International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT22831

A Comprehensive Study on 8 Puzzle Problem without Heuristic and with Heuristic Algorithm

Palash Dutta Banik*, Asoke Nath
Department of Computer Science, St. Xavier's College (Autonomous), Kolkata, India

Article Info
Volume 8, Issue 2
Page Number : 300-308

Publication Issue :
March-April-2022

Article History
Accepted: 10 April 2022
Published: 22 April 2022

ABSTRACT

Artificial Intelligence (AI) is now one most challenging and important field of computer science. With the help of AI one can make the machine can think like a human being and deliver solutions like a human brain. Now a day's AI replacing many old powerful technologies. There are many important areas where AI is used like a tool to achieve better solutions. Before introduction of AI algorithms, we were using search techniques like BFS, DFS, etc. But after the evolutionary change in computer science, now search techniques like Heuristic Search, A* algorithm, etc. were introduced to solve standard problems. With the help of some knowledge, one can easily find a solution for large and complex problems. In the present study the author will try to explore how one can solve 8-puzzle problem using heuristic search algorithm and also using non heuristic algorithm.

Keywords : Artificial Intelligence, Informed Search, Uninformed Search, Heuristic Function, Depth First Search, Breadth First Search, Heuristic Search Function, A* algorithm.

I. INTRODUCTION

Prior to 1949, computers were able to execute commands, but they could not remember what they were doing as they could not keep those commands. In 1950, Alan Turing discussed how to build smart machines and test this ingenuity in his paper "Computing Machinery and Intelligence". Five years later, the first AI program was launched at the Dartmouth Summer Research Project on Artificial Intelligence (DSPRAI). This event has been the catalyst for AI research over the next few decades.
Computers were fast, cheap and easily accessible between 1957 and 1974. Machine learning algorithms improved and, in 1970, one of the DSPRAI executives told Life Magazine that there would be a standard 3-8-8 machine-based human years. Despite its success, the inability of computers to store or process information quickly created obstacles in pursuit of practical wisdom for the next decade.

AI was revived in the 1980s with the expansion of the algorithmic tool kit and the most dedicated funds. John Hopefield and David Rumelhart introduced "deep learning" strategies that allow computers to learn using information. Edward Feigenbaum introduced "expert programs" that mimic human decision-making. Despite the lack of government funding and public

hype, AI prospered and many important goals were achieved in the next two decades. In 1997, former world chess and Grandmaster Gary Kasparov lost to IBM's Deep Blue, a computer program that played chess. That same year, a speech recognition software developed by Dragon Systems was implemented on Windows. Cynthia Breazeal also developed Kismet, a robot that can detect and express emotions. In 2016, Google's AlphaGo program defeated Go master Lee Se-dol and in 2017, Libratus, a supercomputer playing poker defeated the best human players.

AI (Artificial Intelligence) [1]: AI is a branch of computer science concerned with the study and creation of computer systems, which have some form of intelligence: Systems that learn new concepts and tasks, systems that can reason and draw useful conclusions about the world around us, systems that can understand a natural language and systems that can perform other types of work which needs human types of intelligence. Here intelligence is the ability to acquire, understand and apply the knowledge or the ability to exercise some thoughts and reasons.

II. Uninformed Search [2]

Before the area of AI started, the Uninformed Search techniques was available, which do not require any kind of knowledge or information about the goal state. It has more time and space complexity, this type of searching method is also known as the Brute force searching method.

Example - BFS, DFS, bidirectional search, etc.

III. Informed Search[3]

Informed search algorithms contain an array of knowledge such as how far we are from the goal node, path cost, how to reach the goal node, etc.

This is a type of searching method where we require the knowledge to find the steps to reach the solution or the goal state. Here, we always search with some information, for which this method can find quick solutions. This is also known as Heuristic Searching Method.

These informed search algorithms are more useful for large search spaces.

Example- A* algorithm, Heuristic Graph Searching, Best First Search, etc.

IV. Heuristic Function h(n)[4]

Heuristic Function is a type of function which is used in informed search and it finds the most promising path. It takes the current state of the node as it's input and produces the estimation of how close the present node is from the Goal node.

When state space exponentially grows or the problem is Non-polynomial, the Heuristic function is used to reduce time/space.

V. Main Focus

Given a 3×3 board with 8 tiles (every tile has one number from 1 to 8) and one empty space. The objective is to place the numbers on tiles to match the final configuration using the empty space. One can slide four adjacent (left, right, above, and below) tiles into the empty space.

i.e., in 8 puzzle problem we have total of 4 moves, Up, Down, Right, Left.

Suppose, we have 8 puzzle problem which has a **Starting State/Initial State** of

1	2	3
B	4	6
7	5	8

Our **Goal State** is –

1	2	3

4	5	6
7	8	B

Now, if uninformed search like DFS (Depth First Search) or BFS (Breadth First Search) is applied to find the Goal state without Heuristic then one has to consider the following:

VI.1.Using DFS[5][12]

It's a type of uninformed search or blind search algorithm. It always goes to the deepest node. It may be possible that DFS will not give us a solution every time. It uses Stack (**L**ast **I**n**F**irst **O**ut).

We can perform a depth-first search on state-space tree. In this way we can reach all possible states from the initial state.

In this solution, successive moves can take us away from the goal rather than bring us closer. The search of the state-space tree follows the leftmost path from the root regardless of the initial state. An answer node may never be found in this approach.

Time complexity – O (b^d), where "b" is the branch factor and "d" is depth.

VI.2.Using BFS:[6][11]

It's a type of uninformed search or blind search algorithm. In this method we search level wise that is why this method is called as Level Search Technique. If our goal state is closer from the initial state then BFS is better to use. It uses Queue (**F**irst **I**n**F**irst **O**ut).

We can perform a Breadth-first search on the state space tree. This always finds a goal state nearest to the root. But no matter what the initial state is, the algorithm attempts the same sequence of moves as DFS.

Time complexity – O (b^d), where "b" is the branch factor and "d" is depth.

VI.3. Explain with an example

Here, R = Right Move, L = Left Move, D = Down Move, U = Up Move

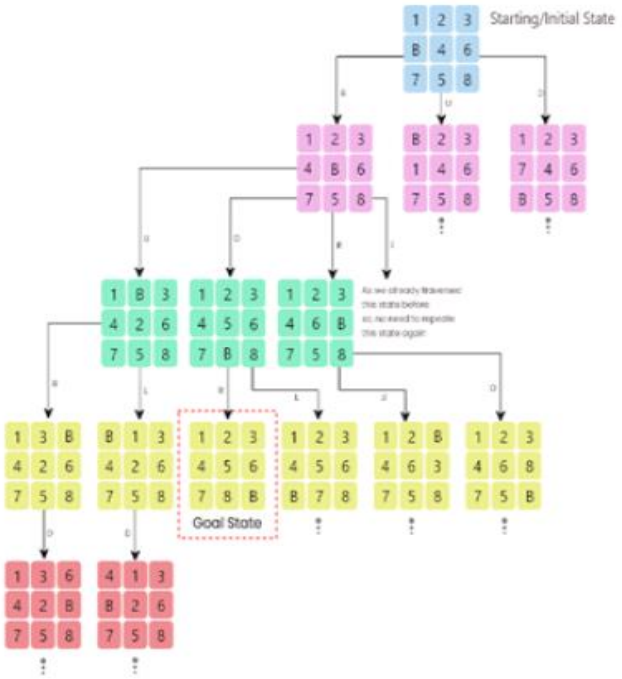

Fig.1 : 8-puzzle problem with uninformed search methods

VI.4. Understanding the Complexity with the help of this example

As we know the complexity is O (b^d), Where b is the branching factor and d is the depth of the solution. In simple words, d is the distance between the starting state to the goal state.

Depth depends on the complexity of the starting matrix and for this particular example, the depth is 20 and the branching factor is 3.

In worst case time complexity is – O $(3)^{20}$

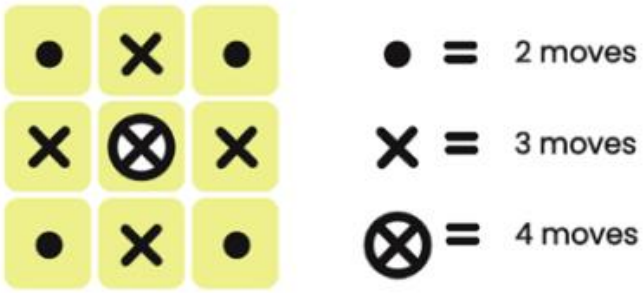

Fig. 2 : Branching Factor Calculation

Branching Factor calculation:

{(2 * 4) + (3 * 4) + (1 * 4)} = 24
now, the average branching factor is (24 / 9) = 2.66 ≈ 3, as 9 is the total number of spaces.

VI.5. Drawbacks of this type of uninformed search method and why we use the informed search method i.e., heuristic search

In case of easy problems, it is useful and an optimal solution is guaranteed but whenever the complexity of the matrix increases exponentially, the machine will take a large amount of time to reach the goal state. That is why to achieve the quickness to find the solution heuristic methods are more useful.

VI.6. Heuristic Search Algorithm[7]

This algorithm is a simple algorithm that guarantees finding a solution. If we can perform this algorithm systematically then surely there exists a solution. With the help of heuristic information, we can find which node is most promising by the heuristic function h(n). It takes the node "n" and returns the real non-negative number which is the average cost of the route from point "n" to the goal area. The function h (n) underestimates if h (n) is less than or equal to the actual cost of the low-cost route from point n to point.

A heuristic activity is a way of informing a search about the path to a goal. Provides an intuitive way to predict which node neighbor will lead to a goal.

Now, if we use informed search like Branch and Bound technique to find the Goal state with Heuristic then –

The search for an answer node can often be speeded by using an "intelligent" ranking function (Heuristic Function), also called an approximate cost function to avoid searching in sub-trees that do not contain an answer node. It is similar to the backtracking technique but uses a BFS-like search.

An example of the Branch and Bound approach is A* algorithm. So, we can use the A* algorithm to solve 8 puzzle problem in an intelligent way.

A* algorithm extends the path that minimizes the following function –

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

Here,

- "n" is the last node on the path.
- g(n) is the cost of the path from start node to node "n".
- h(n) is the Heuristic Function that estimates the cost of the cheapest path from "n" to the Goal node.

Note: A* algorithm maintains two lists, OPEN list and CLOSED list.

VI. A* Algorithm[8]

Step 1: Place the starting node into OPEN list and find it's f (n) value.

Step 2: Remove the node from OPEN list, if it has the smallest f (n) value.If, it is the Goal Node then Stop and Return Success.

Step 3: Else, remove the node from OPEN list and find all the successors of the node.

Step 4: Find the f (n) values of successors; place them into OPEN list and place the removed node into CLOSED list.

Step 5: Go to step 2.

Step 6: Exit.

Time complexity – O (b^d), where "b" is the branch factor and "d" is depth.

VI.1. Advantages

I. A* algorithm is a complete and optimal algorithm to find the best way to reach the goal state.

II. It is used to solve all most every complex problem.

VII. 2. Disadvantages

I. A* algorithm is complete if the branching factor is finite and every path has its fixed cost.
II. At some cases this A* algorithm can face complexity problems.
III. The speed execution of an A* algorithm is highly dependent on the accuracy of the Heuristic Function, which is used to compute the h(n).

VIII. Explain with an example

Here, R = Right Move, L = Left Move, D = Down Move, U = Up Move and H is the Heuristic value.

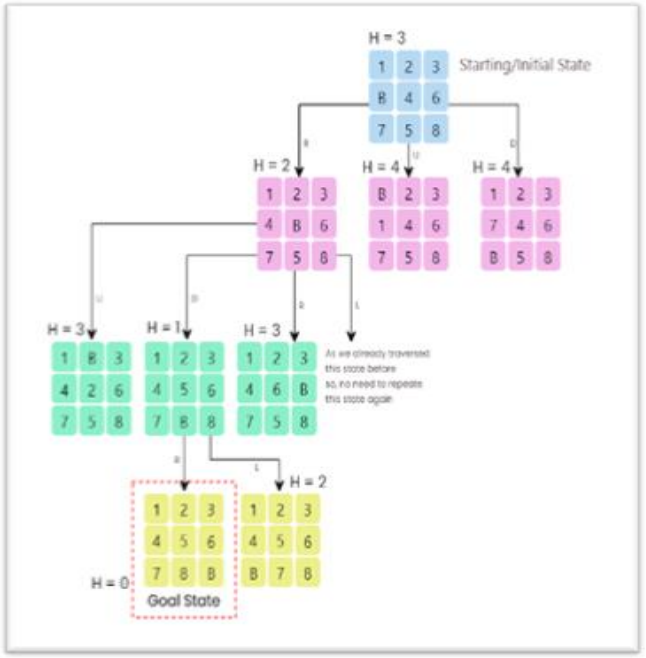

Fig.3 : 8-puzzle problem with A* algorithm

IX. PUZZLE PROBLEM WITH A* ALGORITHM PYTHON CODE

```
from copy import deepcopy
from colorama import Fore, Back, Style

#direction matrix
DIRECTIONS = {"U": [-1, 0], "D": [1, 0], "L": [0, -1], "R": [0, 1]}
#target matrix
END = [[1, 2, 3], [4, 5, 6], [7, 8, 0]]

# unicode for draw puzzle in command promt or terminal
left_down_angle = '\u2514'
right_down_angle = '\u2518'
right_up_angle = '\u2510'
left_up_angle = '\u250C'

middle_junction = '\u253C'
top_junction = '\u252C'
bottom_junction = '\u2534'
right_junction = '\u2524'
left_junction = '\u251C'

#bar color
bar = Style.BRIGHT + Fore.CYAN + '\u2502' + Fore.RESET + Style.RESET_ALL
dash = '\u2500'

#Line draw code
first_line = Style.BRIGHT + Fore.CYAN + left_up_angle + dash + dash + dash + top_junction + dash + dash + dash + top_junction + dash + dash + dash + right_up_angle + Fore.RESET + Style.RESET_ALL
middle_line = Style.BRIGHT + Fore.CYAN + left_junction + dash + dash + dash + middle_junction+ dash + dash + dash + middle_junction + dash + dash + dash + right_junction + Fore.RESET + Style.RESET_ALL
last_line = Style.BRIGHT + Fore.CYAN + left_down_angle + dash + dash + dash + bottom_junction + dash + dash + dash + bottom_junction + dash + dash + dash + right_down_angle + Fore.RESET + Style.RESET_ALL

#puzzle print function
def print_puzzle(array):
    print(first_line)
    for a in range(len(array)):
        for i in array[a]:
            if i == 0:
print(bar, Back.RED + ' ' + Back.RESET, end=' ')
            else:
```

```
print(bar, i, end=' ')
        print(bar)
        if a == 2:
            print(last_line)
        else:
            print(middle_line)
#it is the node which store each state of puzzle
class Node:
    def __init__(self, current_node, previous_node, g, h,
dir):
self.current_node = current_node
self.previous_node = previous_node
self.g = g
self.h = h
self.dir = dir
    def f(self):
        return self.g + self.h
def get_pos(current_state, element):
    for row in range(len(current_state)):
        if element in current_state[row]:
            return                                    (row,
current_state[row].index(element))
#it is a distance calculation algo
def euclidianCost(current_state):
    cost = 0
    for row in range(len(current_state)):
        for col in range(len(current_state[0])):
            pos = get_pos(END, current_state[row][col])
            cost += abs(row - pos[0]) + abs(col - pos[1])
    return cost
#getadjucent Nodes
def getAdjNode(node):
listNode = []
emptyPos = get_pos(node.current_node, 0)

    for dir in DIRECTIONS.keys():
newPos = (emptyPos[0] + DIRECTIONS[dir][0],
emptyPos[1] + DIRECTIONS[dir][1])
        if 0 <= newPos[0] <len(node.current_node) and 0
<= newPos[1] <len(node.current_node[0]):
newState = deepcopy(node.current_node)
newState[emptyPos[0]][emptyPos[1]]                    =
node.current_node[newPos[0]][newPos[1]]
newState[newPos[0]][newPos[1]] = 0
            # listNode += [Node(newState,
node.current_node, node.g + 1,
euclidianCost(newState), dir)]
listNode.append(Node(newState, node.current_node,
node.g + 1, euclidianCost(newState), dir))

    return listNode

#get the best node available among nodes
def getBestNode(openSet):
firstIter = True
    for node in openSet.values():
        if firstIter or node.f() <bestF:
firstIter = False
bestNode = node
bestF = bestNode.f()
    return bestNode
#thisfunctionn create the smallest path
def buildPath(closedSet):
    node = closedSet[str(END)]
    branch = list()
    while node.dir:
branch.append({
            'dir': node.dir,
            'node': node.current_node
        })
        node = closedSet[str(node.previous_node)]
branch.append({
        'dir': '',
        'node': node.current_node
    })
branch.reverse()
    return branch
#main function of node
def main(puzzle):
open_set = {str(puzzle): Node(puzzle, puzzle, 0,
euclidianCost(puzzle), "")}
closed_set = {}
    while True:
```

```
test_node = getBestNode(open_set)
closed_set[str(test_node.current_node)] = test_node
      if test_node.current_node == END:
         return buildPath(closed_set)
adj_node = getAdjNode(test_node)
      for node in adj_node:
         if str(node.current_node) in closed_set.keys()
or str(node.current_node) in open_set.keys() and
open_set[
            str(node.current_node)].f() <node.f():
            continue
open_set[str(node.current_node)] = node
      del open_set[str(test_node.current_node)]
if __name__ == '__main__':
   #it is start matrix
br = main([[1, 2, 3],
           [0, 4, 6],
           [7, 5, 8]])

print('total steps : ', len(br) - 1)
print()
print(dash + dash + right_junction, "INPUT",
left_junction + dash + dash)
   for b in br:
      if b['dir'] != '':
         letter = ''
         if b['dir'] == 'U':
            letter = 'UP'
elif b['dir'] == 'R':
            letter = "RIGHT"
elif b['dir'] == 'L':
            letter = 'LEFT'
elif b['dir'] == 'D':
            letter = 'DOWN'
print(dash + dash + right_junction, letter, left_junction
+ dash + dash)
print_puzzle(b['node'])
print()

print(dash + dash + right_junction, 'ABOVE IS THE
OUTPUT', left_junction + dash + dash)
```

OUTPUT:

X. Comparison between Informed and Uninformed Search methods[9][10]

In the case of uninformed searching, the time complexity is O (b^d), and the space complexity is O (b^d), and uninformed search methods always gives us an optimal solution.

In case of informed searching or heuristic method, we use some quickness to find out our result quickly.

The most important benefit of heuristic method is, we can convert a non-polynomial problem into a polynomial problem and find the solution quickly but heuristic method my not give us an optimal solution in some cases.

XI. CONCLUSION

Although Artificial Intelligence has some limitations, in case of the larger picture searching with some knowledge gives us extra benefits and AI gives the machines a human like thinking and processing ability. We can use this technology in various fields where we required human like intelligence from a machine. With the help of AI and Heuristic Search we can perform tasks in a better and efficient way and it can provide us a set of best solutions in a small amount of time. In some years AI will acquire the hole computer industry with its intelligence and knowledge.

XII.REFERENCES

[1]. Hewlett Packard Enterprise Website, 2020, https://www.hpe.com/in/en/what-is/artificial-intelligence.html?jumpid=ps_czxkg6qts3_aid-520061736&ef_id=Cj0KCQjwgYSTBhDKARIsAB8KukvXmK-Kaaxnev0jzv4LOdFDAshmpvqYDNq5_RRvA4v0-0HKpFRDL20aAsXTEALw_wcB:G:s&s_kwcid=AL!13472!3!558204152872!e!!g!!what%20is%20artificial%20intelligence!14386686693!128518517985&

[2]. Varun Singla, Gate Smashers YouTube, 13 Dec, 2019, https://www.youtube.com/watch?v=_CrEYrcImv0

[3]. Varun Singla, Gate Smashers YouTube, 15 Dec, 2019, https://www.youtube.com/watch?v=nmWGhb9E4es

[4]. javaTpoint Website,2022, https://www.javatpoint.com/ai-informed-search-algorithms#:~:text=Heuristics%20function%3A%20Heuristic%20is%20a,agent%20is%20from%20the%20goal.

[5]. geeksforgeeks website, 2022, https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/

[6]. geeksforgeeks website, 2022, https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/

[7]. javaTpoint Website, 2022, https://www.javatpoint.com/heuristic-techniques

[8]. javaTpoint Website, 2022, https://www.javatpoint.com/heuristic-techniques

[9]. geeksforgeeks website, 2022, https://www.geeksforgeeks.org/difference-between-informed-and-uninformed-search-in-ai/

[10]. Vinita, intellipaat website, 3 July, 2019, https://intellipaat.com/community/3654/what-is-the-difference-between-informed-and-uninformed-searches#:~:text=An%20uninformed%20search%20is%20a,current%20state%20to%20the%20goal.&text=Uses%20knowledge%20to%20find%20the%20steps%20to%20the%20solution.

[11]. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms, Third Edition, 2017, Page – 594 to 595

[12]. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms, Third Edition, 2017, Page – 603 to 608

AUTHOR PROFILE

Dr. Asoke Nath is working as Associate Professor in the Department of Computer Science, St. Xavier's College (Autonomous), Kolkata. He is engaged in research work in the field of Cryptography and Network Security, Steganography, Green Computing, Big data analytics, Li-Fi Technology, Mathematical modelling of Social Area Networks, MOOCs, Quantum Computing etc. He has published more than 257 research articles in different Journals and conference proceedings.

Mr. Palash Dutta Banik is a student of St. Xavier's College, currently pursuing M.Sc. in Computer Science. His interests lie in the field of Quantum Computing, Machine Learning, Coding, App Development, Cryptography and Network Security, UI Design, Cyber Security, AI and real-world project implementation of these fields.

Journal URL : https://ijsrcseit.com/CSEIT22831

Cite this article as :

Palash Dutta Banik, Asoke Nath, "A Comprehensive Study on 8 Puzzle Problem without Heuristic and with Heuristic Algorithm", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 8 Issue 2, pp. 300-308, March-April 2022. Available at doi : https://doi.org/10.32628/CSEIT22831