Performance Comparison of Tarry and Awerbuch Algorithms

Najla Alam

***Abstract*— Tarry and Awerbuch are examples of widely used wave algorithms. These message passing schemes can be embedded in particular algorithms, where they appear as sub tasks viz. broadcasting, synchronization, and computing global functions. The definition of a wave algorithm is as follows; it exchanges a finite number of messages and then makes a decision, which depends causally on some event in each process. It is assumed that there is a special type of event called a decide event. Tarry is a traversal algorithm, a special class of wave algorithms in which all events of a wave are totally ordered by causality relation. I am presenting a performance comparison of Tarry and Awerbuch with respect to two characteristics, message complexity and time complexity. Message complexity is defined as the number of messages it takes the algorithm to carry out specified task Time complexity is measured by the number of messages in the longest chain of causally dependent event.**

I. Introduction

This paper presents performance analyses and comparisons of Wave algorithms, Tarry's algorithm and Awerbuch's algorithm. Details of how the experiment was carried out and what results were obtained, the conclusions drawn are discussed following a brief description of the algorithms. Two specific parameters are used to make performance comparisons; Message complexity and Time complexity. My results supported the expected values for Message and Time complexities obtained using a fixed set of parameter while varying two. First set of readings is taken for sparser graphs and the second for denser graphs. To study the difference in performances of the algorithms the network size is increased and readings tabulated at each point. The results obtained are plotted as, Message Complexity/Number of nodes and Time Complexity/Number of nodes.

The remainder of the paper is organized as follows. The next section gives brief descriptions of the subject algorithms. In Section 3 I provide an elaborate description of the experimental set up. In Section 4 I present the results obtained with references to graphs and tabular data. The succeeding Section discusses the overall inferences/conclusions drawn from the experiment conducted and the results obtained. The final Section contains possible areas of future work.

II. Algorithms

In Tarry's algorithm each process maintains an array of boolean values for its neighbors which is used to keep track of the neighbors to which it has sent token and a variable to store its *father*. 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 there is any such to which it has not sent before, 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 p has stored its *father* in the tree in the variable *father*.

Awerbuch's algorithm computes spanning trees with the additional property of being depth-first search trees. An important mechanism in Awerbuch's solution is that token is prevented from being transmitted though a frond edge. In the algorithm it is ensured that each process knows, at the moment when it must forward the token, which of its neighbors have been visited already. The process then chooses an unvisited neighbor, or sends the token to its father if no such neighbor exists. When process p is first visited by the token, p informs each neighbor r, except its father, of the visit by sending a vis message to r. The forwarding of the token is suspended until p has received an ack message from each neighbor. This ensures that each neighbor r of p knows, at the moment p forwards the token, that p has been visited. When, later, the token arrives at r, r will not forward the token to p, unless p is r's *father*.

III. Experimental Setup

The algorithms under discussion work on arbitrary topologies. For taking measurements of the performance parameters (message complexity and time complexity) arbitrarily connected graphs with random number of nodes were generated and the algorithms were executed on them. Each vertex represents a pair of x,y coordinates on a 10x10 geometric graph of real values. An edge/link exists between two vertices if the distance between them is less than one. This method results in multiple disconnected graphs components on the x,y plane. The largest graph component is selected to serve as an input to run the subject algorithms.

Two sets of readings were taken. Keeping a fixed value for vertex to edge ratio, the number of nodes were increased. To study the behavior of the performance parameters with varying density and size of the graph, five values were obtained by executing algorithms on five randomly generated graphs for particular size and density ranges. Each data point on the resultant plot represents an average of five values, delivering an increased accuracy of results. The size of the graph/number of nodes is determined by the formula,

$$noofnodes = density.\frac{100}{\pi}$$

To vary the size of the graph, the parameter density is varied from 1 to 10. To carry out performance analyses average of five values of message complexity/ time complexity for each density value are taken and plotted against the corresponding density values.

The second set of readings is taken using a similar set up and method with the following variation. In the generated graphs, edge is defined between two vertices if the distance between them is less than or equal to 1.5 resulting in denser graphs with higher vertex: edge ratio.

IV. Results

Figure 1 shows the simulation results and effect on Message complexity is observed as number of nodes are increased on sparsely connected graphs. The results show a comparison of Tarry and Awerbuch algorithms in term of Message complexity. For both algorithms an increase in the size of the graph results in higher values of message complexity, however, Awerbuch's performance is worse than Tarry's. There is a linear increase in values of Message complexity, therefore Awerbuch's worse performance remains a constant multiple of Tarry's function for Message complexity.

Fig. 1.

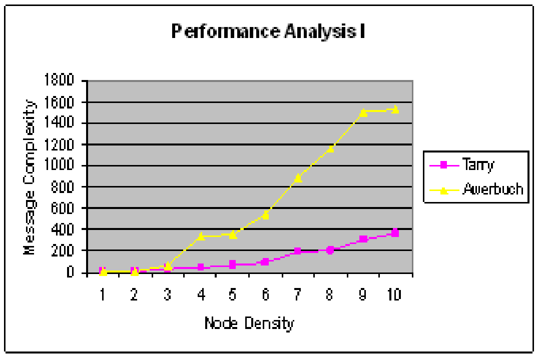
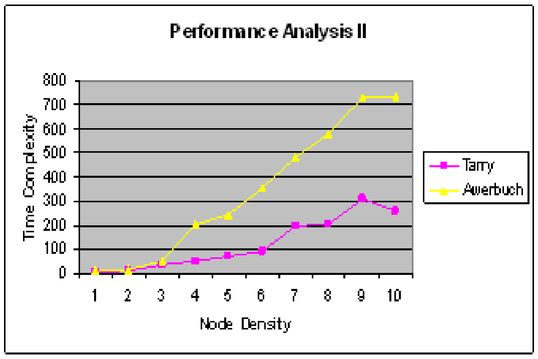


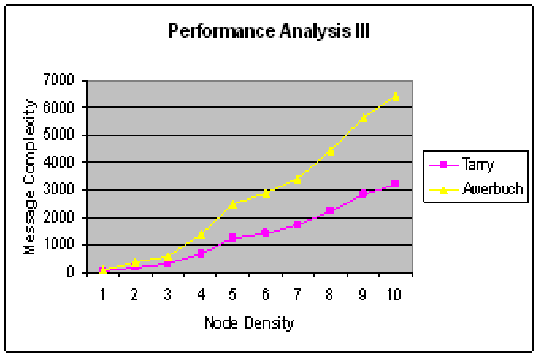
Figure 2 is obtained by computational values of Time complexity at each data point for both Tarry and Awerbuch algorithms. The effect on time complexity is seen as number of nodes are increased on sparsely connected graphs. The performance comparison results are as previously seen, Tarry's algorithm shows a better performance than Awerbuch's, with increasing values for Time complexities for both algorithms with an increasing in the graph sizes.

Fig. 2.



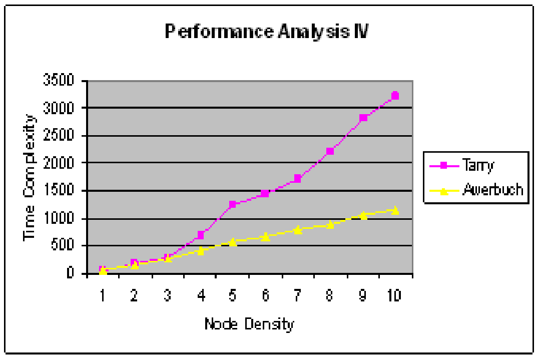
The following figure, Figure 3, shows results for similar nature of experimentation as the last two with the exception of using graphs of higer densities. The purpose is to observe whether or not this change effects the performance of Awerbuch. The Message complexities obtained with increasing number of nodes shows once again better performances for Tarry's than Awerbuch's with linear increments for both algorithms.

Fig. 3.



In the last set of results, Figure 4, we observe that Awerbuch's algorithm out performs Tarry's. The varying parameter is increasing number of nodes, as defined using the afore-mentioned formula, and values for Time complexity are recorded at each data point. Each data point represents an average of set of five reading taken at each graph size. Another difference made to measure Time complexity, which produced differing results from Figure 2 was the use of denser graphs (higher vertex to edge ratio).

Fig. 4.



V. Assumptions

Due to time constraints the values for message and time complexities in this experiment were calculated using the theoretical formulas.

In Tarry's algorithm because the token is sent at most once in each direction through each channel, it is passed at most 2E times before the algorithm terminates. Time complexity is equal to message complexity as Tarry's algorithm is a traversal algorithm.

TABLE I

THEORETICAL VALUES FOR PERFORMANCE COMPARISON

Algorithm	Message complexity	Time complexity
Tarry	2E	2E
Awerbuch	4N - 2	4E

In Awerbuch's algorithm each frond carries two vis messages and two ack messages. Each tree carries two token messages, one vis (sent from father to son), and one ack (from son to father). Consequently, 4E messages are exchanged. The token traverses serially each of the N-1 edges twice, which costs 2N-2 time units. At each node the token waits at most once, before it can be forwarded, for the exchange for vis / ack messages, which gives rise to a delay of at most two time units at each node.

VI. Conclusions and Future Work

For experiment done with less dense graphs, Tarry performs better than Awerbuch both in terms of Message complexity and Time complexity. As the number of nodes increase Awerbuch shows far greater increase in Time and Message complexities than Tarry. When the same experiment was run on denser graphs with increasing number of nodes, as seen before, Tarry showed better performance in terms of Message complexity whereas Awerbuch performed better with respect to Time complexity (Table I).

This experiment can be extended to include other algorithms such as the Distributed Depth-first search, Cidon's solution.

References

[1] Introduction to Distributed Algorithms," by Gerard Tel
[2] http://deneb.cs.kent.edu/ mikhail/classes/aos.f07/