Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 7.056

IJCSMC, Vol. 10, Issue. 7, July 2021, pg.84 – 90

Finding Optimal Algorithm in Artificial Intelligence

Dr. I.Parvin Begum
Assistant Professor Department of Computer Science, M.M.E.S Women's Arts and Science College, Melvisharam, Tamil Nadu, India
parvinnadiya@gmail.com

Dr. I.Shahina Begam
Assistant Professor Department of Computer Applications, M.M.E.S Women's Arts and Science College, Melvisharam, India
sbshahintaj@gmail.com
DOI: 10.47760/ijcsmc.2021.v10i07.012

Abstract— Present days many artificial intelligence search algorithms are plays a important to figure out the problem of shortest path finding. The paper presents the detailed study of heuristic search and blind search techniques. The paper focus additional in the direction of blind search strategies such as Breadth First Search, Depth First Search, and Uniform Cost Search and informed explore strategies like A*, and Best First Search. The paper consist of effective of search procedure, their qualities, and demerits, where these algorithms are applicable, also at last comparison of search techniques based on complexity, optimality and completeness are presented in tabular structure.

Keywords- Artificial Intelligence, path finding, blind search algorithm, working strategy, Uniform Cost Search

I. INTRODUCTION

Artificial Intelligence (AI) is a subdivision of computer science that aims to create a computer to feel like a person human being [1], [2]. At the present time, it is individual of the large amount trendy increasing tools in this tech-world. It resolves lots of of the valid world problems more competently. Method start is individual between the major troubles in Artificial Intelligence. As a result of with different penetrating algorithms and technique we can answer this difficulty [1].

Present are two types of AI technique which can be worn to resolve these kind of troubles. Those are, **uninformed search** algorithm and **informed search** algorithm. The uninformed search algorithm is also known as **blind search,** we don't have any information about the number of steps or the path costs from the current state to the goal [3].

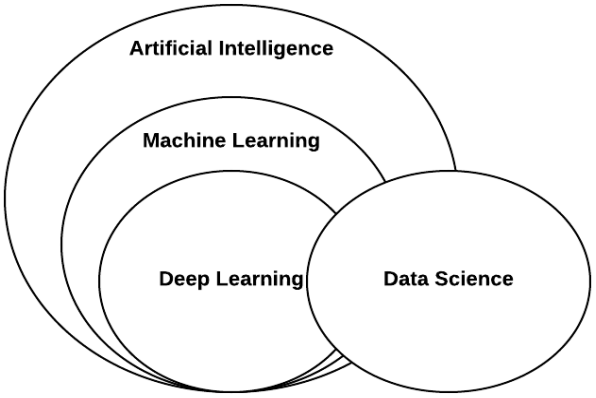

Fig.1. Artificial Intelligence with supplementary field

Artificial intelligence contain difficulty solve (Search techniques), Knowledge and Reasoning (Resolution, Knowledge Representation, Fuzzy Logic), Adversarial Search: Game Playing (Minimax Algorithm, Alpha-Beta Pruning), Uncertain Knowledge and Reasoning (Uncertainty, Probabilities, Bayesian Networks), and Expert system. The following paper presents more about problem-solving techniques in artificial intelligence. The paper focuses more on BFS, DFS, UCS, A*, and Best First Search.

II. SEARCH ALGORITHM APPROACHES

A. **Uninformed search**

A search is a blind search or brute force search which includes search techniques such as Breadth First Search, Depth First Search, Uniform Cost Search, Iterative Deepening, and Bidirectional Search. Uninformed search does not contain any information about the number of steps to reach from current state to goal state. The uniformed search will consider the path which is most promising at that moment. It will not consider the optimum path to reach the goal.

B. **Informed search**

It uses heuristic function (To estimate how close a given state is from goal state) to solve a problem. Informed search uses generate and test approach. It includes hill climbing, steepest hill climbing, A*, AO*, and Best First Search algorithms. Informed search is more efficient than uninformed search. In knowledgeable search heuristic function is used as a representation that will lead us to the goal state. The informed search will not traverse search tree blindly. It will reflect on the next node to pass through based on some estimate task (Heuristic Function) to reach the goal state from current state.

III. WORKING OF SEARCH TECHNIQUES (UNINFORMED AND INFORMED)

A.BFS (Breadth first search)

In BFS each and every one nodes are extended level by level. Initial increase every nodes at initial stage in the explore tree, then expands all the nodes of the second level and this way it reaches the goal. In Breadth First Search the frontier is actualized as a queue which works as First in First out (FIFO). It is a poor strategy when all solution has a long way length or then again there is some heuristic information accessible [2]. It is not operating as soon as memory

constraint is elevated. O (b^d) is time complexity as well as O (b^d) space complexity, where b is branching factor in addition to d is solution depth.

For example, consider a graph (Figure 2). Table 1 explain open list and closed list for BFS. The open list is locate of nodes however to search and stopped list is set of nodes already been explored. At stage 0 node 1 is extended primary node of children of 1-2, 3, and 4 are additional to the stand in line. After with the purpose of according to queue move toward node 2 are expanded and child of node 2-6 is added to the queue. This way search reaches the goal state. The Applications are:

- Intended for unweighted diagram smallest amount spanning tree in addition to direct path.
- Mutual network.
- Communal network Websites.
- Global Positioning System Navigation systems.
- Designed for trying whether the graph is bipartite or not.

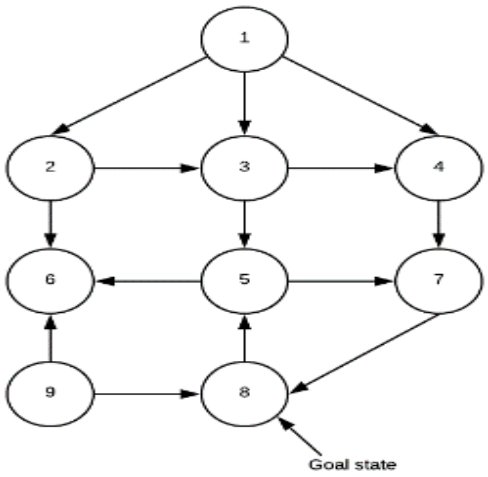

Fig.2 Graph for BFS and DFS

TABLE I. OPEN AND CLOSED LIST FOR BFS

OPEN LIST	CLOSED LIST
1	1
2,3,4	2
3,4,6	3
4,6,5	4
6,5,7	6
5,7	5
7	7
8-GOAL STATE	-

B.DFS (Depth First Search)

In development initiate as of the basis node along through situate elsewhere up and about in the route of the cordial unexpanded node in a search of goal node. In DFS the boundary is actualized as a Last in first out which mechanism as Stack. In addition known as recursive algorithm for the reason that it is put into service by means of the Last in First Out. $O(b^m)$ Time complexity is in addition to space complexity is $O(b^m)$, where b is branching factor and m is maximum depth. For example, consider a graph (Figure 2). Table 2 demonstrate unlock list along with stopped up list in support of DFS. By the side of level 0 node 1 is lengthened initial. Offspring of node 1-4, 3, 2 be supplementary in the direction of the LIFO. Afterward according to stack move toward node 2 is stretched along with child of node 2-6 is extra to the Last in First Out. Subsequently node 6 is lengthened. Since it do not contain whichever child it will backtrack to node 3 and node 3 will be expanded. These technique nodes inside the strength be lengthened and search reaches the goal state. The Applications are:

- Identify loops in a graph.
- Identifying mechanism which is powerfully linked from first to last a graph.
- Decision the answer of the problem in which only one resolution exists.

TABLE II. OPEN AND CLOSED LIST FOR DFS

OPEN LIST	CLOSED LIST
1	1
4,3,2	2
4,3,6	6
4,3	3
4,5	5
4,7	7
4,8- GOAL STATE	-

C. UCS (Uniform Cost Search)

It develops the node with inexpensive pathway. It is execute using the precedence first in first out. In the direction of compute cost of each node, consider this equation, c (m) = c(n) + c(n, m). where c(m) is the cost of the current node, c(n) is the cost of the previous node, and C (n, m) is the weight of the edge. The successor can be removed which are already in a queue with higher cost. O (b ⌊1+C*/e⌋) is T.C as well as S.C O (b ⌊1+C*/e⌋), where C is the optimal solution cost and each activity costs at least ε. The Applications are:

- Solving Maze problem.
- Path finding.

For example, consider graph (Figure 3). Open list and closed list for UCS are shown in Table

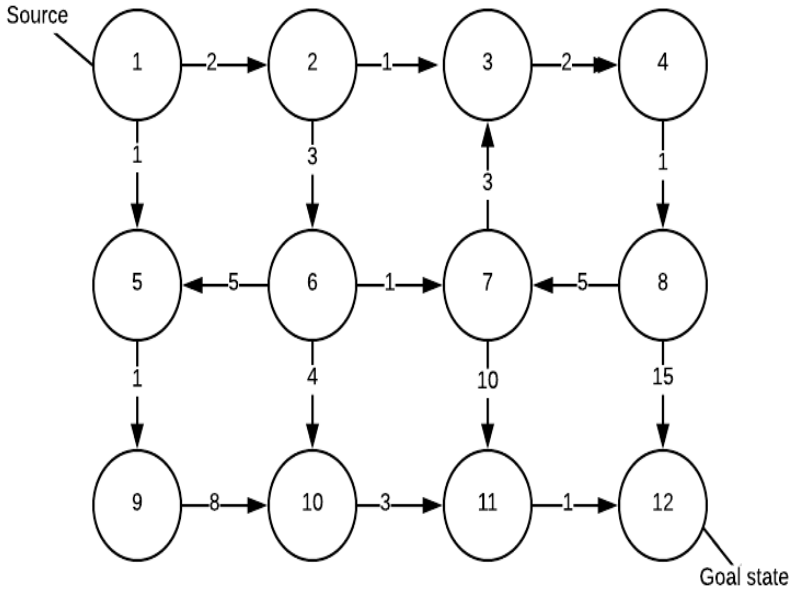

Fig.3 Graph for UCS

TABLE III. OPEN ALONG WITH CLOSED LIST FOR UCS

OPEN LIST	CLOSED LIST
$1^{(0)}$	$1^{(0)}$
$2^{(2)}, 5^{(1)}$	$5^{(1)}$
$2^{(2)}, 9^{(2)}$	$2^{(2)}$
$9^{(2)}, 6^{(5)}, 3^{(3)}$	$9^{(2)}$
$6^{(5)}, 3^{(3)}, 10^{(10)}$	$3^{(3)}$
$6^{(5)}, 10^{(10)}, 4^{(5)}$	$6^{(5)}$
$4^{(5)}, 7^{(6)}, 10^{(9)}$	$4^{(5)}$
$7^{(6)}, 10^{(9)}, 8^{(6)}$	$7^{(6)}$
$10^{(9)}, 8^{(6)}, 11^{(16)},$	$8^{(6)}$
$10^{(9)}, 11^{(16)}, 12^{(21)}$	$10^{(9)}$
$12^{(21)}, 11^{(12)}$	$11^{(12)}$
$12^{(13)}$- Goal State	-

D. A*

It underline of UCS furthermore pure heuristic search in the direction of effectively calculate best possible resolution [2]. For computing cost of every node, consider this equation, f(m) = c(m) + h(m), where c(m) = c(n) + c(n, m), h(m) is heuristic function. f(m) compute the most reduced aggregate cost. A large amount condensed assessment of f is selected at every node in support of development. Euclidean space second-hand while acceptable in the direction of travel in several path**,** Manhattan expanse second-hand while acceptable in the direction of travel in simply four directions-right, left, top, and bottom otherwise oblique distance second-hand when allowed to move in eight directions, same as the movement of king in chess) are second-hand as Heuristic function. If the assessment of f of two nodes be similar afterward the node has lowly h assessment be selected intended for development. The computation ends as soon as a goal is determined designed for development. Acceptable informed search formula ($h(n) \leq h^*(n)$) carry visit node support starting the stopped list in the direction of open list in the direction of acquire an best possible result. Time complexity is $O(b^d)$ and space complexity is $O(b^d)$, where b is branching factor and d is solution depth [4]. For

example, consider graph (Figure 3). Open and closed list associated with the example is shown in table III. The Applications are

- Traffic navigation system.
- Games.
- Finding shortest path.
- Real-time path re-planning of an unmanned surface vehicle avoiding underwater obstacles.

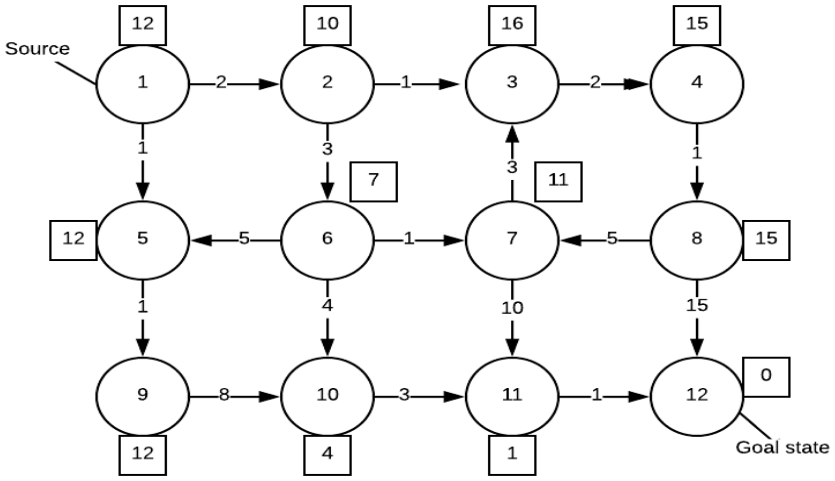

Fig.3 Graph for A*

TABLE IV. OPEN AND CLOSED LIST FOR A*

OPEN LIST	CLOSED LIST
$1^{(12)}$	$1^{(12)}$
$2^{(12)}, 5^{(13)}$	$2^{(12)}$
$5^{(13)}, 3^{(19)}, 6^{(12)}$	$6^{(12)}$
$5^{(13)}, 3^{(19)}$	$5^{(13)}$
$3^{(19)}, 7^{(17)}, 10^{(13)}, 9^{(14)}, 10^{(13)}$	$10^{(13)}$
$3^{(19)}, 7^{(17)}, 10^{(13)}, 9^{(14)}, 10^{(13)}, 11^{(13)}$	$11^{(13}$
$3^{(19)}, 7^{(17)}, 9^{(14)}, 12^{(13)}$ - GOAL STATE	-

D. Best First Search (Greedy search)

It is a unification of BFS along with DFS. BFS be executing with the precedence First in first Out. The advantage of Depth First Search is that it gives a solution without calculating all nodes. At the same time as BFS appear by the side of an explanation exclusive of exploration certain with the purpose of the process perform not obtain fixed. BFS is a combination of

these two, replace linking paths. Next to each step the nodes among the created ones, the best suitable node is selected in support of assist development, capacity be there this node contain a position to a comparable intensity or different, hence can flip between Depth First and Breadth First Search [5]. It is also known as greedy search. Time complexity is O (b^d) and space complexity is O(b^d), where b is branching factor and d is solution depth [4].

IV. COMPARISON OF VARIOUS SEARCH ALGORITHMS

The performance of the algorithm is evaluated based on four parameters as follows:

1. Time complexity: Time taken by the algorithm to find a solution.
2. Space complexity: Memory required by the algorithm to perform the search.
3. Optimality: result present through the algorithm resolve forever exist best possible otherwise not.
4. Completeness: Is that calculation ensured to discover an answer.

TABLE V. COMPARISON OF SEARCH ALGORITHMS

Algorithms	Time Complexity	Space Complexity	Optimality	Completeness
BFS	O (b^d)	O (b^d)	YES	YES
DFS	O (b^d)	O (d)	NO	NO
UCS	O(b ⌊1+C*/e⌋)	O(b ⌊1+C*/e⌋)	YES	YES
A*	O (b^d)	O (b^d)	YES	YES
BFS (Best First Ssearch)	O (b^d)	O (b^d)	YES	NO

V. CONCLUSION

I conclude that heuristic search is more capable and acceptable than blind search. I have explained from the example open and closed list for UCS and A*, it is clear that in UCS has more nodes are required to expand to reach the goal state at the same time as in A* only few nodes are required to reach the goal state. Also, UCS is more time consuming than A*. So A* is auxiliary competent and optimal than UCS.

REFERENCES

[1] J.Sturaj Russel and Peter Norvig , Artificial Intelligence A Modern Approach , Third Edition , Prentice Hall , Engle wood Cliffs , New Jersey 07632.

[2] Deepika Garg , " Comparative study of various searching Algorithms" Proceeding of National Conference on Innovative Trends in Computer Science Engineering (ITCSE-2015) held at BRCMCET, Bahal on 4th April 2015.

[3] Shabina Banu mansuri, Shiv kumar "Comparative Analysis of Path Finding Algorithms ", IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN:2278-0661,P-ISSN:2278-8727, Vlume 20, Issue 5, Ver.I(Sep-Oct 2018), PP.38-45.

[4] Chandel, and Manu sood, Searching and Optimization Techniques in Artificial Intelligence: A Comparative study and Complexity Analysis, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 3, Issue 3, March 2014.

[5] Mr.Girish P Potdar, and Dr. R.C. Thool, Comparision of Various Heuristic Search Techniques for Finding Shortest Path, International Journal of Artificial Intelligence and Application (IJAIA). Vol.5, No.4, July 2014.

