# Analyses of Efficiency and Effectiveness of Uninformed Search Algorithms

2/6/18

## I. Introduction

We have learned five search algorithms for uninformed search strategies to solve problems so far: Breadth-first search, depth-first search, limited-depth search, iterative-deepening search, and uniform-cost search. To test the efficiency and effectiveness of each algorithm, we generate two problems to exam such properties. The first problem is the eight-puzzle problem which contains eight numbered tiles and a blank space in a 3 * 3 board. By moving an adjacent tile to the blank space, we try to make the initial configuration to match with the goal configuration. In the problem, the search tree can be unlimited, and solutions are somewhere in the tree.

The second problem is the jumping problem: “Dr. Smith must hop across a wide chasm with only a handful of supports to step upon along the way” (2018, Weinman). We could input the number of supports to make the problem easier or more difficult. The depth of the search tree in the problem should be smaller than the inputted supports because our goal is to “minimize the total number of hops it takes him to reach the other side” (2018, Weinman). By performing easy and difficult eight-puzzle, and short and long jumping problem in each search algorithm, we find that breadth-first and uniform-cost search are the best algorithms for the eight-puzzle problem, and depth-first search and depth-limited search with a big enough limit are the best methods to solve the jumping problem.

## II. Results and Analysis

The efficiency of search algorithms varies when the agent solves distinct types of problems. First, we inputted the moves of the eight-puzzle problem as three, which means it should take no more than three actions to achieve the goal state.

| Search Type | Action | Expansion |
|---|---|---|
| breadth-first-search | 3 | 10 |
| depth-first-search | 31 | 31 |
| depth-limited-search (depth = 3) | 3 | 4 |
| depth-limited-search (depth = 30) | 29 | 184672 |
| iterative-deepening-search | 3 | 17 |
| uniform-cost-search | 3 | 17 |

Breadth-first search and depth-limited search with limit three work as the most efficient search algorithms because they expand least nodes to achieve the solution. Iterative-deepening-search and Uniform-cost search are appropriate algorithms as well. Iterative-deepening search would go through each node with the increasing depth. Since then, if the required depth is small enough, the method can find the solution effectively. Uniform-cost-search would expand the node with the lowest path cost first. Thus, in a problem with equal step costs, it works as iterative-deepening search. Depth-first-search and depth-limited-search with limit 30 are relatively inefficient. By using depth-first-search, the agent keeps expanding the deepest node to find a nonoptimal solution. The least efficient search algorithm of the problem is depth-limit search with limit 30, it expands more than 18,000 times to find a possible solution. In conclusion, breadth-first search, iterative-deepening search, and uniform-cost search expand only necessary nodes and get the optimal solution effectively.

Second, we inputted the moves as nine and got the following result:

| Search Type | Action | Expansion |
|---|---|---|
| breadth-first-search | 9 | 338 |
| depth-first-search | 2659 | 2714 |
| depth-limited-search (depth = 3) | #f | #f |
| depth-limited-search (depth = 30) | 29 | 107399 |
| iterative-deepening-search | 9 | 789 |
| uniform-cost-search | 9 | 559 |

Once the eight-puzzle problem gets harder, search algorithms work in the same way as they do in solving an easy eight-puzzle problem. The depth-first search would expand more than 2,000 times to get the solution and obviously, the solution is not the optimal one. Depth-limit search with limit 30 works inefficiently, and it does not find the optimal solution as well. Thus, breadth-first search, iterative-deepening search, and uniform-cost search are still more efficient search algorithms to achieve the goal state even for a harder eight-puzzle problem.

To discuss the solution costs for each search algorithm, breadth-first search, iterative-deepening search, depth-limit search with limit three, and uniform-cost search take minimal actions to find the solution, but depth-first search and depth-limit search with a limit 30 could not find the optimal solution even though they are complete. The solution they find requires more steps, and thus the solution costs of these two search algorithms are relatively enormous, which are 31 and 29 actions required to solve the problem compared to the best solution of three actions.

When the problem becomes harder, breadth-first search, iterative-deepening search, and uniform-cost-search can still find the optimal solution by taking nine steps to achieve the goal state. However, if we set the limit of depth-limit search as a small number, we may not get the solution because the required depth is larger than the limit. With unnecessary expansions, depth-first-search and depth-limited search with limit 30 take a vast number of steps to get the solution, so these two algorithms are not recommended to be used when we need to minimize the solution cost.

In sum, when the agent solves a problem for what the solution is somewhere in the search tree, but the search tree can develop infinitely, breadth-first search, iterative-deepening search, and

uniform-cost search are better algorithms than depth-first search and depth-limit search.

Third, we created a jumping problem which the number of supports is three:

| Search Type | Action | Expansion |
|---|---|---|
| breadth-first-search | 3 | 4 |
| depth-first-search | 3 | 4 |
| depth-limited-search(depth = 3) | 3 | 4 |
| depth-limited-search(depth = 30) | 3 | 4 |
| iterative-deepening-search | 3 | 10 |
| uniform-cost-search | 3 | 4 |

To solve the problem, breadth-first-search, depth-first-search, depth-limited-search, and uniform-cost search expand only four times, but the iterative-deepening search expands ten times. The result is exactly what we expected because iterative-deepening search would expand nodes by increasing depth each time so that it takes redundant procedures to achieve the same solution as other algorithms do.

Next, we increase the course length of the jumping problem to 26:

| Search Type | Action | Expansion |
|---|---|---|
| breadth-first-search | 9 | 6712 |
| depth-first-search | 9 | 43 |
| depth-limited-search(depth = 3) | #f | #f |
| depth-limited-search (depth = 30) | 9 | 43 |
| iterative-deepening-search | 9 | 10329 |
| uniform-cost-search | 9 | 14046 |

Now, depth-first-search and depth-limited-search with a big enough limit work better than other algorithms because the solution of the problem would be in the deeper location in the search. Since then, breadth-first-search, iterative-deepening-search, and uniform-cost-search would

have unrequired expansions to get the deeper node. It is worth mentioning, if the limit for depth-limited search is smaller than the depth of solution, it cannot return a solution to the problem.

To consider action costs of different algorithms, all algorithms take three actions to solve the short jumping problem and take nine actions to solve the long jumping problem. This outcome happens because the depth of the search tree is limited by the number of supports. Therefore, depth-first search and depth-limited search would not continue to expand the tree infinitely but becomes the most efficient method to get the optimal solution.

In short, when the agent faces a problem with the limited length of the search tree, algorithms could achieve the optimal solution if the limit of limited-depth search is large enough. But the nodes that would be expanded will be different.

## III. Conclusion

In such problems, all algorithms could find a solution except for the depth-limited search with a limit three. The differences between the solutions are the efficiency and required actions for these algorithms. In a problem with infinite search tree and same step cost, such as the eight-puzzle problem, the most efficient algorithms would be breadth-first search and uniform-cost search because they would go through nodes for each depth. The unrecommended algorithms would be depth-first search and depth-limit search because they take surplus actions to get the solution. However, If we need to solve a problem which the depth of search tree is limited, such as the jumping problem, depth-first search and depth-limit search with a big enough limit are encouraged to use, because they can ignore disturbed nodes to save workspace.

## IV. Reference

Weinman,J.(2018). [ebook] Available at: https://www.cs.grinnell.edu/~weinman/courses/CSC261/2018S/assignments/search.pdf [Accessed 6 Feb. 2018].