Comparison of Tarrys Traversal and Awerbuchs Depth first search

Brian Bartman
Computer Science
Kent State University
Kent, Ohio
bbartman@kent.edu

***Abstract* — The results and summarization of the different comparisons of Tarry's and Awerbuch's algorithm. The tests compared message complexity, time complexity and serial execution time.**

Keywords- Distributed algorithms;Depth first search; Traversal algorithms; Wave algorithms;

I. INTRODUCTION

In this paper I present comparisons of different factors of both Tarry's [2] and Awerbuch's [1] algorithm. Each test was performed on multiple different topologies with increasing numbers of nodes. The experiment was designed to show the differences between both Tarry's and Awerbuch's algorithms, and to than make a decision as to which algorithm would be better used under which condition.

The rest of the paper is structured as follows: section II contains a brief background on the two different algorithms, section III contains an explanation of the experiment and the variables being tested, section IV presents the results of the experiment, section V presents my conclusion and section VI discusses future work.

II. TARRY'S AND AWERBUCH'S ALGORITHMS

Bothe Tarry's and Awerbuch's algorithms can be used to explore a network and retrieve information about it in the form of a tree. While both do produce a tree only Awerbuch's produces a depth first spanning tree. The algorithms have different time complexities both of which will be demonstrated within the experiment.

Awerbuch's algorithm has a time complexity of 4N-2 and a message complexity of 4E, where N is the number of nodes within a graph and E is the number of edges.

Tarry's algorithm is a traversal algorithm, which means that both its time and message complexity are going to be the same. Tarry's has a time complexity of 2E.

III. EXPERIMENT

I tested message complexity, time complexity and serial execution time. The time and message complexities were used to show exactly how long and how much would be sent during the execution of each algorithm. The third measure mean of serial execution time is mean to determine exactly how many guarded command would need to be executed in order to complete the algorithm, this measurement could also be used as a more generalized form of how much work is being done by the entire system as whole.

Each algorithm was tested on the following topologies: chain, clique, ring and star. Each of the different topologies was tested for constructions between three and 25 nodes. Each number of nodes between three and 25 was each tested five times and the average from each of those execution taken.

IV. RESULTS

The results of both the time and message complexities match the expected results based on the general time and message complexities determined by the algorithm. Tarry's algorithm performed better on sparsely connected topologies in both time and message complexities as expected. Awerbuch's algorithm performed better on the densely connected clique topology because the number of edges within a clique increases exponentially.

Consider a chain as an example of a sparsely connected graph which has N-1 edges in its graph. Tarry's out performs Awerbuch's algorithm on a chain topology as seen for message complexity in figure 4.1 and for time complexity on figure 4.2.

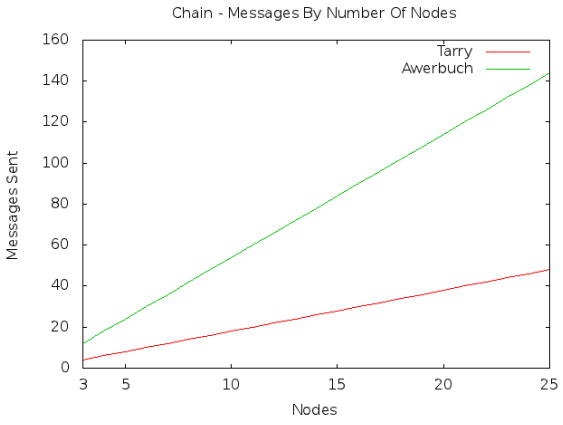

Figure 4.1: Chain topology messages by number of nodes.

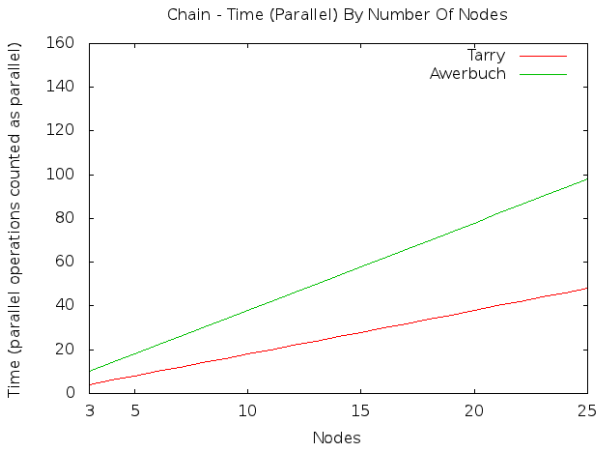

Figure 4.2: Chain topology time by number of nodes.

While sparsely connected graphs are best for Tarry's algorithm when a different topology which is more densely connected in the case of a clique Awerbuch's outperforms Tarry's in time complexity but not in the number of messages sent as shown in figure 4.3 and figure 4.4.

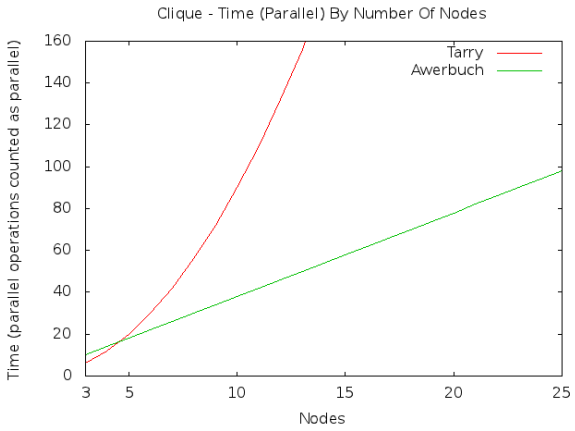

Figure 4.3: clique topology time by number of nodes.

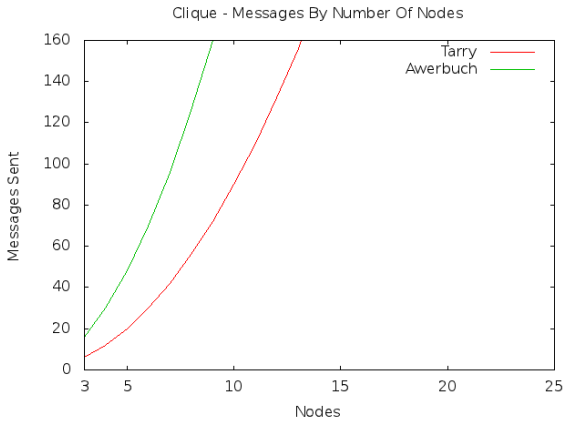

Figure 4.4: clique topology messages sent by number of nodes.

As figure 4.5 shows for the more densely connected clique graph Awerbuch's algorithm does better.

The cases which I find more interesting is those in which compare the amount of work done by both algorithms, where work is defined as the number of guarded command executions divided by the number of nodes. Figure 4.5 depicts the amount of work being done by both algorithms based on the number of guarded command executions for a clique topology. This is also depicted as non-parallel time because if the algorithm was run sequentially and simply than counting each guarded command execution that give you the amount of work.

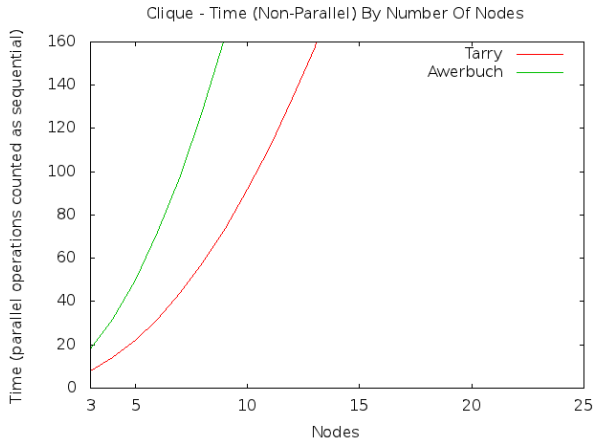

Figure 4.5: Clique topology work graph.

Figure 4.5 shows the two different amounts of work being done by both Tarry's and Awerbuch's algorithm where Awerbuch's requires much more than tarry's does.

To verify the correctness of my algorithms execution the program which runs the guarded commands uses software to output graphs of traversals and the paths which messages take through the different channels.

V. Conclusion

After comparing the results from both algorithms I feel that there is no better algorithm but which algorithm you use needs to be based on several factors. The different factors consist of exactly how many nodes are in your graph, how densely or sparsely connected the graph is and how much of each of those nodes do you need to devote to actually completing the task.

VI. Future Work

For future work I would like to develop a measure of connectivity for which algorithm will perform better. So for an arbitrary topology what is the number of either edges or nodes for which Tarry's will perform better and which are those which Awerbuch's will perform better.

[1] A. Baruch, "A New Distributed Depth-First-Search Algorithm," Information Processing Letters 20, pp. 147-150, 1985.

[2] G. Tarry, "Le Problem Des Labyrinthes," *Nouvelles Annales de* Mathematique 14, 1895.