A Novel Ant Algorithm for Solving the Minimum Broadcast Time Problem

Yehudit Hasson and Moshe Sipper

Department of Computer Science, Ben-Gurion University, Israel
{hassonj,sipper}@cs.bgu.ac.il, www.moshesipper.com

Abstract. The problem of Minimum Broadcast Time (MBT) seeks to find the minimal number of times steps required to broadcast a message through a communication network. We describe a novel ant algorithm for solving this NP-Complete problem and compare it to three other known algorithms, one of which is genetic. Through experimentation on randomly generated graphs we show that our algorithm finds the best solutions.

1 Introduction

Communication in networks—in its many guises and forms—is one of the prime areas of research and application today. The performance of a network is influenced by many parameters, one of the most important being the time required to send a message between two communication sites: the message delay. In this paper, we describe a technique to reduce the delay when sending an identical message from one computer (or set of computers) to all other computers in the network—a process known as *broadcasting*. Broadcasting can be used to send control messages throughout a network, or for cable television.

A communication network can be modeled as a connected, undirected graph, wherein the nodes represent computers and the edges represent the communication lines between them. A node communicates with another by transmitting a message, or making a *call*. Theoretically, there is no limitation on the amount of information that can be exchanged during a given call. A *round* is a series of calls carried out simultaneously, each round assumed to require one unit of time. The efficiency of a broadcast scheme is measured by the number of time units it takes to complete the broadcast.

The problem of Minimum Broadcast Time (MBT)—our focus in this paper—is defined as follows: Given a connected, undirected graph G = {V,E} and a subset of nodes, $V_0 \subseteq V$, which initially contain a given message; and given that at each time step every node can transmit the message to *one* other node that has not already received the message; then, find a transmission plan that minimizes the number of steps needed to execute the *broadcast* operation, namely, transmit the message to all nodes in the graph.

Formally, let $V_0 \subseteq V$ be a distinguished vertex (or set of vertices) that holds the message at step 0. A broadcast from V_0 is a sequence $V_0, E_1, V_1, E_2, \ldots, E_k, V_k$ such that for $1 \leq i \leq k$ the following constraints hold:

1. $V_i \subseteq V$, $E_i \subseteq E$, and $V_k = V$.
2. Every edge in E_i has exactly one end point in V_{i-1}.
3. No two edges in E_i have an endpoint in common.
4. $V_i = V_{i-1} \cup \{v : (u, v) \in E_i\}$.

The MBT problem is to find the minimal k needed to complete the broadcast.

In this paper we describe an ant algorithm for solving MBT. The next section delineates previous work on solving MBT, followed by Section 3, which describes ant algorithms. Section 4 presents our novel algorithm, and our results are detailed in Section 5. Finally, we present concluding remarks and future work in Section 6.

2 Previous Work

Since the MBT problem is NP-Complete [1], available solutions are heuristic in nature. These are reviewed in this section.

Scheuermann and Wu [2] represented the broadcast operation as finding at each step an optimal matching in a bipartite graph.

$G = \{V, E\}$ is called a *bipartite* graph if $V = X \cup Y$, $X \cap Y = 0$, and each edge $e \in E$ has one endpoint in X and one in Y. A *matching* in a bipartite graph is a subset $E_s \subseteq E$ such that no two edges in E_s have an endpoint in common. A *maximal matching* in such a graph is the largest set $E_s \subseteq E$.

Scheuermann and Wu developed an algorithm based on dynamic programming that builds a state tree, wherein each node represents different choices of maximal matching. The algorithm searches for the shortest path in the tree by applying backtracking with pruning techniques. Being exponential in the network size, this algorithm becomes inefficient for large networks (over a few dozen nodes).

Scheuermann and Wu [2] also developed a few heuristics based on greedy techniques. Each node in the graph is assigned a d-value, which may either be the node degree, the length of the shortest path to the farthest node from it, or the maximum between these two values. The d-value can be calculated for different variations of the graph:

- The original graph G.
- The subgraph $G - S$, where S denotes the subgraph formed by the nodes that have the message.
- The subgraph $(G - S) \oplus E_r$, where E_r denotes the subgraph of G formed by the edges with both ends in R (i.e., both ends are nodes without a message), and $\oplus$ is the xor operator.

At each time step, the algorithm searches for an optimal matching between the group of nodes that have the message and the group of those that do not.

Scheuermann and Wu presented two types of search mechanisms:

1. LWMM (Least Weight Maximum Matching), which searches for a maximum matching with minimum weight (the weight of an edge is the d-value of the target node).

2. AM (Approximate Matching), which attempts to find a matching between the target node with the highest d-value and the source node with the lowest vertex degree.

Hoelting, Schoenfeld, and Wainwright [3] developed a genetic algorithm-based (GA) heuristic to the problem. The algorithm begins with a random population of chromosomes, each one being a permutation of the graph nodes.

In order to calculate an individual's fitness, the chromosome is reversed and divided into two lists: the s-list, containing nodes with the message, and the r-list, containing nodes without the message. The algorithm traverses the lists from left to right, trying to match a node in the s-list with a node in the r-list (if more than one match exists, only the first one found is taken into account). The node that receives the message in the current step is added to the end of the s-list in the same order as in the original chromosome. The fitness is the number of steps needed to complete the broadcast.

The crossover operation takes two chromosomes and compares them according to a global precedence vector (GPV), which contains the nodes in the graph sorted according to their vertex degree in ascending order. The idea behind this operation is to force lower-degree vertices toward the front of the chromosome (left) and nodes with a higher degree toward the end of the chromosome (right). To decode the chromosome (in order to evaluate the individual) it is reversed and the message is transmitted to nodes with high degree first. The mutation operation switches between two alleles in the chromosome. The GA used in Section 5 is based on that of Hoelting, Schoenfeld, and Wainwright [3].

3 Ant Algorithms

Swarm Intelligence algorithms [4], which have received increased attention in recent years, are inspired by swarms in nature, e.g., of bees, ants, wasps, and birds. Such swarms give rise to intelligent behavior through the continual operation and interaction of thousands of autonomous, relatively simple members. No central controller is involved in the process.

When the swarm members do not communicate directly but rather do so indirectly—by using the environment as a "blackboard"—this is called stigmergy. Ants, for example, interact by depositing a chemical substance known as a pheromone along their trails. This substance can be sensed by fellow nest members, which tend to follow higher concentrations of pheromone, in turn increasing said concentration yet further. This is a form of indirect communication through the environment. Another example is nest building by wasps, which was simulated beautifully by Theraulaz and Bonabeau [5].

The Ant Colony System (ACS) algorithm was developed by Dorigo, Maniezzo, and Colorni [6], who applied it to the traveling salesman problem (TSP). Since then, much research has been carried out on so-called ant algorithms.

Das *et al.* [7] applied an ant colony approach to the Minimum Power Broadcast problem in wireless networks, a problem that shares certain similarities with

MBT. In this problem we assume a fixed N-node network with a specified source node, which must broadcast a message to all other nodes in the network. In a wireless network (as opposed to a wired one), a node can transmit a message to *any* other node in the network, and thus multiple nodes can be reached by a single transmission. The power required to transmit a message between two nodes depends on the Euclidean distance between them and a channel loss coefficient.

An ant under Das *et al.*'s scheme maintains a strategy for building a broadcast tree. The ant decides according to its strategy to which node the message should be sent. There are two types of ants (strategies):

1. Greedy "vision," preferring to send the message to the closest node.
2. Wide vision, choosing the next node according to roulette-wheel selection.

The algorithm executes several ants of both types in parallel. For each ant a broadcast tree is built, whereupon two functions are applied to improve its cost, the cost of a tree being the number of steps needed to execute the broadcast. The first function—multiple transmission removal (MTR)—removes multiple transmissions from a node, because the highest-powered transmission will also cover nodes which are reached by lower-power transmissions from that node. The second function—edge trimming (ET)—removes a transmission edge if no new node is reached by it.

The algorithm stores the best tree built so far and updates the pheromone on the tree edges after each ant's traversal and after choosing the best tree. In the end, the algorithm returns the tree with the best cost.

4 Solving MBT using Ants

Our algorithm employs nine types of ants, each of which dynamically builds a broadcast tree. An ant decides to which node to send the message in the broadcast tree, based on the node's local environment—i.e., its neighbors.

An ant maintains the following information:

- s-list: nodes with message.
- t-list: nodes without message.
- mutation: an ant has a small probability of being "olfaction-less," i.e., its decisions are not influenced by the trail left by other ants.

Tree-building by an ant is an iterative process, which starts with the s-list containing only the source node, and ends when the s-list contains all nodes in the graph. In each step, the ant builds an edge-list, with all the edges connecting nodes in the s-list (source nodes) with nodes in the t-list (target nodes). The ant chooses the next edge from the edge-list according to three parameters: 1) source value, 2) target value, and 3) trail. The chosen edge is added to the tree and any other edge in the edge-list that connects the corresponding source or target is removed from the list. The process of choosing edges from the edge-list continues until the edge-list is empty. Then all nodes that have reached this step are added to the s-list and removed from the t-list—and the step counter

is increased by one. The number of steps needed to build the tree is the tree's cost.

The value of an edge is evaluated according to its source and target nodes. In order to do so, each node i maintains information about its neighbors:

- nm-list$_i$: neighbors of node i with message.
- nwm-list$_i$: neighbors of node i without message.
- nwms-list$_i$: neighbors of node i without message that cannot receive message at this step (because they have no edge to a node currently holding message).

The source is assigned a higher value (S), the smaller the number of neighbors that can receive the message from it (nwm-list). Moreover, the value of the target (T) can be calculated by three different methods (depending on the ant type):

1. Number of neighbors that have a message and can send it to the target node at this step (nm-list). The target value is higher, the smaller the size of the nm-list.
2. Number of neighbors without a message (nwm-list).
3. Number of neighbors without a message that cannot receive the message at this step (nwms-list).

For methods 2 and 3, the target value is higher the larger the list size. An ant can choose an edge whose sum of source and target values is maximal, or it can choose first the source that has maximum value and then the target connected to that source with the highest target value—and vice versa. We thus have 9 types of ants, one ant for each version (Table 1): as can be seen, our use of nine types of ants arises from the nine heuristics attained by the various ways of combining source and target costs into an edge score. The use of nine ant types increases diversity, thus boosting the search process.

Table 1. The ants differ in the way they choose the next edge from edge-list. S and T are the respective source and target values for edge (i, j). The ant can choose an edge with max S+T, or first choose the source with max S and then the edge with max T connected to that source—and vice versa. In case more than one edge has the same value, selection is random. The precise formulas for calculating S and T are specified in Figure 1.

Ant 1 – S + T_1	Ant 2 – S + T_2	Ant 3 – S + T_3
Ant 4 – S $\rightarrow$ T_1	Ant 5 – S $\rightarrow$ T_2	Ant 6 – S $\rightarrow$ T_3
Ant 7 – T_1 $\rightarrow$ S	Ant 8 – T_2 $\rightarrow$ S	Ant 9 – T_3 $\rightarrow$ S

The algorithm stores the best tree built so far and updates the trail on the tree edges after running all ants for one cycle, according to the number of ants that choose the edge and the step at which they chose it. In the end, the algorithm returns the tree with the best cost. Figures 3 and 4 describe the pseudocode of our algorithm (the nomenclature is given in Figure 2).

$$\mathrm{S} = \tau_{ij}^{p} * (\frac{|V|}{|nm-list_i|+1})^{s} \qquad \mathrm{T}_2 = \tau_{ij}^{p} * (\frac{|nwm-list_j|*100+1}{|V|})^{t}$$
$$\mathrm{T}_1 = \tau_{ij}^{p} * (\frac{|V|}{|nm-list_j|+1})^{t} \qquad \mathrm{T}_3 = \tau_{ij}^{p} * (\frac{|nwms-list_j|*100+1}{|V|})^{t}$$

Fig. 1. Computing S and T values for edge (i, j). τ_{ij} is the pheromone amount on this edge; p is set to 0 when ant's mutation is false, otherwise it is set to 1; s and t were both set empirically to 1.

$T_k(t)$ – tree built by ant k at iteration t	$Y_k(t)$ – cost of said tree
T-LIST$^{best}(t)$ – list of best trees at iteration t	$Y^{best}(t)$ – cost of best tree at t
T^{best} – best tree found so far	Y^{best} – cost of best tree
$P_s(e)$ – e's node which is in s-list	$P_t(e)$ – e's node which is in t-list
$E_{st}(step)$ – list at broadcast $step$ containing all edges with source node in s-list and target node in t-list	
$E(step)$ – list containing all edges chosen at $step$	

Fig. 2. Nomenclature for pseudocodes in Figures 3 and 4.

```
1. t ← 0    // iteration index
2. τ_e ← τ_0, ∀e ∈ E   // pheromone amount on edge e
3. while (t < t_max)
     Y^best(t) ← ∞
     for k = 1 to NUMANTS // run ant k
         type = k mod ANT-TYPES // ant's type, where ANT-TYPES=9
         T_k(t) ← buildTreeByAnt(type)     // pseudocode given in Figure 4
         if Y_k(t) = Y^best(t) then T-LIST^best(t) ← T-LIST^best(t) ∪ T_k(t)
         if Y_k(t) < Y^best(t) then
             T-LIST^best(t) ← { T_k(t) }
             Y^best(t) ← Y_k(t)
     endfor
     for i = 1 to num elements in T-LIST^best(t)
         foreach e ∈ T-LIST_i^best(t) do    // edge in tree i of T-LIST^best(t)
             sumSteps ← sumSteps + 1/step    // step when edge reached
             numVisitors ← numVisitors + 1    // num ants visiting edge
         endforeach
     endfor
     if (t = 0) OR (Y^best(t) < Y^best)
         T^best ← T-LIST_0^best(t)
         Y^best ← Y^best(t)
     endif
     foreach e ∈ E do    // update pheromone on edge
         τ_e(t+1) ← τ_e(t)^ρ + (numVisitors/(NUMANTS+1))^α + (sumSteps/(NUMANTS+1))^β
         numVisitors ← 0
         sumSteps ← 0
     endforeach
     t ← t + 1
4. endwhile
5. return T^best and Y^best
```

Fig. 3. ANT-MBT-algorithm: main. The following values were set empirically: $\alpha = 20$, $\beta = 20$, $\rho = 0.1$, and $\tau_0 = 1$.

```
1. initialize ant:
    has-mutation ← true with probability MUTATION-RATIO
    T ← { }
    foreach node i∈V do
        nm-list_i←{}   // neighbors of node i with message
        nwm-list_i←{ Neighbors{i} }  // neighbors of i without message
        nwms-list_i←{ Neighbors{i} }  // neighbors of i in nwm-list that
                                      // can't receive message at this step
    endforeach
2. execute in step 0:
    step ← 0    // broadcast step
    s-list ← { sourceNode }    // nodes with message
    t-list ← { V - sourceNode }   // nodes without message
    nwms-list_sourceNode ← { }
    foreach neighbor i of sourceNode
        move sourceNode from nwm-list_i to nm-list_i
        remove sourceNode from nwms-list_i
    endforeach
3. while t-list is not empty
    step ← step + 1
    build E_st(step)
    foreach e ∈ E_st(step)
        foreach neighbor i of P_t(e)
            remove P_t(e) from nwms-list_i
        endforeach
    endforeach
    have-msg-list ← { i | i ∈ s-list and nwm-list_i != { } }
    while have-msg-list is not empty
        //choose the next edge in the broadcast tree according to
        //ant's type and has-mutation value
        foreach e ∈ E_st(step)
            compute S ant T values   // see Figure 1
        endforeach
        chosenEdge ← choose the next edge   // see Table 1
        E(step) ← E(step) ∪ chosenEdge
        remove P_s(chosenEdge) from have-msg-list
        remove every e ∈ E_st(step) with one endpoint
               equal to either P_s(chosenEdge) or P_t(chosenEdge)
        nwms-list_P_t(chosenEdge) ← { }
        foreach neighbor i of P_t(chosenEdge)
            move P_t(chosenEdge) from nwm-list_i to nm-list_i
            if nwm-list_i is empty and i ∈ have-msg-list
                remove i from have-msg-list
        endforeach
    endwhile
    T ← T ∪ E(step)
    update s-list and t-list
4. endwhile
5. return T
```

Fig. 4. ANT-MBT-algorithm: buildTreeByAnt(*type*). This function returns the broadcast tree built by the ant. MUTATION-RATIO was set empirically to 0.05.

5 Results

Given a connected undirected graph G = {V,E}, with $V_0 = \{u\}$ and $|V| = n$, we can conclude a number of things about the broadcast:

1. A broadcast from a vertex u defines a spanning tree rooted at u.
2. In each step the set of nodes that have received the message increases by at least one. Therefore, the time needed to execute a broadcast in G is a value in the set $\{\lceil \log_2\{n\} \rceil, \ldots, n-1\}$.

We tested the algorithms on graphs containing 15 to 250 nodes with low edge connectivity probabilities, running 10 randomly generated networks for each variation. Note that a low connectivity probability entails a harder problem, since in highly connected graphs the broadcast is easily achieved.

Both GA and Ant are population algorithms, each member in the population representing a candidate solution. We compared both algorithms using the same parameter values (population size, number of cycles, mutation rate), and under the same assumptions:

1. If an optimal solution (treeCost = $\log |V|$) is found, execution of the algorithm is stopped and the solution is returned.
2. The best solution in a cycle moves on to the next generation.
3. The algorithm is said to have converged to a solution if 85% of the population have held the same solution for the past 10 generations, and there is no better solution.

Table 2 shows our results. As can be seen our algorithm emerges as the winner. Though some results may seem only slightly better than the other algorithms, one should bear in mind that *every decrease of even a single broadcast step is significant and hard to attain.*

Table 2. A comparison of algorithm performance rates. For GA and ANT, population size = $2|V|$ (where $|V|$ is the network size), number of generations/cycles = 50, and mutation rate = 0.05. We ran the variants of LWMM and AM that were reported in [2] to produce the best results: 1) LWMM: d-value is the vertex degree, calculated in the subgraph $G - S$; 2) AM: d-value is the vertex degree, calculated in the subgraph $(G-S) \oplus E_r$. Results for AM and LWMM are per one run (algorithms are deterministic), and for ANT and GA averaged over 10 runs.

Network size	Edge Probability	AM	LWMM	GA	ANT
15	0.1	6.9	6.9	**6.3**	**6.3**
30	0.1	6.5	6.4	6.3	**5.9**
60	0.05	8.4	8.3	7.8	**7.7**
60	0.075	7.4	7.5	7.1	**6.6**
60	0.1	7.0	7.0	6.7	**6.4**
120	0.05	8.0	7.9	8.0	**7.3**
120	0.075	8.0	7.3	7.4	**7.0**
120	0.1	8.0	**7.0**	**7.0**	**7.0**
250	0.05	9.0	8.2	9.0	**8.0**
250	0.075	9.0	**8.0**	8.2	**8.0**
250	0.1	9.0	**8.0**	**8.0**	**8.0**

Figure 5 shows the effects of population size on performance of the GA and ANT algorithms. For the GA, increasing the number of the chromosomes improves performance, while for ANT the effect of increasing the ant population is less significant—the ANT algorithm finds better solutions even with a small number of ants.

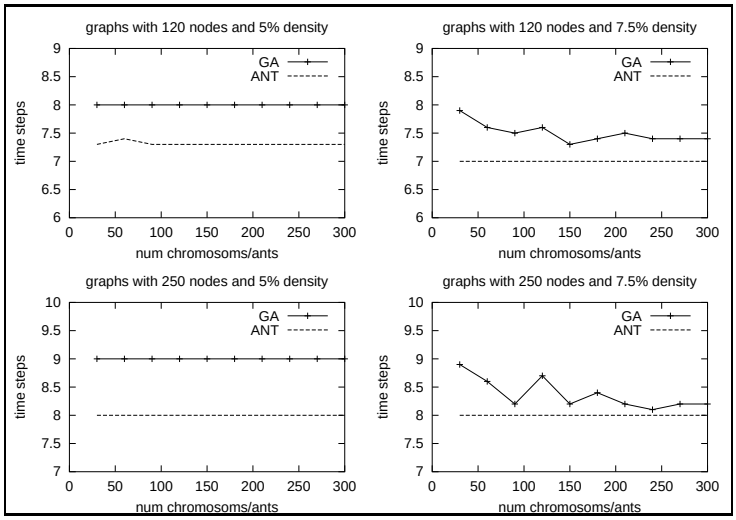

Fig. 5. Effect of population size on performance (shown for the hard problem range, i.e., low density—for higher densities the problem becomes easier).

Given G={V, E}, the time complexity of the four algorithms discussed herein is a function of several parameters: $|V|$ – number of nodes, $|E|$ – number of edges, $|N|$ – maximal degree of a node, $|K|$ – number of broadcast steps needed to reach a solution, $|C|$ – number of cycles/generations, and $|CH|/|ANTS|$ – number of chromosomes/ants. The time complexities are:

- AM: $O(|K||V|^2)$. This algorithm has proved to be the fastest of the four.
- LWMM: $O(|K||V|^3|E|)$.[1]
- GA: $O(|C||CH||K||V|^2)$.
- ANT: $O(|C||ANTS||K||V||E|)$.

As an example, if $|V|$=60, $|N|$= 5, and $|CH|/|ANTS|$=$2|V|$, then AM and LWMM take about a second to complete, the GA takes 0.2 seconds per generation, and ANT takes a second per cycle.

[1] This algorithm finds a least weight maximum matching in a bipartite graph according to the Ford-Fulkerson algorithm [8], i.e., it finds a least weight augmenting path and defines the symmetric difference of the path with the old matching to be the new matching. In the implementation we find an augmenting path with minimal last edge because the weights of the intermediate edges cancel each other out [2].

6 Concluding Remarks and Future Work

We presented a novel ant algorithm for solving the hard problem of Minimum Broadcast Time, and showed that for hard graph instances (namely, with low edge density), it surpasses three other state-of-the-art algorithms.

Our algorithm can be extended to other problems: each ant finds a solution based on some form of problem-specific knowledge implemented as a heuristic; it improves its solution by interacting with (the solutions found by) other ants.

Two variations of the MBT problem we intend to explore in the future are:

- The Gossip Problem. In this variation of MBT every node holds a message, which needs to be sent to all other nodes. A node can transmit to its neighbor all the messages it has received so far. The objective is to minimize the time for every node to receive all messages [9, 10].
- The Multicast Routing Problem, wherein the message is to be sent to a subset of nodes in the graph. Additional constraints can be added to the problem, e.g., minimizing the delay of sending a message from the source to a target, and minimizing the cost of sending the message to all target nodes [11].

References

1. Garey, M.R., Johnson, D.S.: Computers and Interactability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, San Francisco, CA (1979)
2. Scheuermann, P., Wu, G.: Heuristic algorithms for broadcasting in point-to-point computer networks. IEEE Transactions on Computers **C-33** (1984)
3. Hoelting, C.J., Schoenefeld, D.A., Wainwright, R.L.: A genetic algorithm for the minimum broadcast time problem using a global precedence vector. In: Proceedings of the 1996 ACM Symposium on Applied Computing, Philadelphia, PA, ACM Press (1996) 258–262
4. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. SFI Studies in the Sciences of Complexity. Oxford university Press (1999)
5. Theraulaz, G., Bonabeau, E.: Coordination in distributed building. Science **269** (1995) 686–688
6. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics—Part B **26** (1996) 1–13
7. Das, A.K., II, R.J.M., El-Sharkawi, M.A., Arabshahi, P., Gray, A.: The minimum power broadcast problem in wireless networks: An ant colony approach. In: Proceedings of IEEE CAS Workshop on Wireless Communications and Networking, Pasadena, CA (2002)
8. Ford, L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press, New Jersey (1962)
9. Baker, B., Shostak, R.: Gossips and telephones. Discrete Mathematics **2** (1972) 191–193
10. Hedetniemi, S.M., Hedetniemi, S.T., Liestman, A.L.: A survey of gossiping and broadcasting in communication networks. Networks **18** (1988) 320–349
11. Hakimi, S.L.: Steiner's problem in graphs and its implications. Networks **1** (1971) 113–133