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: 6.199

IJCSMC, Vol. 8, Issue. 6, June 2019, pg.57 – 65

Comparative Analysis of Various Uninformed Searching Algorithms in AI

Gayathri R
Master of Computer Applications, Sree Saraswathi Thyagaraja College, Pollachi, Tamil Nadu, India
gayathrimca1820@gmail.com

Abstract— In Artificial Intelligence the search algorithm plays a vital role to figure out the problems of the shortest path finding. These search algorithms can be classified into two categories. That is, uninformed search algorithm and informed search algorithm. In this article, I am only paying attention on various uninformed search algorithms such as, Depth First Search (DFS), Breadth First Search (BFS), Iterative Deepening Search (IDS), Uniform Cost Search (UCS) and Depth Limit Search (DLS). This paper also includes how these algorithms do work in real time applications. I made various comparisons of these searching algorithms based on time complexity, space complexity, optimality and completeness. Out of all the above said algorithms the comparison result shows that Uniform Cost Search provides the optimal solution.

Keywords— Artificial Intelligence, shortest path finding, uninformed search algorithm, working strategy, Uniform Cost Search

I. INTRODUCTION

Artificial Intelligence (AI) is a branch of computer science that aims to make a computer to think like a human being [1], [4]. Nowadays, it is one of the most popular growing technologies in this tech-world. It solves many of the real world problems more efficiently. Route founding is one among the biggest problems in Artificial Intelligence. By using various searching algorithms and techniques we can solve this problem [1].

There are two types of AI techniques which can be used to solve these kinds of problems. 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 [5]. The informed search algorithm is **heuristic search** which uses heuristic function (to estimate how close a given state is from goal state) to solve a problem [5]. The paper focuses more on uninformed search algorithm such as, Depth First Search (DFS), Breadth First Search (BFS), Iterative Deepening Search (IDS), Uniform Cost Search (UCS) and Depth Limit Search (DLS).

II. VARIOUS TYPES OF UNINFORMED SEARCH ALGORITHMS

A. *Breadth First Search (BFS)*

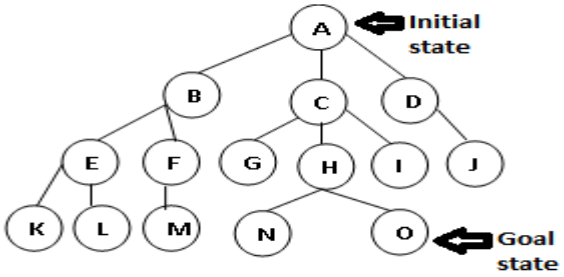
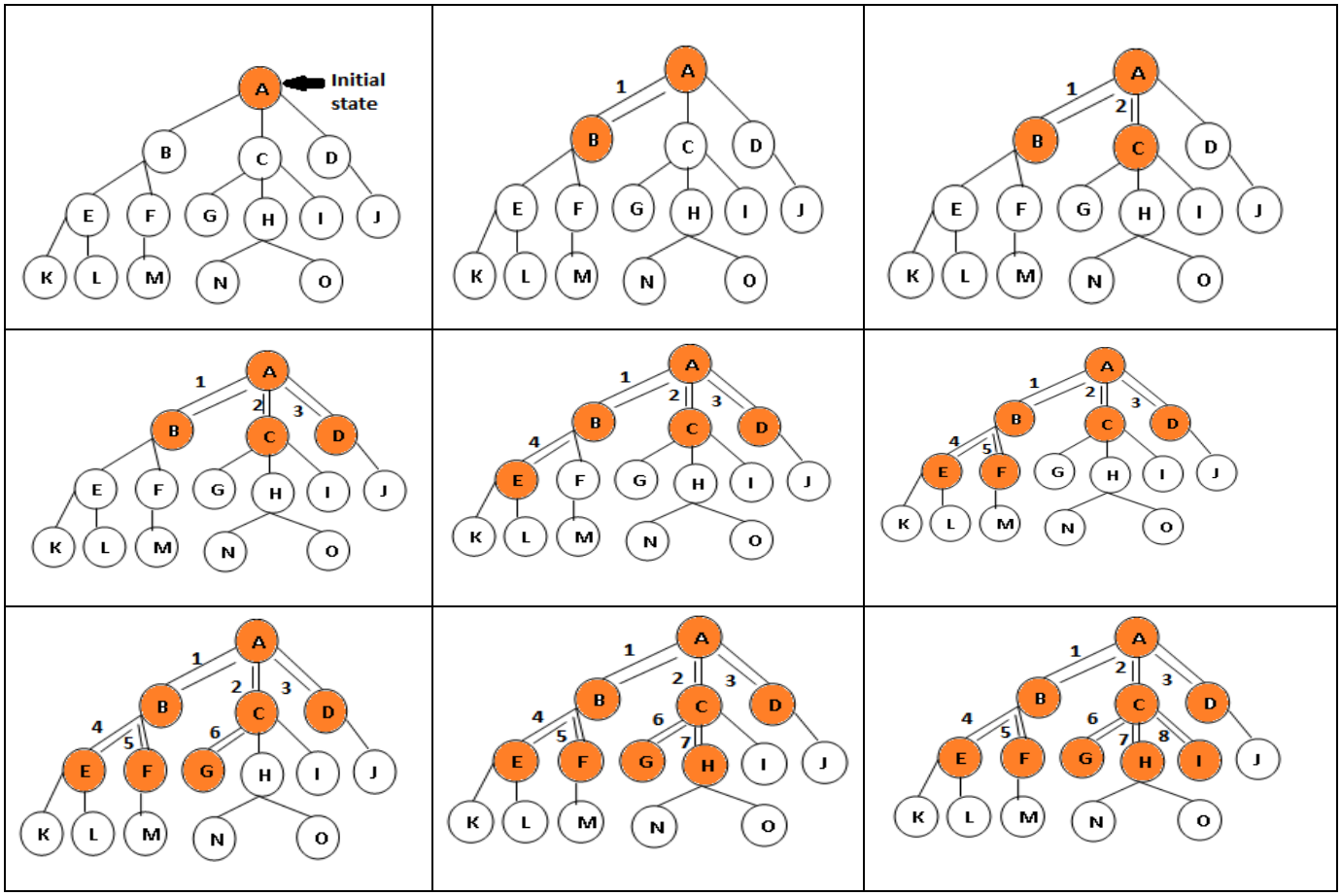


Fig.1 Graph for BFS

In Breadth First Search (BFS) all the vertices are explored or traversed level by level. That is, it first expands all the nodes at first level in the search tree, then expands all the nodes of the second level and this way it reaches the goal. This algorithm can be implemented by using **queue** data structure. That is, it works based on the **First In First Out (FIFO)** principle. The Time complexity is $\mathbf{O(b^{d+1})}$ and space complexity is $\mathbf{O(b^{d+1})}$, where **'b'** is branching factor and **'d'** is the solution depth [2].



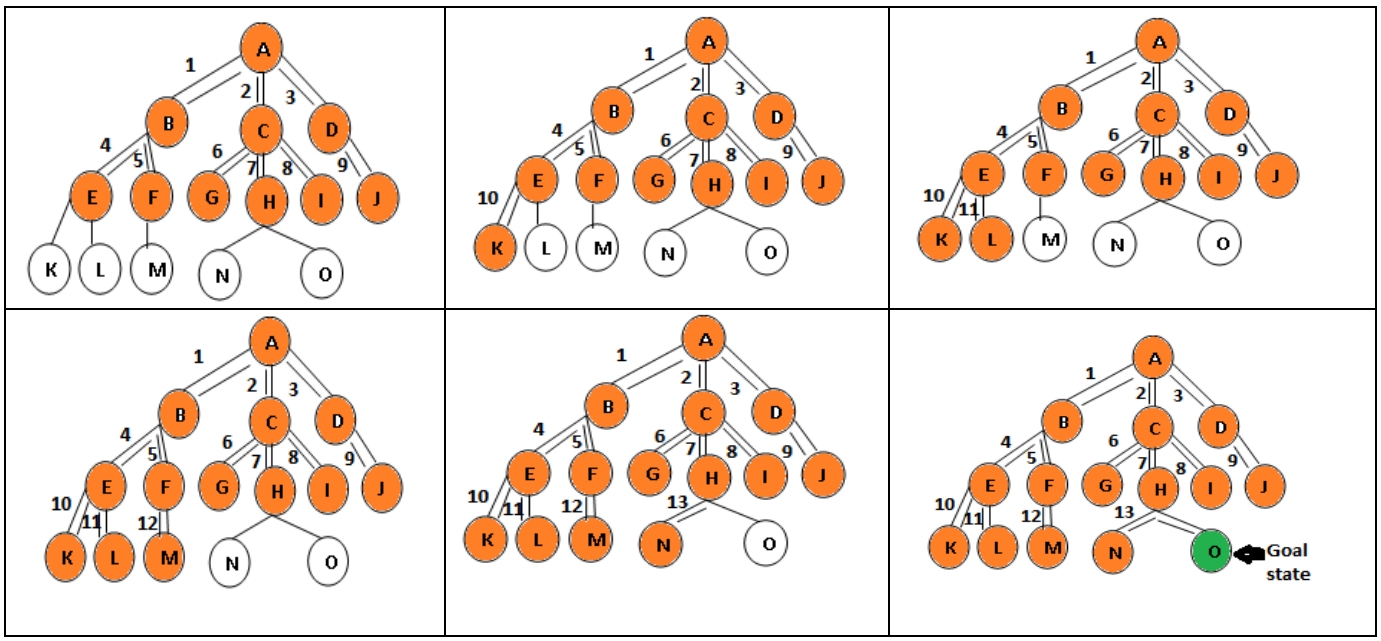


Fig. 2 Representation of Tree Traversal using BFS

TABLE I. OPEN AND CLOSED LIST FOR BFS

Open list (Unexplored nodes)	Close list (Visited nodes)
A	A
B,C,D	B
C,D,E,F	C
D,E,F,G,H,I	D
E,F,G,H,I,J	E
F,G,H,I,J,K,L	F
G,H,I,J,K,L,M	G
H,I,J,K,L,M	H
I,J,K,L,M,N,O	I
J,K,L,M,N,O	J
K,L,M,N,O	K
L,M,N,O	L
M,N,O	M
N,O	N
O ← Goal state	-

The above example demonstrates how to find goal state with optimal path by using BFS algorithm. The search starts from the initial node 'A' and it visits all the nodes in first level such as B, C, D and it adds visited node on the queue, then it explores the second level vertices and so on. This way search reaches the goal state. Table-I shows the open and close list for BFS algorithm. The open list is set of nodes yet to explore and closed list is set of nodes already been explored.

B. Depth First Search (DFS)

In Depth First Search (DFS) expansion starts from the initial node in the graph and it explores or traverses deepest unexplored node of that vertex, in this way it reaches the goal node. It is also called **edge based method** and it works in the recursive fashion where the vertices are explored along a path. This algorithm can be implemented by using **stack** data structure. That is, it works based on the **Last In First Out (LIFO)** principle. Depth First Search traverses each vertex exactly once and edge is inspected twice. The Time complexity is **$O(b^m)$** and space complexity is **O(bm)**, where **'b'** is branching factor and **'m'** is maximum depth [2].

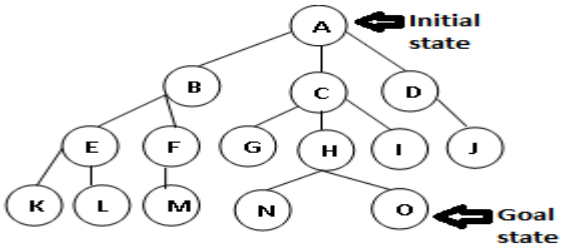


Fig. 3 Graph for DFS

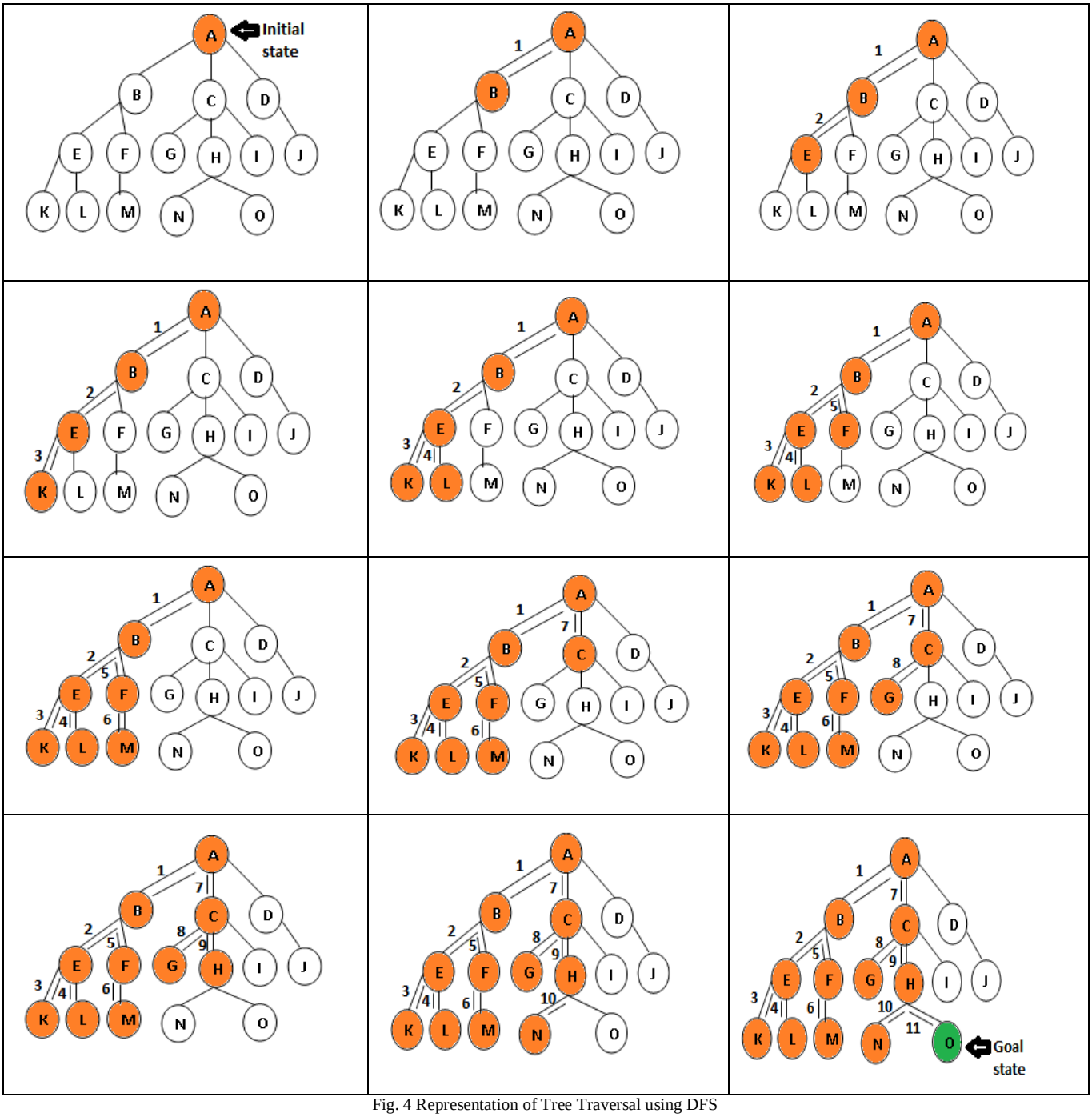


Fig. 4 Representation of Tree Traversal using DFS

TABLE II. OPEN AND CLOSED LIST FOR DFS

Open list (Unexplored nodes)	Close list (Visited nodes)
A	A
B,C,D	B
E,F,C,D	E
K,L,F,C,D	K
L,F,C,D	L
F,C,D	F
M,C,D	M
C,D	C
G,H,I,D	G
H,I,D	H
N,O,I,D	N
O,I,D  Goal state	-

The above example demonstrates how to find a goal state with optimal path by using DFS algorithm. The search starts from the initial node 'A' and it visits deepest path of that vertex such as, B, E, K,L and it adds visited node on the stack, then it backtracks to previous level and examines the next nearest vertices in the graph and so on. This way the DFS search reaches the goal node. Table-II shows the open and close list for DFS algorithm.

C. Iterative Deepening Search (IDS)

The Iterative Deepening Search (IDS) is one of the state space search strategy in which the nodes are expanded on depth by depth. Each depth bound is considered as iteration. That is, a depth-limited version of depth-first search is run repeatedly with increasing depth limits until the goal is found. It is also known as Iterative Deepening Depth First Search (IDDFS). The **Time complexity** is **O(b^d)** and **Space complexity** is **O(bd)**, where **'b'** is the branching factor and **'d'** is the depth of the shallowest solution. The IDS also works based on the **Last In First Out (LIFO)** principle, that is, stack data structure [2].

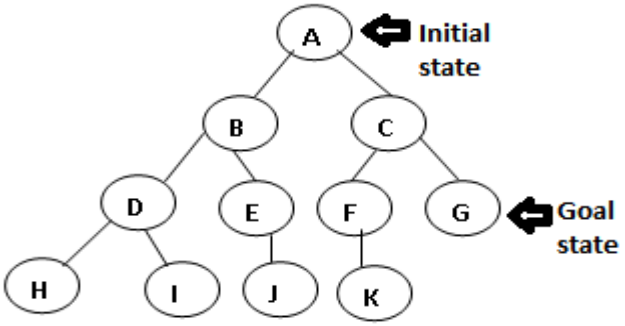
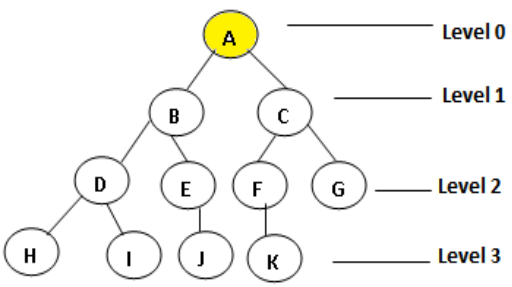
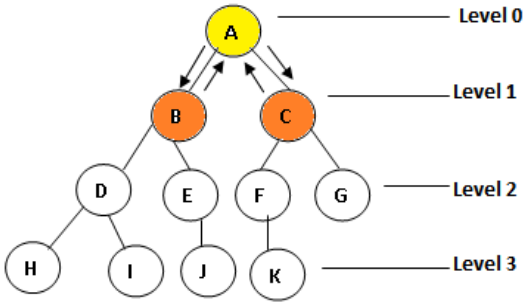


Fig. 5 Graph using IDS



IDS: Depth bound 0 (Level – 0)



IDS: Depth bound 1 (Level – 1)

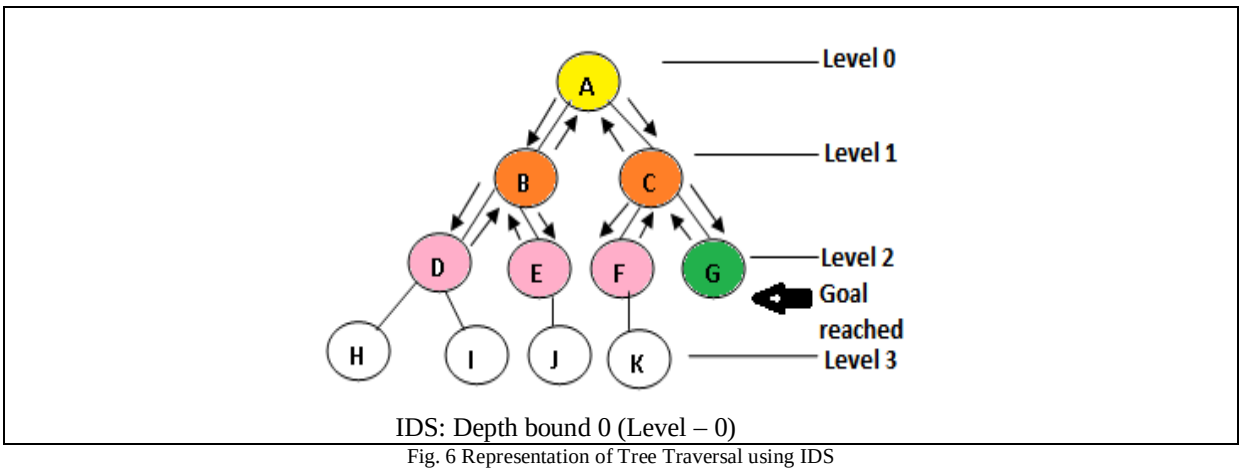


Fig. 6 Representation of Tree Traversal using IDS

TABLE.III RESULT TABLE FOR IDS:

Depth (Level)	Iterative Deepening Search
Level – 0	A
Level – 1	A,B,C
Level – 2	A,B,C,D,E,F,G

The above example shows how do the IDS algorithm works to find the goal node. Each iteration depends upon the depth (level) of the tree. It starts with Level – 0 and continues its iteration until the goal node is reached.

D. Uniform Cost Search (UCS)

The Uniform Cost Search (UCS) is a state space search algorithm in which it founds the goal state based on the cost of the node. That is, the nodes are expanded with minimum cost path. To calculate cost of every 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. The time complexity is **O(b ⌊1+C*/e⌋)** and the space complexity is **O(b ⌊1+C*/e⌋)**, where **C** is the optimal solution cost and each activity costs at least **e** [3].

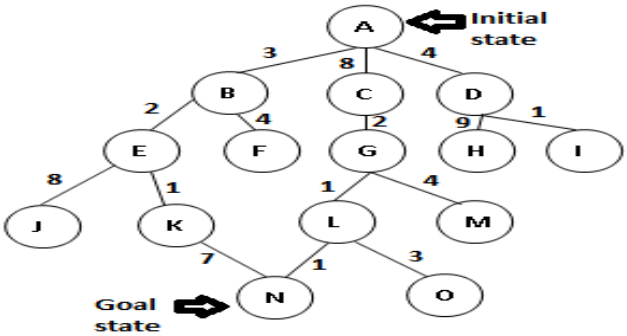
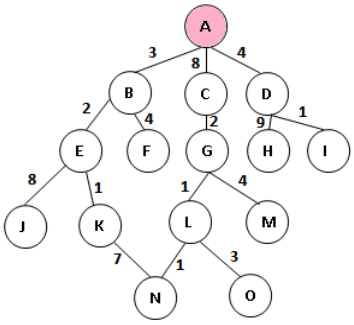
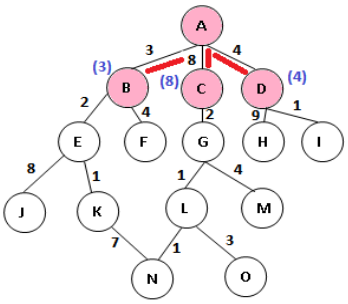


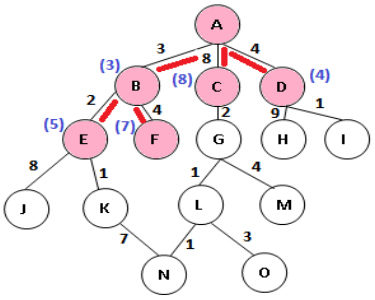
Fig. 7 Graph using UCS



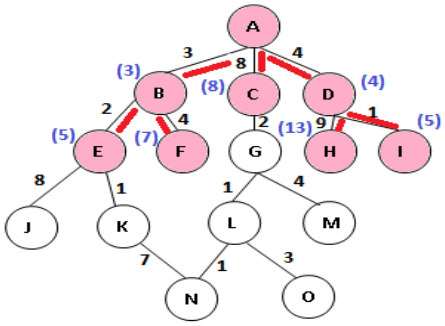
Initial State: **A**



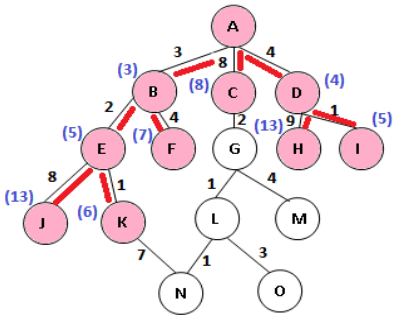
Iteration-1:
{A→B,3},{A→C,8},{A→D,4}



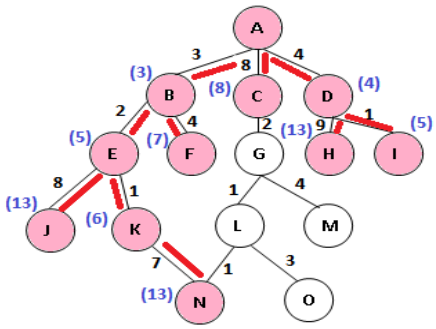
Iteration-2:
{A→B→E,5},{A→B→F,7}



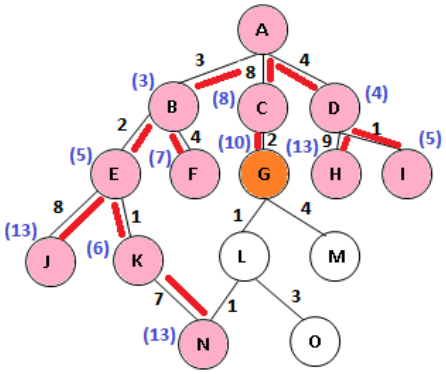
Iteration-3:
{A→D→H,13},{A→D→I,5}



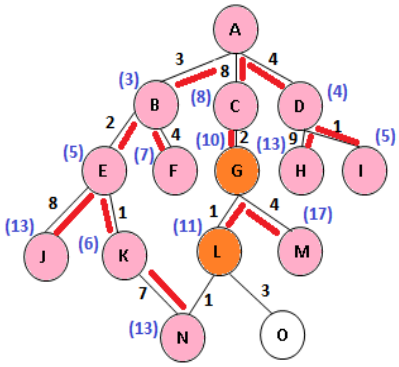
Iteration-4:
{A→B→E→J,13},
{A→B→E→K,6}



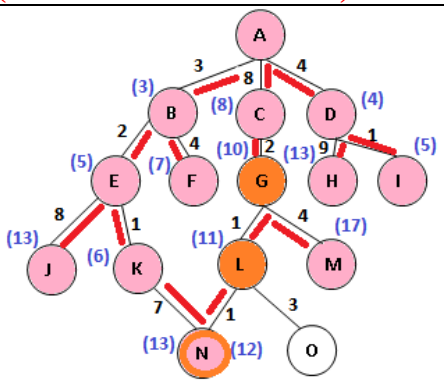
Iteration-5:
{A→B→E→K→N,13}
(Goal state reached with cost 13)



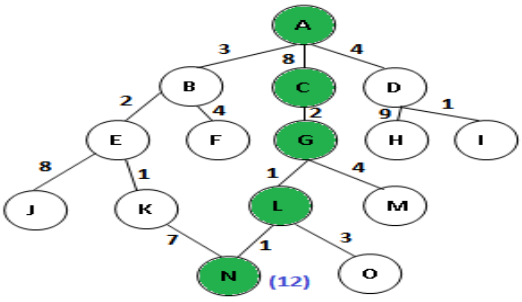
Iteration-6:
{A→C→G,10}



Iteration-7:
{A→C→G→L,11},
{A→C→G→M,17}



Iteration-8:
{A→C→G→L→N,12}
(Goal state reached with cost 12)



Minimum cost from A to N is **{A→C→G→L→N, 12}** .

Fig. 8 Representation of Tree Traversal using UCS

TABLE. IV ITERATIONS FOR UCS

Iteration	UCS Traversal (Path with cost)
Iteration – 1	{A→B,3},{A→C,8},{A→D,4}
Iteration – 2	{A→B→E,5},{A→B→F,7}
Iteration – 3	{A→D→H,13},{A→D→I,5}
Iteration – 4	{A→B→E→J,13},{A→B→E→K,6}
Iteration – 5	{A→B→E→K→N,13}
Iteration – 6	{A→C→G,10}
Iteration – 7	{A→C→G→L,11},{A→C→G→M,17}
Iteration – 8	{A→C→G→L→N,12}

The above example shows how does the Uniform Cost Search algorithm works to find the goal state with minimum cost. The UCS traversal starts from the initial state **A** and it examines the first level nodes **B, C** and **D**. Then it compares the cost of each path then it chooses the minimum cost path and continues the traversal. In our example, in first level search we get three different paths that is, {A→B, 3}, {A→C, 8}, {A→D, 4}. Out of these the minimum cost path is A→B (cost is 3). That is why this algorithm chooses that path to continue the traversal. In every iteration the USC algorithm follows this technique to continue the traversal. In this way the algorithm finds the goal node with minimum cost.

E. Depth Limit Search (DLS)

The Depth Limit Search (DLS) algorithm is a variation of Depth First Search (DFS) algorithm, which works based on the pre-specified limit. That is, if we put a limit ***l*** on how deep a depth first search can go, we can guarantee that the search will terminate (either in success or failure). If there is at least one goal state at a depth less than ***l***, this algorithm is guaranteed to find a goal state, but it is not guaranteed to find an optimal path. The time complexity is $\mathbf{O(b^l)}$ and the space complexity is **O(b*l*)**. For most problems we will not know what is a good limit ***l*** is until we have solved the problem [2].

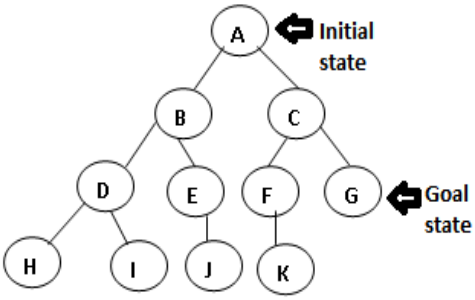


Fig. 9 Graph for DLS

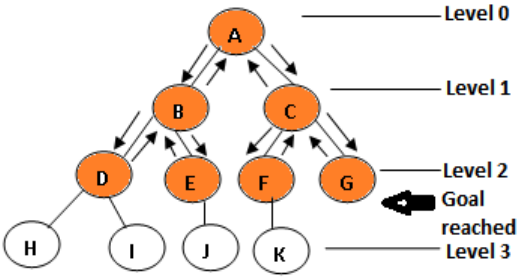


Fig. 10 Depth Limit Search (DLS) with ***l*** = 2

TABLE – V. OPEN AND CLOSE LIST FOR DLS

Open list (Unexplored nodes)	Close list (Visited nodes)
Depth bound (***l*** = 2)	
Open=[A]	Close=[ ]
Open=[B,C]	Close=[A]
Open=[D,E,C]	Close=[A,B]
Open=[E,C]	Close=[A,B,D]
Open=[C]	Close=[A,B,D,E]
Open=[F,G]	Close=[A,B,D,E,C]
Open=[G]	Close=[A,B,D,E,C,F]
Open=[ ]	Close=[A,B,D,E,C,F,G]

The above example shows the working of Depth Limit Search (DLS) algorithm. It works similar to Iterative Deepening Search (IDS) but the only difference is, it sets the **limit (*l*)** for traversal then it goes only in that particular limit (it may be reach the goal or fail to meet it), whatever it stops the traversal if the limit is reached means. In the graph, **A** is the initial state and the specified limit ***l*** is **2**. That means, the traversal starts from the initial vertex A and ends with the specified level 2. Open and closed list associated with the example is shown in Table - V.

III. COMPARISONS OF VARIOUS UNINFORMED SEARCH ALGORITHMS

The following parameters are used to evaluate the performance of the searching algorithms.

A. Time Complexity

The time complexity of an algorithm is an expression for the **worst – case** amount of time it will take to run [3].

B. Space Complexity

The space complexity of an algorithm is an expression for the **worst – case** amount of memory that the algorithm will use (number of nodes) [3].

C. Optimality

A search algorithm is optimal if, when it finds a solution it is the best solution [3].

D. Completeness

A search algorithm is complete if, whenever at least one solution exists, the algorithm is guaranteed to find a solution within a finite amount of time [3].

Evaluation of search strategy, **b** is the branching factor; **d** is the depth of the shallowest solution; **m** is the maximum depth of the search tree; ***l*** is the depth limit [3].

TABLE VI. COMPARISON OF VARIOUS UNINFORMED SEARCH

Criteria	BFS	DFS	IDS	UCS	DLS
Time Complexity	$O(b^{d+1})$	$O(b^{m})$	$O(b^{d})$	$O(b^{\lfloor 1+C^*/\epsilon \rfloor})$	$O(b^{l})$
Space Complexity	$O(b^{d+1})$	O(bm)	O(bd)	$O(b^{\lfloor 1+C^*/\epsilon \rfloor})$	O(b*l*)
Optimality	Yes	No	Yes	Yes	No
Completeness	Yes	No	Yes	Yes	No

IV. CONCLUSION

In this article, I have explained various search algorithms with examples among that Uniform Cost Search (UCS) is more efficient. It takes minimum time to determine the goal state with optimal path. Instead of UCS, Breadth First Search (BFS) and Iterative Deepening Search (IDS) are optimal because it always expands the shallowest unexpanded node. Out of these uninformed search algorithms Depth Limit Search (DLS) is the worst case algorithm. Because in this strategy, we don't have any guarantee to reach the goal state. Similarly the Depth First Search (DFS) is also worst, because it takes more steps to reach the goal and the memory space it has taken is depends upon the depth of the algorithm. Hence the Uniform Cost Search (UCS) finds an optimal solution than other types of uninformed search algorithms.

REFERENCES

[1] Stuart J. Russell and Peter Norvig, *Artificial Intelligence A Modern Approach*, Third Edition, Prentice Hall, Englewood Cliffs, New Jersey 07632.

[2] Deepak Khemani, *A First Course in Artificial Intelligence*, McGraw Hill Education (India), 2013.

[3] Maharshi J. Pathak, Ronit L. Patel, Sonal P. Rami, "Comparative Analysis of Search Algorithms", *International Journal of Computer Applications* (0975 – 8887) Volume 179 – No.50, June 2018.

[4] Deepika Garg, "Comparative Study Of Various Searching Algorithms", Proceedings of National Conference on *Innovative Trends in Computer Science Engineering (ITCSE-2015)* held at BRCMCET, Bahal on 4th April 2015.

[5] 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, Volume 20, Issue 5, Ver. I (Sep - Oct 2018), PP 38-45.

