Informed and Uninformed Search Strategies On Water Sort Puzzle Problems

Farnood Sotoudeh
AmirKabir University of Technology
Jahrom, Iran
sotoodehfarnood@gmail.com

Parmida Hosseinmardi
Amirkabir University of Technology
Tehran, Iran
parmidamardii@gmail.com

Sara Mohammadzadeh
Amirkabir University of Technology
Tehran, Iran
sara.gsm5380@gmail.com

***Abstract*—Water sort puzzle, also known as ball sort puzzle, is a puzzle game which the player tries to sort colored water or balls in glasses in a way all the glasses are completely full or empty and each glass has the same color. The puzzle has different levels, as levels increase the difficulties increase. In this paper, different search algorithms in Artificial Intelligence name Uninformed and Informed searches are performed on the puzzle, to compare the performance of them and find the most optimal one.**

Keywords—Water Sort Puzzle, Ball Sort Puzzle; Artificial Intelligence, Uninformed search, Informed search;

I. INTRODUCTION

Artificial intelligence is a study to simulate human intelligence in agents to either act humanly, think humanly, think rationally or act rationally. To solve puzzle problems, we want the agent to act rational which means acting to achieve the best outcome.[1] Search algorithms are problem solving methods which are implemented on agents to act rational. There are two types of search algorithms, uninformed search and informed search. Uninformed search algorithms such as Depth First Search, Breadth First Search and Uniform Cost Search and Informed search algorithms such as Greedy Search and A* Tree Search are discussed here.[2]

Water sort puzzle starts from initial states, which there are some glasses filled by different colored water and two empty glasses. The goal is to reach the state in which same colors are poured into one glass. Rules of the games are:

- Pour is possible only if two glasses have same colors on top
- We can only pour top water from one glass to other
- Two same colors are merged into one with more volume
- Each glass can have certain amount of water and cannot be poured more [3]

In this paper, the goal is to find the path from initial state to goal state by mentioned algorithms and find the most optimal one and compare their performances.

II. SEARCH ALGORITHMS

Search algorithms can be used to solve various search problems in AI. Search problem can be defined as follows:

- The environment can be defined by a set of states which is called state space. Fig. 1 illustrates one state of the Water Sort Puzzle state space.
- The agent starts from initial state.
- There is a set of goal states which are the states the puzzle is solved
- Agent has several actions which by applying one of them, the state will change to another state
- A transition model returns the result of the action
- Action cost function is used to measure the performance. It returns a numeric cost of applying action in one state to reach another state

A path is a sequence of actions and the solution of the puzzle is a path from initial state to one of the goal states.[1]

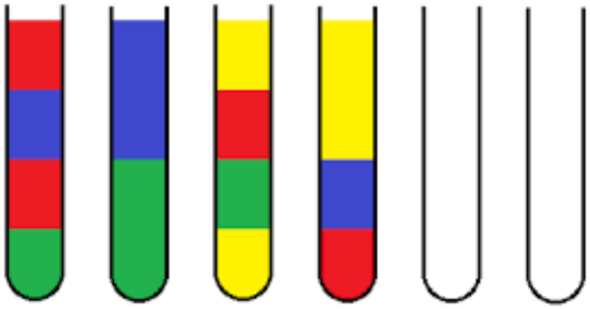

Fig.1. Sample state of Water Sort Puzzle state space

There are two types of search algorithms which are used based on the search problem:

- Uninformed search algorithms: They do not have additional information about the goal and are known as blind search. These algorithms start from the initial state and traverses the tree to reach goal state. Breadth-first search, Depth First Search and Uniform Cost Search are discussed in this section.
- Informed search algorithms: They have additional information about the goal by something called heuristic. Greedy Search and A* Tree Search that are discussed, are in this section. [2]

Search algorithms have four properties to compare their efficiency:

- Completeness: The algorithm is complete if there exists any solution, it returns the solution
- Optimality: The algorithm finds the best solution
- Time Complexity: Amount of time the algorithm needs to be completed
- Space Complexity: Maximum amount of memory needed during the search [4]

There are also three terms which we need to know:

- b: maximum branching factor in a tree
- d: maximum branching factor in a tree
- m: maximum depth state space

A. *Breadth-first search*

BFS is an uninformed search which starts from the initial state names root and travers the tree breadthwise. It first expands the root, then its successors, then expands their successors, and so on. A first-in-first-out queue is implemented to know the order of the nodes need to be expanded. Older nodes get expanded before new ones. Fig.2 shows the algorithm.[5]

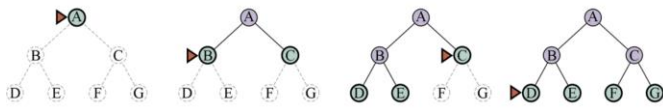

Fig.2. BFS algorithm [1]

BFS is a complete algorithm if b is finite and is optimal and if the costs are the same for each step. It's time and space complexity is $O(b^{d+1})$. It is clear that the space is big problem here.

B. *DFS*

DFS is also an uninformed search. The traverse starts from root and continues as far as possible along each branch till it reaches a leaf then it will backtrack and check for unexplored states.[2] Here a last-in-first-out queue is implemented to know the order of the nodes need to be expanded. Fig.3 shows the algorithm.

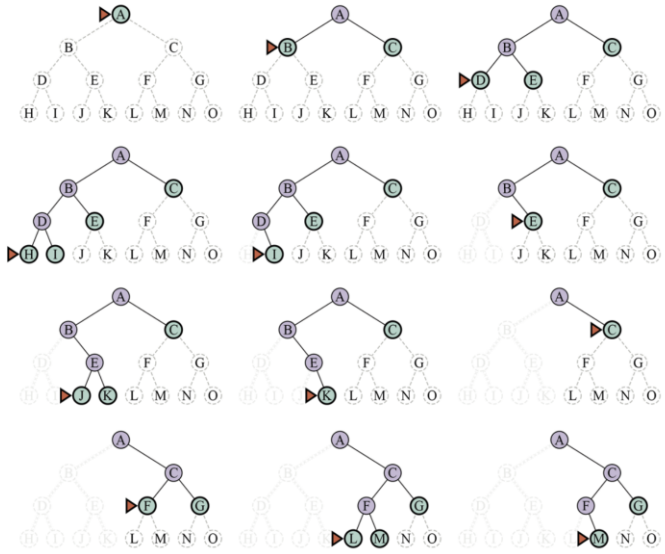

Fig. 3. DFS algorithm [1]

DFS is not a complete algorithm because it may fail in infinite depth space or spaces with loops, but it is complete in finite spaces. It is not optimal. Its time complexity is $O(b^m)$ and space complexity is O(bm). So, DFS requires less memory than BFS.

C. *Uniform Cost Search*

Another uninformed search is UCS. UCS is used when the costs are not the same. It doesn't go breadth or depth. It starts from root and expands it and choose the unexpanded node with lowest cost and expands it. Again, it looks for an unexpanded node with lowest cost and expands it, and so on. UCS is a complete algorithm if step cost $\geq \varepsilon$. It is also optimal. If the cost of the optimal solution be C^*, the time and space complexity of UCS is $O(b^{\lceil \frac{C^*}{\varepsilon} \rceil})$.

D. *Greedy Search*

Greedy search is the first informed algorithm which will be discussed. It is a form of best-first search. Best- first search is an algorithm to find shortest path from root to goal node. [6] Here the node with the lowest h(n) value will be expanded first. h(n) is the heuristic function which estimates the cost of the cheapest path from note n to goal state. It is not complete because it may

get stuck in loops but it can be complete in finite space with repeated state checking. It is not optimal. Both time and space complexity are $O(b^m)$.

E. A Search*

A* is the second informed search which we will discuss. It is the most common informed search. It is also a best-first search with evaluation function f(n). f(n) is summation of h(n) and g(n). h(n) is the heuristic function which estimates the cost of the cheapest path from node n to the goal state and g(n) is the path cost from initial node to node n. [1] Here the node with the lowest f(n) value will be expanded first. A* is complete if there be a lower bound on cost. It is optimal if h(n) is admissible, which means it never overestimates the cost of reaching goal from current node, and it is monotonic. Its time complexity depends on the heuristic function. For space complexity, it keeps all the nodes in memory.

III. Results and Discussion

To solve the Water Sort Puzzle, we have used the mentioned algorithms. Fig.1 illustrated the state of the puzzle, to implement that a matrix is used that each row shows a bottle and each column shows the bottle level. I.e., the matrix[i][j] shows the color of water in level j of bottle i. So, DFS and BFS can be applied easily on the puzzle. For UCS, we need to calculate the cost of each node. Here, Action is to pour one bottle into another. Each action's cost is 1, which means from initial state how many actions were applied to reach the current state. In Greedy search, we should calculate the heuristic. The heuristic is implemented as follow:

- Check each glass separately
- Count the number of same colored water on the top of the glass minus 1. For example. Fig.1 second glass has 2 same-colored waters at top
- Multiply the result by 100
- Multiply the result by -1 because the lowest h(n) is needed

So, the value of each bottle from left to right in fig. 1 is 0, -100, 0, -100, 0, 0. At the end, we should calculate the h(n). h(n) is equal to summation of values of the bottles. In A* we should calculate the f(n) which is summation of h(n) and g(n). h(n) is same as the one in Greedy Search and g(n) is calculated in the same way as UCS. To compare the performance of these algorithms we use the puzzle in fig. 4. The number of nodes expanded and number of moves from initial state to goal state in each algorithm is shown in table.1

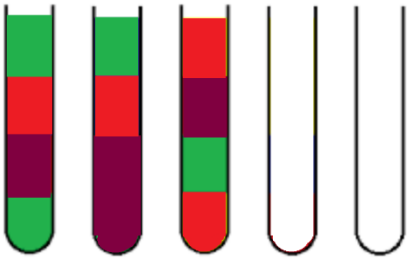

Fig. 4

Algorithm	Node expanded	Move's number
DFS	73	19
BFS	1684	10
UCS	1684	10
Greedy	87	10
A*	87	10

Table. 1

It is observed that A* and Greedy perform the best. DFS is not optimal and can not find the shortest path. BFS and UCS expand many nodes to find the goal. Here BFS and UCS act the same because the g(n) for all the nodes at same level is the same and as the generation passes, the g(n) increases too. Also, A* and Greedy act the same because g(n) is too small comparing to h(n).

Conclusion

So, A* is best algorithm comparing to others if a good heuristic is chosen. DFS uses less memory than BFS and UCS but it is not optimal. Space is big problem in BFS. Greedy goes directly to the goal which may lead to local subgoal and UCS check lots of unnecessary nodes. So, A* can be a good choice because it is a combination of UCS and Greedy search algorithms.

REFERENCES

[1] Artificial Intelligence, A Modern Approach

[2] https://www.geeksforgeeks.org/search-algorithms-in-ai/

[3] https://apps.apple.com/us/app/water-sort-puzzle-sort-color/id1597897013

[4] https://www.javatpoint.com/search-algorithms-in-ai

[5] https://www.analyticsvidhya.com/blog/2021/02/uninformed-search-algorithms-in-ai/

[6] https://www.mygreatlearning.com/