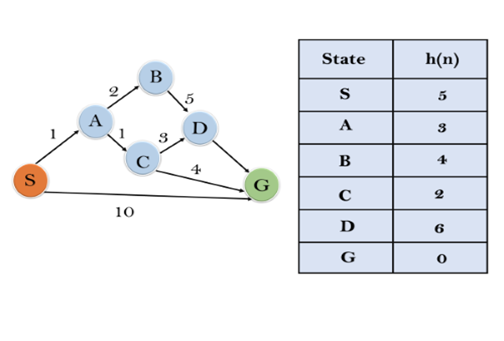
* A\* search algorithm is the best algorithm than other search algorithms.
* A\* search algorithm is optimal and complete.
* This algorithm can solve very complex problems.

**Disadvantages:**

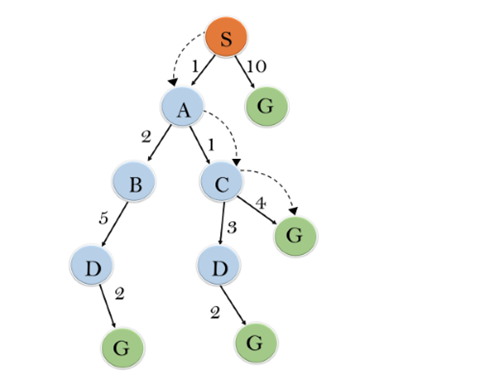
* It does not always produce the shortest path as it mostly based on heuristics and approximation.
* A\* search algorithm has some complexity issues.
* The main drawback of A\* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various large-scale problems.

**Example (taken from https://rcet.org.in/):**

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



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

**Complete**: A\* algorithm is complete as long as:

* Branching factor is finite.
* Cost at every action is fixed.

**Optimal**: A\* search algorithm is optimal if it follows below two conditions:

* Admissible: the first condition requires 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 on 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.

**Space Complexity**: The space complexity of A\* search algorithm is O(b^d)

**Hill Climbing**

Belonging to the family of local search algorithms, hill climbing is a simple optimization algorithm used in AI to identify and give the optimal solution to a given problem where objective is to choose best solution out of a set of possible solutions. This algorithm ‘perpetually advances in the direction of elevated terrain or increased value. When it reaches a peak value where none of its neighbors have a greater value, it ends’.

In this method, search algorithm generally starts with an initial solution. Since its goal is to find best solution out of set of possible 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 of this algorithm node are- State and Value.

**Types of Hill Climbing:**

* **Steepest Ascent Hill Climbing:** This algorithm first evaluates all the neighbouring nodes and then chooses the one that is closest to the solution state and leads to best improvement.
* **First-Choice Hill Climbing**: In this the algorithm randomly selects 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 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 choosing which way to move through the search space.

**Advantages of Hill Climbing:**

1. 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.
2. Memory efficiency: This algorithm is memory-efficient as it has to maintain only current state’s date.
3. Rapid and Swift Convergence: to solution makes this search algorithm useful especially when time is crucial.
4. Easily Modifiable: This algorithm can easily be modified and extended to include additional heuristics or constraints.

**Disadvantages of Hill Climbing:**

1. **Susceptibility to Local Optima:** Sometimes these search algorithms get stuck at locally optimal solutions which is always not good and preferred.
2. **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.
3. **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 guided by heuristic information or knowledge | This strategy explores blindly and lacks heuristic guidance |
| **Heuristic Function** | This search algorithm utilises heuristic functions to estimate the cost from state to goal | This search method does not use heuristic functions |
| **Efficiency** | Being informed, this search strategy is much more efficient in exploring fewer states | This search strategy may sometimes explore a large number of states also |
| **Completeness** | Following this strategy there is always no guarantee whether a solution exists or not | This strategy will always find the best solution that exists |
| **Optimality** | This strategy are programmed to find and suggest optimal solutions with the admissibility of heuristics. | This strategy does not find 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 ideally suitable 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 data to begin their search for speedy and accurate solutions. Some such common usage of an informed search is listed as under:

* **Navigating by Pathfinding**: 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 utilises informed search algorithms 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 activities like path following, obstacle avoidance, and 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 allocation and reduce conflicts in scheduling applications.
* **Routing on a Network**: The strategy has been used widely in computer networks in selecting the best paths for data packets 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.

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.