WWJMRD 2018; 4(12): 96-99
www.wwjmrd.com
International Journal
Peer Reviewed Journal
Refereed Journal
Indexed Journal
Impact Factor MJIF: 4.25
E-ISSN: 2454-6615

Vinay Menon
School of Computer Science and Engineering,
Vellore Institute of Technology, Vellore, India

Geraldine Bessie Amali.D
School of Computer Science and Engineering,
Vellore Institute of Technology, Vellore, India

Gopichand G
School of Computer Science and Engineering,
Vellore Institute of Technology, Vellore, India

Santhi H
School of Computer Science and Engineering,
Vellore Institute of Technology, Vellore, India

Gayatri P
School of Computer Science and Engineering,
Vellore Institute of Technology, Vellore, India

Correspondence:
Geraldine Bessie Amali.D
School of Computer Science and Engineering,
Vellore Institute of Technology, Vellore, India

Performance Analysis of Various Uninformed and Informed Search Strategies on 8 Puzzle Problems - A Case Study

Vinay Menon, Geraldine Bessie Amali.D*, Gopichand G, Santhi H, Gayatri P

Abstract
The 8-puzzle is basically a type of puzzle game in which the area is divided into a uniform grid of 9 spaces which is made up by a square grid of size 3*3. Another version of this puzzle game is that of the 15-puzzle which instead has 16 spaces in a 4*4 grid. The contents of the 8-puzzle are arranged in such a manner as there is one empty grid where the contents can be moved into for a re-arrangement of the puzzle in order to get the optimal state. Usually the optimal state is that of rearranged contents in such a manner as the numbers constituting the grades are arranged in ascending order. The different algorithms for performing the 8-puzzle problem is performed for finding the one that is most optimal and also a comparison of performance is being done where different parameters are taken into concern.

Keywords: Breadth first search, Depth first search, A* search, Hill Climbing Search

Introduction

The main objective of the 8-puzzle is to reach a given goal state from an initialized state with a number of steps leading to the goal state. As a result, it is observed that there exists not only a single way but multiple routes to reach this desired goal state. All the possible routes to the given goal from the initial state will not be of uniform length. Some of these routes will be extremely short in comparison to other routes which will take a large number of steps to reach the optimum state. There may also occur instances where the there is no way to reach the optimum goal with a given algorithm but there may exist another algorithm that can converge to the same goal state from the same initial state. Typical algorithms that are used for such 8-puzzle problems is that of searching algorithms where the problem is solved step by step by searching for a solution. This solution is basically a sequence of steps that results from the first state initialized to the final optimum state. The biggest problem involved in the 8-puzzle problem is to find the shortest solution,that is the solution containing the least number of steps from initial state to final state. The basic idea behind the 8-puzzle problem is that the space that contains no number is legally exchanged with any tile that is adjacent to it horizontally or vertically. For a given 8-puzzle problem initialized with a given state and a given final state, the search space is represented by all the possible states that exist while traversing from the initial state to the final state. There exists 3,62,880 different solution sets of the 8-puzzle.The four directions of movement are up, down, left and right. In the traditional approach, the cost of moving from one state to another by moving the tile is equal to one. Therefore, for performing the 8-puzzle problem we will see the working of different search algorithms such as Breadth First Search, Depth First Search, A*(A star), Hill Climbing and Steepest Ascent Hill Climbing. Different metrics for distance will also be assumed for the calculation of the heuristic value for some of the above mentioned algorithms. Various other metrics will also be looked into [11].

Literature Review

The study puts focus on the analysis of social networks to understand the reason for maintaining relationships on such social networks, determining users that are similar in their hobbies and interest, as well as identifying content that is currently popular in some particular community as well as hot topics that are circulating around within a community. The proposed social search uses aspects of breadth first search. Since the entire social network graph cannot be studies, small sample of the graph is used for analysis through a method called graph sampling [1]. The paper [2] proposes an implementation of DFS on lazy functional language in order to construct algorithms from the individual sections. This implementation of algorithm formulation is agreeable with a given proof. The study [3] involves a group of search algorithms that are known as the star algorithms, each of which are of different types and have considerable variations. This study [4] focuses on the implementation of Genetic algorithm in solving the N-puzzle problem. Previous methods in solving the N-puzzle problem have also been recorded in terms of performance. The reason why Genetic algorithm is very well explained and justified. The results of comparison prove that the Genetic algorithm presents with better complexity than the other observed methods in performing N-puzzle. This research focuses on this implementation and stresses on the extension of this in solving similar NP Hard problems using Genetic algorithms. [10] [11].

Search Algorithms

A search algorithm is basically used for search problems where it is necessary to attain a given optimum state using the search space in a given problem domain. Traditionally, the idea of applying search algorithms is always related to data structures such as arrays, search trees, linked and several other data structures where the goal is determined by actions such as finding the minimum value present in the data space provided by such data structures. For any given problem domain, these algorithms will have to be modified to suit the needs that include satisfying a given optimum condition which could be either finding a given value or attaining a particular state. The different algorithms that are used for solving the 8-puzzle problem will be explained. The output of the Breadth First Search is that of the sequence of actions that are generated as the traversal from the root state to the final optimum state.

Breadth First Search

Breadth First Search is a type of search algorithm that is based on the Breadth First Traversal seen in trees and graphs. In this strategy, the initial state of the problem is taken as the root and the search space is represented as nodes where the immediate neighbors of a given node result from a single move or operation to the node. In Breadth First Search, the searching occurs in such a fashion where the immediate neighbors are explored first and once all the given immediate neighbors that exist within a single level is completely visited and no optimum state is found, it moves onto the next level of neighbors with the goal of finding the optimum state. Therefore, the different levels of neighbors are visited linearly as the number of levels of neighbors. The Breadth First Search algorithm uses the queue data structure to perform the search which works on a "First In First Out" principle. Depending on the problem domain and the given goal state, the time complexity can be estimated linearly with the number of vertices and the number of edges. The number of edges can vary from 1 to the square of the number of vertices. Therefore, in practical terms, as the size of the search space increases so does the time complexity involved in finding the optimum state.[5]

Depth First Search

Depth First Search is a type of searching algorithm that is based on the Depth First Traversal seen in trees and graphs. In this strategy, the initial state of the problem is taken as the root and the search space is represented as nodes where the immediate neighbors of a given node result from a single move or operation to the node. In Depth First Search, the searching starts from the root node and it goes as far in each branch and when the last node in the branch is visited; it backtracks all the way up and moves into the next branch. A more detailed explanation of this is, a given node goes to its first immediate neighbor, from there it goes to the first immediate neighbor of the last visited neighbor and goes on until there are no more immediate neighbors and then it back tracks all the way back to the root node where it then moves onto the next immediate neighbor, and this goes on until the optimal state or goal state is attained. Similar to Breadth First Search, the time complexity varies linearly with the number of edges and number of vertices. This algorithm performs good when the goal state or the optimal state is far along a branch without the need to go through all the levels in order to reach this state as in Breadth First Search. The Depth First Search algorithm uses a stack data structure which follows a "Last In Last Out" principle. [6][7]

A* Algorithm

A^* (A star) algorithm is a very widely used algorithm for search problems and it is so because of its enhanced performance when it comes to searching and also the accuracy in getting the given solution out of the search space. Generally, this algorithm is considered very greater than other approaches to searching. The algorithms that are known to outperform the A star algorithm generally are overthrown by other algorithms that can preprocess the search space which gives them an added advantage that they know exactly which route to take so that they reach the final state. This algorithm is an informed search algorithm that finds the goal state by searching through all the possible routes to the solution so that the route with the least cost is found. From all these routes, the algorithm considers that route which seems to bring upon the goal state in the fastest time. It applies a weighted graph methodology where a particular edge from one node to another is given a weight value which will decide which route is to be taken from a given node where the least weight is chosen upon to minimize the cost incurred. A star algorithm must proceed in such a manner as to minimize a cumulative function of the cost already incurred in the path as well as the heuristic which is used to estimate the cost of the least expensive path from the current node to the goal state. [8]

$$f(n) = c(n) + h(n) \quad (1)$$

Where f (n) is calculated as the sum of the cost already incurred c(n) and the heuristic h(n) that gives the cost of the least expensive route to the optimum goal state.

Hill Climbing Algorithm

Hill Climbing search algorithm is another search algorithm that focuses on the use of heuristics. In Artificial Intelligence, it is used for optimization problems. Hill Climbing search is well known to provide a sufficiently good route to the final goal state. This algorithm is used to bring the solution by maximizing or minimizing a function from the inputs that are given. Even though it may not bring about the best possible or optimal solution, this algorithm is well known to provide a very good solution with a very good time complexity. The basis of this algorithm is that it goes through all the possible solutions present in the search space to find the optimal solution. Once it finds a reasonably good solution, the algorithm will terminate and return the given solution as the best route to the goal state which may or may not be the actual optimal state in the given search space. This search algorithm is known to have a greedy approach in finding the relevant solution. According to the greedy approach, the search moves along those routes which minimizes the cost function to reach the optimal state. The greedy approach can in turn lead to a very long route towards the goal state because it only takes the cheaper route from the node it currently is in. This greedy approach can be evidently seen in the variant of Hill Climbing search known as Simple Hill Climbing Algorithm. [9]

Steepest Ascent Hill Climbing Algorithm

Steepest Ascent Hill Climbing differs from the traditional Hill Climbing algorithm in the sense that it chooses the successor nodes in such a way that it does not select the first node that it encounters; rather it chooses the best node among all the successors to the current node. Steepest Ascent Hill Climbing algorithm has features of the Breadth First Search algorithm as well. Under this algorithm comes the topic of foothills. It basically deals with local maxima where it is better than the neighbor nodes but not better than nodes that are comparatively far away from the current local maxima. There are situations in the current node where two routes have approximately similar values and hence it is not clear as to which route to take in order to reach the optimal state. Such states are known as plateau states. Sometimes, during the routing in the algorithm, we might end up in a state which does not lead to any good routes and basically cannot proceed further from the current node. In such situations, backtracking becomes important so that such a state can be escaped by choosing another route which is eventually better. Such a state is known as ridge state.

Results and Discussion

The Manhattan distance is the measure of distance between two points along the axes at right-angle. The name comes from the shortest path taken between two sources in the city of Manhattan which has a grid shaped streets.

In the case of 8-puzzle problem, we use the equation stated below where x_i and y_i are the x and y co-ordinates of the i^{th} tile in states.

$$h(s) = \sum_{i=1}^{8}(|x_i(s) - \bar{x}_i| + |y_i(s) - \bar{y}_i|) \qquad (2)$$

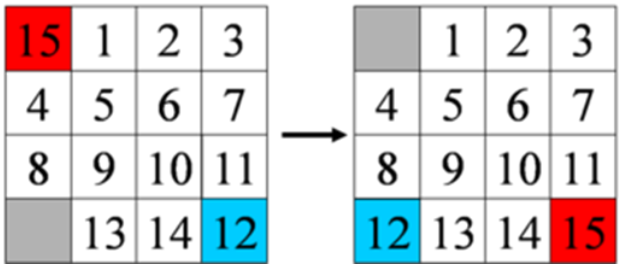

Fig.1: A sample move using Manhattan distance

The Hamming distance between two strings of equivalent length is the number of positions at which the relating symbols are not the same. Basically, it is the least number of substitutions or minimum number of errors that has to be made to convert on string to another.

The approach used here for problem relaxation by creating a heuristic which will not only reduce the cost to find the solution but also not in raise at the same time. In this case, we have combined the values for Hamming distance as well as Manhattan distance as heuristic value for a-star, steepest ascent hill climbing and hill climbing searches.

We make use of the concept of priority queue to order our moves. The priority is based on the heuristic value and the then adding that value to the depth. If ordered properly, the queue will allow the next best move for expansion. In this way, we obtain the right solution in minimum number of moves if the solution exists for the particular search algorithm.

We manually entered the start and goal states before coming to a conclusion. One such implementation is given below.

Start State:

2	5	8
0	1	3
4	7	6

Goal State:

1	2	3
4	5	6
7	8	0

We calculated the time taken, number of initialized nodes as well as the number of moves made by the search algorithm to reach the goal state regardless if solution is found or not.

Table 1: Comparison of the algorithms based on space complexity and number of moves

Algorithm	Nodes Explored	Number Of Moves
DFS	258938	102
BFS	7802	14
Hill Climbing	No solution found	8
Steepest Ascent HC	No solution found	23
A*	127	14

Table 1 indicates the various numbers of nodes initialized and the number of moves to reach the goal state for the above start and goal state

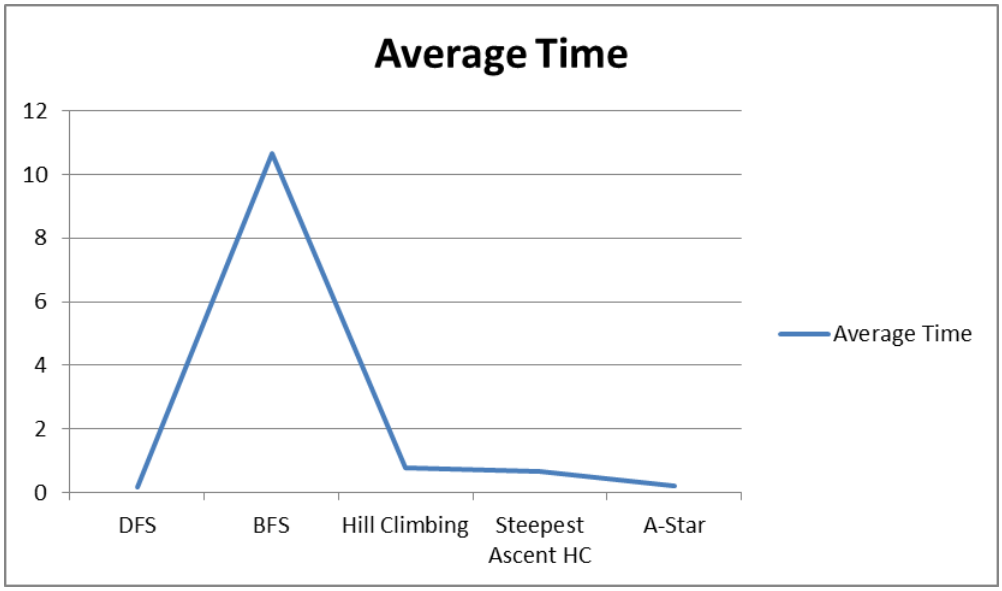

Fig.2: Comparison of the average time taken to reach the goal using the various search techniques

We performed the 8-puzzle problem for the above mentioned search algorithms. We used the multiple start and goal states apart from the one mentioned above. It is observed that DFS is not an optimal search algorithm for this problem while BFS can be for most of the cases as the cost per move is one. Hill climbing and its variant, steepest ascent hill climbing are not suitable for most of the cases even though it has the best solutions for some. A^* with the combined heuristic is shown to be best for almost all the cases and seems to be better compared to simply using one heuristic metric. The comparison of the performance of the various algorithms is presented in figure 2.

Conclusion

Even though all the search algorithms find an optimal solution, they find it in different amount of time. The more refined the heuristic is, the quicker we get the results. It is also clear that even though the methods are considered to be search algorithms, it is not always that we can find the solution for a particular problem. Another thing that is evident is that the reverse of the same problem whereby the goal state and start state is exchanged, we don't always get the same number of steps as solution and at times, there are no solutions for some of the methods that could find the solution for the reverse states.

References

1. Bhagyashree.P & Gayathri.G.S. (2016). A Survey on Breadth First Search & Metropolis Hasting Random Walk. *International Journal of computer science and mobile computing*, 5(3). 1-5.
2. King, D. J., & Launchbury, J. (1995, January). Structuring depth-first search algorithms in Haskell. In *Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (pp. 344-354). ACM.
3. Nosrati, M., Karimi, R., & Hasanvand, H. A. (2012). Investigation of the*(star) search algorithms: Characteristics, methods and approaches. *World Applied Programming*, *2*(4), 251-256.
4. Bhasin, H., & Singla, N. (2012). Genetic based algorithm for N-puzzle problem. *International Journal of Computer Applications*, *51*(22).
5. Awerbuch, B. (1985). Reducing complexities of the distributed max-flow and breadth-first-search algorithms by means of network synchronization. *Networks*, *15*(4), 425-437.
6. Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM journal on computing*, *1*(2), 146-160.
7. Awerbuch, B. (1985). A new distributed depth-first-search algorithm. *Information Processing Letters*, *20*(3), 147-150.
8. Nosrati, M., Karimi, R., & Hasanvand, H. A. (2012). Investigation of the*(star) search algorithms: Characteristics, methods and approaches. *World Applied Programming*, *2*(4), 251-256.
9. Mühlenbein, H., Schomisch, M., & Born, J. (1991). The parallel genetic algorithm as function optimizer. *Parallel computing*, *17*(6-7), 619-632.
10. [10] Reinefeld, A. (1993). Complete Solution of the Eight-Puzzle and the Bene t of Node Ordering in IDA*. In *International Joint Conference on Artificial Intelligence* (pp. 248-253).
11. Mitchell, M., Holland, J. H., & Forrest, S. (1994). When will a genetic algorithm outperform hill climbing? In *Advances in neural information processing systems* (pp. 51-58).