A Comparison of Search Algorithms on the Tower of Hanoi Problem

Wen Chuan Lee
College of Science and Engineering
University of Minnesota, Twin Cities
Minneapolis, MN
Email: lee@leewc.com - leex7095@umn.edu

***Abstract*—The abstract goes here. (To be written in later drafts.)**

I. Introduction

A. The Algorithms

Since the late 1950s, search has been considered to be influential to the field of Artificial Intelligence. Early AI programs had an algorithm that would search through possible states to arrive at an eventual solution, some even being able to backtrack when the algorithm reached a dead end. This paradigm was considered by McCorduck and others to be "reasoning by search" [1].

One of the more well-known algorithms of search is the breadth-first search (BFS). As an uninformed search algorithm, BFS expands the root node and then the successors of the root node, followed by their successors and so on [3]. All nodes at a specific depth are searched before the following depth are expanded. An advantage of the BFS algorithm is that it is complete, in which breadth first search will eventually find a solution that exists in the state space, if there is one. It is also considered to be the optimal if and only if the path cost is a nondecreasing function of the depth of the node [3].

Bidirectional breadth-first search (bidirectional BFS) is a modified version of BFS in which two simultaneous BFS searches are started, one from the initial state and one from the final state. However, the key difference is that instead of a goal state test at every level, this is replaced by a check to see if the two frontiers have an intersecting node. If there is one, then the solution path is found. The distinction on this is the assumption that the sum of 2 simultaneous searches will be less than the complexity of a single search from the initial state.

B. The Tower of Hanoi Problem

The tower of Hanoi is a mathematical and logical toy problem that has exactly 3 pegs with a varying number of disks. The goal state of the problem is to have all disks moved from one peg to another peg with regards to specific rules.The complexity of the problem increases exponentially with the increased number of disks. This is because with *n* number of disks, the *minimum* number of moves required to solve the Towers of Hanoi puzzle is $2^n - 1$. [2]

This mathematical problem has 3 simple rules:

1) Only one disk can be moved at a time.
2) A larger disk cannot be moved on top of a smaller disk.
3) Only the top most disk of each peg can be moved to another peg.

In this paper I attempt to make an experimental analysis on the performance and memory of complexity of the breadth first search and the bidirectional breadth first search algorithm on the Tower of Hanoi problem. The problem will be represented in Python along with the search algorithms, after which problems of varying difficulty are provided to the algorithm by increasing the number of disks in the problem. Finally I will analyze and discuss the memory used and the time complexity of each algorithm to solve each of the problem, I will also provide a hypothesis by another team member, which consists of other algorithms used to solve another problem, the sliding puzzle, in which an A* algorithm and backtracking DFS algorithm is used.

II. Hypothesis and Theoretical Analysis

Bidirectional BFS will have a better performance and time complexity than BFS itself. As BFS has to explore each node at every level before moving on to the next, the memory and time required will be much longer while Bidirectional BFS will reduce the time of exploring each node at every level by running two simultaneous searches, that is one from the problem state and one from the goal state.

Both BFS and bidirectional BFS are both complete and optimal if the step costs are all identical (in the case of the Towers of Hanoi problem they are). As bidirectional BFS uses BFS in both simultaneous searches, the algorithm shares the complete and optimal properties of BFS.

The difference however, is the theoretical bounds on time and space. For BFS, the time and space complexity is

$$O(b^d) \tag{1}$$

where *b* is the branching factor of the tree and *d* is the depth of the tree. Specifically for the Towers of Hanoi problem to be represented, the branching factor will be 1 or 2 as only the top most pegs can be moved at each time, as moving back to a previous state is considered useless and does not cause the algorithm to Therefore, the depth of the tree for this solution will be:

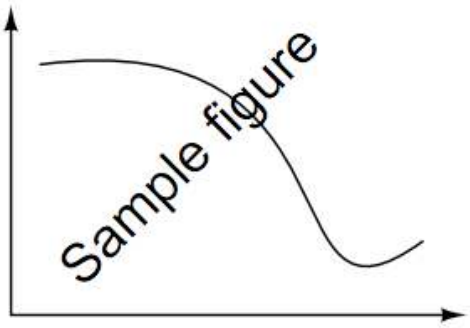

Fig. 1. Graph of Runtime to Number of Disks

$$d = 2^n - 1 \tag{2}$$

For Bidirectional BFS, the time and space complexity will be

$$O(b^{d/2}) \tag{3}$$

By having 2 simultaneous searches, bidirectional BFS attempts to run in less than the time and space complexity of BFS. If the following is true then bidirectional BFS will be much faster than running a single BFS algorithm only. [3]

$$O(b^{d/2}) + O(b^{d/2}) \leq O(b^d) \tag{4}$$

III. Experimental Set-Up

The set up for testing both algorithms will be done by varying the number of disks, *n*. This increases the size of the graph as well as the depth of the graph of states. A state graph (that also contains illegal actions) has a maximum of 3^n for a 3-peg Tower of Hanoi problem. The solution depth will have a size of $2^n - 1$ as previously mentioned.

After running 2 algorithms with an increasing number of disks and recording the number of nodes traversed to find a solution as well as the total computation time to find the solution, a graph of Number of Nodes Traversed to the Number of Disks will be generated, where both the algorithms can be compared. A graph of Computation time to the Number of Disks will also be generated for a concrete analysis of algorithms.

A. Graph of Number of Nodes Traversed to Number of Disks

(to be added later)

B. Graph of Runtime to Number of Disks

(to be added later)

C. Graph of Program Memory Usage to Number of Disks

(to find a method of keeping track of memory usage) (to be added later)

IV. Results

Discuss results here.

V. Conclusion and Future Work

The conclusion and discussion of what to do going forward go here.

References

[1] Pamela McCorduck. *Machines who think, 2nd ed.* A K Peters/CRC Press, 2004.
[2] Miodrag Petkovi. *Famous Puzzles of Great Mathematicians*. AMS Bookstore.
[3] Peter Norvig Stuart Russell. *Artificial Intelligence, A Modern Approach, Third Edition*. Pearson, 2009.