Comparison of Tarry’s Algorithm and Awerbuch’s Algorithm (Spring 2010)

Patel Sanjitkumar M.
Department of Computer Science
Kent State University, Kent, OH - 44242, USA
spatel44@kent.edu

Abstract: **This paper is about comparing Tarry’s algorithm and Awerbuch’s algorithm by varying certain parameters. Tarry’s algorithm is wave algorithm as well as traversal algorithm, while Awerbuch’s algorithm is well known traversal algorithm. In this paper an attempt has been made to compare both algorithms, in terms of their time complexity and message complexity, by varying number of nodes and density of network. The result shows that for message complexity, Tarry’s algorithm performs better than Awerbuch’s algorithm, and for time complexity, Awerbuch’s algorithm performs better than Tarry’s algorithm as number of nodes increases, but for less number of nodes and lower connection probability, Tarry’s algorithm performs better than Awerbuch’s algorithm.**

I. Introduction

In this paper experiments are done to compare two traversal algorithms: Tarry’s algorithm and Awerbuch’s algorithm. Traversal algorithms are the wave algorithms with the following properties:

1. Each computation contains one initiator which starts computation by sending one message.
2. When a process receives a message it either sends out one message or decides.

And a wave algorithm is a distributed algorithm which satisfies following properties:

1. Termination: Each computation is finite.
2. Decision: Each computation contains at least one decide event.
3. Dependence: In each computation each decide event is causally preceded by an event in each process.

The experiments were done to capture message complexity and time complexity of both the algorithms. Message complexity is number of messages it takes the algorithm to carry out specified task, and time complexity is the number of messages in the longest chain of causally dependent events. Data were collected by varying number of nodes and density of the network. Density of network means connection probability and connection probability means if there is an edge between two nodes or not.

The remainder of the paper is organized as follows. The next section gives brief descriptions of the subject algorithms. In Section III, an elaborate description of the experimental set up is given. In Section IV, the results obtained are shown with graphs. In Section V, analysis of results is given. The succeeding section discusses conclusions and future work.

II. Algorithms

This section gives a brief description of algorithms used in this paper. In Tarry’s algorithm, a process never forwards the token twice through the same channel. The initiator starts the algorithm by arbitrarily choosing a neighbor and sending it the token. For each process p, upon receipt of a token from a process q, if father is undefined then q is set as the father. Token is forwarded to a neighbor if it has not been sent before to that neighbor. If no such neighbor exists then token is sent to the father. The algorithm terminates when the token has visited all the processes and at the end, the initiator receives it and decides. Each computation of Tarry’s algorithm defines a spanning tree of the network. The root of this tree is the initiator, and each non-initiator knows its own father. In Awerbuch’s algorithm, a node holding the token for the first time informs all neighbors except its father. It prevents token forwarding over frond edges because each process knows which neighbors were visited before it forwards the token. The node notifies its neighbors that it is visited by sending <vis> messages to them. The token is only forwarded when these neighbors all acknowledged reception. The token is only forwarded to nodes that were not yet visited.

III. Experimental setup

Since, both algorithms work on arbitrary topology, it was necessary to create random topologies. According to [1], random topologies can be represented in matrix form. This kind of matrix are called Adjacency matrix. Considering our topologies as graphs in which each vertex represents node and each edge represents communication channel, according to incidence matrix, if there is an edge between two vertexes, it can be represented as “1” in the matrix, and if there is no edge between two vertexes, it can be represented as "0" in the matrix. Based on this concept, random graphs were generated using random function, which acted as an arbitrary topology for my experiments. Fig 1 shows sample adjacency matrix and

	1	2	3	4
1	0	1	0	1
2	1	0	1	1
3	0	1	0	1
4	1	1	1	0

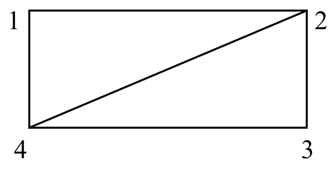

Fig 1. Example of adjacency matrix and corresponding graph

corresponding graph where numbers from 1 to 4 shows nodes and 0 and 1 represents if corresponding nodes have an edge(1) or not(0).

The number of nodes used for the experiments were 3 to 50, because 2 nodes are trivial and don't provide any significant value. Significant difference was seen at nodes 50, so that was the stopping point for experiment. Connection probability used was 25%, 50% and 75%. The number of nodes and connection probability needs to be input at starting of the experiment and program generates graph with given connection probability and number of nodes. Program first generates random graph and checks if graph has connection probability same as it was input, and if it is so, program starts simulation, otherwise it generates other random graph and checks again.

IV. Results

Data were collected by running program three times for each experiment. The results shown below shows average value of data collected. First result shows graph of time complexity with connection probability of 25% as shown in Fig 2. Horizontal line represents number of nodes and vertical line represents time complexity. Result shows that Tarry's algorithm performs better during number of nodes 4 to 14. During number of nodes between 14 and 18 both algorithm performs similar. And after number of nodes 18, Awerbuch's algorithm outperforms Tarry's algorithm. It can be observed that curve for Tarry's algorithm is quadratic while curve for Awerbuch's algorithm is linear.

Second result shows graph of time complexity at connection probability of 50% as shown in Fig 3. Horizontal line represents number of nodes and vertical line represents time-

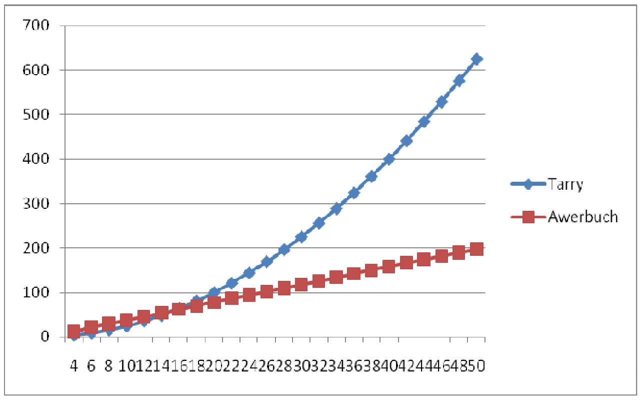

Fig 2. Comparison of time complexity at connection Probability of 25%

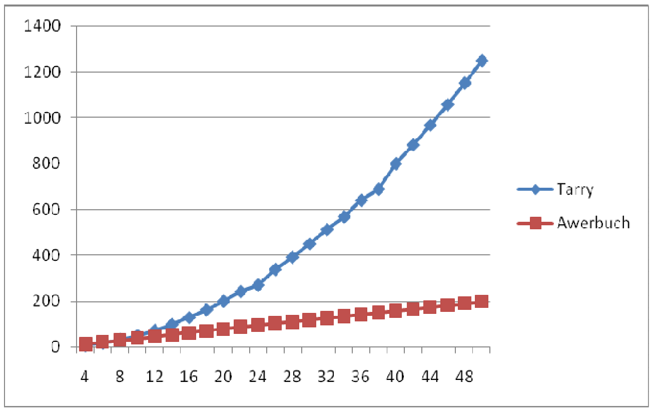

Fig 3. Comparison of time complexity at connection Probability of 50%

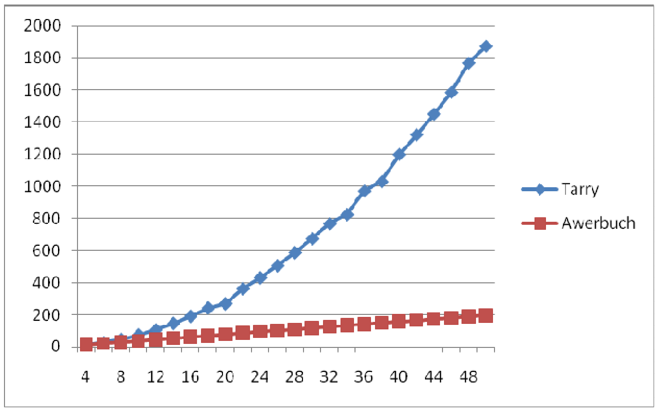

Fig 4. Comparison of time complexity at connection Probability of 75%

complexity. Result shows during number of nodes between 4 and 10 both algorithms perform similar. And after number of nodes 10, Awerbuch's algorithm outperforms Tarry's algorithm. It can be observed that curve for Tarry's algorithm is quadratic while curve for Awerbuch's algorithm is linear.

Third result shows graph of time complexity with connection probability of 75% as shown in Fig 4. Horizontal line represents number of nodes and vertical line represents time complexity. Result shows during number of nodes between 4 and 8 both algorithms perform similar. And after number of nodes 8, Awerbuch's algorithm outperforms Tarry's algorithm. It can be observed that curve for Tarry's algorithm is quadratic while curve for Awerbuch's algorithm is still linear.

Following are the results for comparison of message complexity at connection probability of 25%, 50% and 75%.

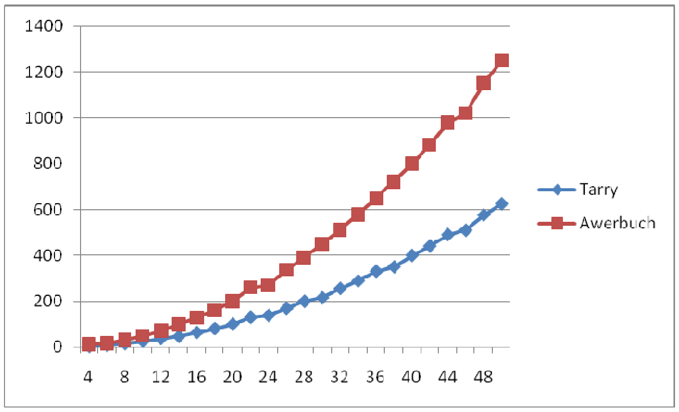

Fig 5. Comparison of message complexity at connection Probability of 25%

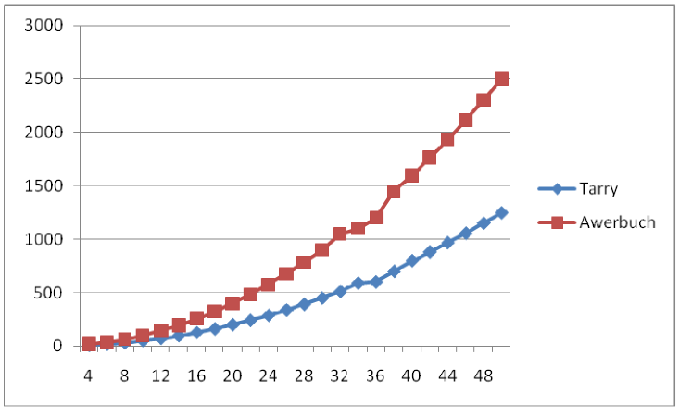

Fig 6. Comparison of message complexity at connection Probability of 50%

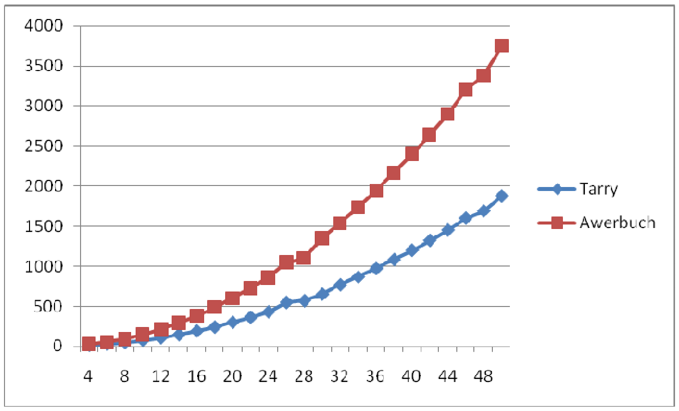

Fig 7. Comparison of message complexity at connection Probability of 75%

Fig 5 shows graph of message complexity at connection probability of 25%. Horizontal line represents number of nodes and vertical line represents message complexity. Result shows that during number of nodes between 4 and 14 both algorithms perform similar. And after number of nodes 14, Tarry's algorithm outperforms Awerbuch's algorithm. It can be observed that curve for Tarry's algorithm and curve for Awerbuch's algorithm is quadratic.

Fig 6 shows graph of message complexity at connection probability of 50%. Horizontal line represents number of nodes and vertical line represents message complexity. Result shows that during number of nodes between 4 and 12 both algorithms perform similar. And after number of nodes 12, Tarry's algorithm performs better than Awerbuch's algorithm. It can be again observed that curve for Tarry's algorithm and curve for Awerbuch's algorithm is quadratic.

Fig 7 shows graph of message complexity at connection probability of 75%. Horizontal line represents number of nodes and vertical line represents message complexity. Result shows that during number of nodes between 4 and 10 both algorithms perform similar. And after number of nodes 10, Tarry's algorithm performs better than Awerbuch's algorithm. It can be again observed that curve for Tarry's algorithm and curve for Awerbuch's algorithm is quadratic.

V. ANALYSIS OF RESULTS

In addition to analysis given in above section, it can be observed that for the graphs of time complexity, for large number of nodes, Awerbuch's algorithm proves very stable and it increases linearly, while for Tarry's algorithm, time complexity increases almost twice the speed of that of Awerbuch's algorithm. But for the small number of nodes, in fact Tarry's algorithm performs better than Awerbuch's algorithm. This was surprising, but it was reasonable. Similarly with graphs of message complexity, it can be observed that for small number of nodes, both algorithms perform almost similarly, but as number of nodes and connection probability increases, Tarry's algorithm performs better than Awerbuch's algorithm.

VI. CONCLUSION AND FUTURE WORK

This paper compared two important algorithms, Tarry's algorithm and Awerbuch's algorithm, based on their performance for time complexity and message complexity by varying number of nodes and connection probability. The conclusion is, for small number of nodes and lower connection probability, time complexity for Tarry's algorithm is better than Awerbuch's algorithm, so it is advisable to use Tarry's algorithm for small number of nodes and lower connection probability if aim is to maximize time complexity. But for large number of nodes and higher connection probability, time complexity of Awerbuch's algorithm is better than Tarry's algorithm. As far as message complexity is concerned, Tarry's algorithm is always better than Awerbuch's algorithm. Because of message overhead for "Ack" and "Vis" messages of Awerbuch's algorithm, it always performs poor than

Tarry's algorithm as far as message complexity is concerned. Also when the graphs become denser, both time and message complexity of Tarry's algorithm, and message complexity of Awerbuch's algorithm are quadratic to the number of nodes in the network, but the time complexity of Awerbuch's algorithm is still linear to the number of nodes in the network.

Future work is to make program more efficient. Instead of calculating random 0's and 1's for whole matrix, program can be modified so that it just calculate random 0's and 1's for upper half of matrix, which can be than used for lower half of matrix as well. Also data was collected on three trials, which is not enough in real time scenario. Future work would be to collect data on 15 or more trials. Also instead of varying number of nodes, number of edges can be varied to collect data. And finally rather than on simulation engine, data can be collected by running experiments on real time distributed systems.

REFERENCES

[1] http://mathworld.wolfram.com/AdjacencyMatrix.html
[2] http://deneb.cs.kent.edu/~mikhail/classes/aos/
[3] "Introduction to Distributed Algorithms" by Gerard Tel