Ant Colony Algorithms for Routing in Sensor Networks

Sanjoy Das[1], Gurdip Singh[2], Sandeep Pujar[2], Praveen Koduru[1]

[1]Electrical and Computer Engineering
Kansas State University
Manhattan, KS 66506

[2]Computing and Information Sciences
Kansas State University
Manhattan, KS 66506

Abstract: The ant systems optimization approach is a new method of solving combinatorial optimization problems. It was originally introduced as a metaheuristic approach for the well-known traveling salesman problem. But it was subsequently shown to be an equally effective algorithm for solving other optimization problems. In this paper, we present an ant colony algorithm for data-centric routing in sensor networks. In each pass of the proposed algorithm, ants are placed at the terminal nodes of the tree to be computed. They are then allowed to move towards one another, along the edges of the graph, until they merge into a single entity. In this process, the paths taken by the ants define a data distribution tree. Edges receive reinforcement in the form of pheromone deposits along the paths taken by the ant. Pheromones eventually accrue most along better edges. In addition to forward and backwork ants, we also use random ants whose purpose is to enable sharing of information pertaining to a node potential with neighboring sensors. Since ant algorithms perform computations solely through local interactions between ant-like agents by means of pheromones, they scale well for large-scale applications, and are particularly attractive for real world systems. The algorithm can easily be used in several practical applications.

1. Data-centric Routing in Sensor Networks

Sensor networks are networks of nodes capable of sensing and tracking in different contexts such as factory floors, transportation systems and hospitals. As these networks are deployed and sensor applications are developed, algorithms for control and management will be necessary. For example, consider a network of sensors deployed in a city to monitor traffic. Example usage of data from such a network involves queries such as "What is rate at which cars are passing a given intersection X?" or "What is the number of vehicles waiting at traffic light Y?" Moreover, different clients may be interested in knowing this information at different sampling rates. This, for instance, may require a *data-centric routing tree* in which data can be efficiently aggregated at intermediate points or may require a *minimum rate tree* in which data is sent at different rates over the edges to satisfy requirements of each destination [1].

The data-centric routing in sensor networks can be formalized as follows. Assume that the network is represented by a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. In Multi-source Single-destination Data-Centric routing (MSDC), there exist a set of source sensors and a single destination node, which are included in a subset T of V. For example, the destination may be interested in a particular phenomena and a set of sensors in the vicinity or related to this phenomena may subsequently send information to this sink). In this case, all sensors are assumed to be sending the same information. Hence, data can be aggregated at intermediate nodes (and duplicate data can be suppressed). The task is to construct an optimal tree that contains the source nodes and the destination such that the tree performs optimal aggregation. Since the source nodes may not be directly connection to the destination there may be nodes other than the source and destination that are part of the tree. However, these intermediate nodes receive and forward the information along the tree (that is, they do not consume or process the information other than maybe eliminating duplicates). This problem was first defined in [1] by Krishnamachari *et al*, which proposed and evaluated some sub-optimal schemes for generating aggregation trees (including a greedy incremental tree approach).

2. The Ant colony approach

Real ants, in spite of their extremely simple behavior, are known to trace out an optimal path from their nest to a food source and back, by means of cooperative activity. Communication between the ants is achieved through the deposition of *trail* pheromones. While negotiating a path to the destination, each ant deposits a trail along its path for other ants to follow. Each ant also tries to follow the trail left by previous ants, but does so in a highly stochastic manner, using the trail only as a rough indicator of the eventual trajectory. The trail that the numerous ants in the nest leave, also keep evaporating with time. Eventually, the pheromone level gets concentrated most along an optimal path from the source to the destination.

Ant colony algorithms are particularly suitable for large-scale distributed systems. Among the advantages they offer over traditional methods are (*i*) scalability, (*ii*) robustness, and (*iii*) suitability for dynamic environments [2]. These algorithms are highly scalable because all computations are solely in the form of local interactions alone, carried out in a purely distributed environment and without any centralized control. Robustness arises not only as ants, which are very simple agents, are less likely to fail, but also because an ant's failure will not seriously affect the performance of the algorithm. Ant algorithms have been shown to carry out computations very effectively in dynamic environments [3,4]. A number of advantages of distributed ant colony algorithms are listed in [5].

Algorithms based on this approach have rapidly gained popularity in network-based applications [6,7,8]. Amongst the earliest of such approaches were ABC and CAF, which assumed symmetric costs across links and were useful in telephone and packet-switching networks [3,6]. Another class of approaches, including Ant-net has begun to appear that consider the case of asymmetric costs [10,11]. Two types of ants are employed - forward ants and backward ants that travel in opposite directions. Forward ants originate from the source nodes in the network and migrate towards the destinations, collecting information concerning path costs between edges along the way. Backward ants take the information collected by the forward ants and retrace their paths, updating the local parameters at each node based on this information. Other similar techniques have begun to make their appearance [12,13,14,15].

Defining ant movements for problems that involve establishing an optimal path from one single source to a destination is accomplished relatively easily. Establishing ant movements for the TSP problem and some other permutation problems is also straightforward as each ant simply executes a cyclic path covering all nodes before it reaches its starting nodes. However having ants define a tree from a given graph is more complex and was studied recently [7,8,9], where the ant colony paradigm was applied to the Steiner tree problem, a problem of significance in many algorithms, which involves tree computations.

A distributed version of the algorithm for data-centric routing was proposed in [16]. The algorithm proposed in this paper improves upon the algorithm by adding a third class of ants to it, called random ants, and making appropriate changes in the algorithm. Tests have been performed on the algorithm to show its effectiveness.

3. Outline of the algorithm

This section describes the distributed ant colony algorithm. The input to this algorithm is again a weighted graph $G = (V, E)$ where each edge has a cost. The terminal nodes $T \in V$ have both the source as well as the destination sensors. The destination node is called d . A direct edge (i, j) between two sensors (nodes) i and j exists if the two sensors can communicate, in which case, the cost of the edge is the Euclidean distance between them, $dist(i, j)$.

The algorithm makes use of three kinds of ants, forward ants that travel from the sources to the destination, exploring new paths and gathering information, backward ants, that travel back to the sources from the destination to update the information in each sensor node as they move, and random ants who dissipate information collected at the nodes among other neighboring nodes. Ants traveling in such a manner have been explored previously for establishing routes in computer networks. In the present method, a tree is obtained from the graph G when the paths traced by forward ants as they merge into each other or reach the destination. This tree defines the paths along which data is to be transmitted from the sources to the destination. Because the forward ants move from the sources to the destination, d, they can also carry packets of data.

When the forward ants move across the network, the movement is governed by two factors, the pheromone trails that are deposited along the edges, and the potential of the nodes. The potential of a node is a heuristic that provides an estimate of how far an ant will have to travel from any the node to either reach the destination d or to aggregate data with another node. In other words, this node potential of each node is a measure of the distance of the node to the tree.

Each sensor node i contains two tables, the pheromone trails, τ_i and the node potential, Ψ_i, each of size equal to $\|nbd_i\|$, the number of sensors in the neighborhood of i (assumed to be known *a priori*). Therefore, $\tau_{i,j}$, the jth entry of the pheromone table is the pheromone concentration from the trail leading from i to a neighboring sensor j. Similarly, each $\Psi_{i,j}$ is the node potential of sensor j stored at node i.

The pheromone trails are all initialized to a sufficiently high value τ_0 to make the algorithm exploratory, and the initial node potentials are based on heuristic estimates. Each sensor node also maintains a variable tag_i, which is initialized to zero, and contains information about how many ants have visited the node.

Forward Ant Movement: The total number of forward ants is equal to the number of source sensors, and each ant begins its path from a source sensor in $T-\{d\}$. Each such forward ant m maintains the tabu list $T^{(m)}$ of nodes already visited, as well as a variable $pCost^{(m)}$, the path cost, that indicates the partial cost contributed by the ant's own path to the Steiner tree. The list $T^{(m)}$ is initialized to the source sensor where the ant is located, while $pCost^{(m)}$ is set to zero.

The probability of an ant moving from any current node i to another one j in nbd_i is given by,

$$p_{i,j} = \frac{(\tau_{i,j})^{\alpha} \big/ (\Psi_{i,j})^{\beta}}{\sum\limits_{k \in nbd_i} (\tau_{i,k})^{\alpha} \big/ (\Psi_{i,k})^{\beta}} \tag{1}$$

where α and β are the two constant exponents associated with the algorithm. In order to prevent the formation of cycles, nodes in $T^{(m)}$ that are already visited are excluded. The next location for ant m is chosen based on this probability, the new location j is pushed into $T^{(m)}$, and tag_j is examined. If tag_j is zero, indicating that location j is previously unvisited, the cost of the path $i \to j$, $dist(i,j)$ is added to $pCost^{(m)}$. A non-zero value indicates that another ant has already visited the node, and therefore the cost of the path $i \to j$ is already incorporated in another ant's $pCost$. Under these circumstances, the forward ant m has already merged into an already existing path. It simply follows the previous ant's path to the destination node.

The destination node, d contains a variable *cost*, the total cost of the tree path from the sources to *d*. When a forward ant m enters the destination node, *d* it increments *cost* by an amount $pCost^{(m)}$. In the present version of the online algorithm, it is assumed that the total number of source nodes is known by the destination at the beginning of the computation. When all forward ants have arrived at the destination, backward ants are generated at the destination. There is a one-to-one correspondence between the forward and the backward ants, and a backward ant, also indexed as *m* acquires the list $T^{(m)}$ of the corresponding forward ant *m*. Algorithm 1 outlines the algorithm for controlling the movement of forward ants through the network.

Algorithm 1: Forward-Ant

```
//forward ant $m$ move from a source $src^{(m)} \in T-\{d\}$ to $d$
forwardAnt($m$, $src^{(m)}$, $d$)
begin
        $f = FALSE$
        $i = src^{(m)}$
        $T^{(m)} = \{\}$
        push($T^{(m)}, i$)
        $cost^{(m)} = 0$
        while $i \neq d$
                if $f = FALSE$
                        $\forall j \in nbd_i - T^{(m)}$, compute
                        $p_{i,j} = \frac{(\tau_{i,j})^{\alpha} / (\Psi_{i,j})^{\beta}}{\sum_{k \in nbd_i} (\tau_{i,k})^{\alpha} / (\Psi_{i,k})^{\beta}}$
                        select $j$ based on probability
                        $next_i = j$
                else
                        $j = next_i$
                endif
                move to $j$
                if $f = FALSE$
                        $pCost^{(m)} = pCost^{(m)} + dist(i,j)$
                endif
                if ($tag^{(j)} > 0$)
                        $f = TRUE$
                endif
```

```
            $tag^{(j)} = tag^{(j)} + 1$
            $i = j$
        endwhile
end
```

Backward Ant Movement: Each time a backward ant moves, it pops $T^{(m)}$ to obtain the next destination. The backward ants carry a copy of the destination variable *cost.* This information is used to update the pheromones. A backward ant that moves from node *j* to node *i* updates the pheromones of node *i* in the following manner,

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \rho\frac{Q}{cost}. \tag{2}$$

In the above equation, the quantities ρ and Q are the usual parameters of the ant colony optimization. The second term in the right of the above equation depends inversely on the total cost. If the path from *i* to *j* eventually leads to a low cost Steiner tree, then the pheromone concentration is incremented by a higher amount. All other entries, $k \neq j$, of the pheromone table are subject to evaporation as before,

$$\tau_{i,k} = (1-\rho)\tau_{i,k}. \tag{3}$$

Updating the tables of node potentials is somewhat more complex. A node's potential is considered low if it is either close to the destination, or brings a forward ant closer to the rest of the Steiner tree. In order to detect the cost of a node to *d*, each backward ant *m* maintains a variable $pCost^{(m)}$, the path cost, similar to a forward moving one, initially zero at the destination d, that gets incremented by an amount equal to $dist(i,j)$ whenever a backward ant moves from *j* to *i*. When a backward ant is in any node, $pCost^{(m)}$ is the cost of the path joining the node to *d.* In order to compute the cost of joining a node to another route, i.e. only a branch of the Steiner tree, another variable $rCost^{(m)}$, called the route cost, is used by backward ants, which is updated in the same manner. However, $rCost^{(m)}$ is reset to zero each time a backward ant detects a split in a path leading to more than one branch of the Steiner tree. A split, leading to another branch is detected by examining the tag_i variable of a node *i.* If the previous node of the backward ant was *j*, then node *i* is a separate branch if tag_i < tag_j. A backward ant m leaving a node j decrements the tag_j variable such that at the end of the first set of backward ants travel back to the sources in *S*, these variables are reset to zero for future ants. The updating rule for $\Psi_{i,j}$ is,

$$\Psi_{i,j} = \gamma \times rCost^{(m)} + \lambda \times pCost^{(m)} \tag{4}$$

where γ and λ are two additional parameters. Since from equation (8), the forward ants are biased towards preferring nodes with lower potentials, the second term in the above equation allows them to hop closer to the destination d while the first term lets the forward moving ants prefer towards nodes that have already been included in the route of another ant. This updating is carried out only if the node potential gets lowered. The objective here is to identify the least expensive way to get to the destination, if a previous ant has found a lower potential for a node then updating it as in equation (4) is not applied.

Algorithm 2 outlines the describes the movement of backward ants through the network.

Algorithm: Backward-Ant

```
//backward ant $m$ move from destination $d$ to source $src^{(m)}$
Backward Ant($m$ , $d$ , $src^{(m)}$)
begin
        $pCost^{(m)} = 0$
        $rCost^{(m)} = 0$
        while $i \neq src^{(m)}$
                j = pop($T^{(m)}$)
                move to $j$
                if ($tag_i \neq tag_j$)
                        $rCost^{(m)} = 0;$
                else
                        $rCost^{(m)} = rCost^{(m)} + dist(i, j)$
                endif
                $pCost^{(m)} = pCost^{(m)} + dist(i, j)$
                $T_{j,i} = (1-\rho)T_{j,i} + \rho \frac{Q}{cost}$
                $\forall l, T_{j,l} = (1-\rho)T_{j,l}$
                if $\Psi_{i,j} > \gamma \times rCost^{(m)} + \lambda \times pCost^{(m)}$
                        $\Psi_{i,j} = \gamma \times rCost^{(m)} + \lambda \times pCost^{(m)}$
                endif
                $i = j$
        endwhile
end
```

For simplicity, we have not provided an outline of the overall algorithm. It is clear however that the task of the destination is to gather information from each arriving forward ant, to compute the total cost of the tree. The backward ants are released only after all of the forward ants arrive at the destination. Likewise, each source sensor can only send forth a new forward ant only after the arrival of the backward ant in a previous iteration. It is clear that in our present version, even in the distributed environment, the sensors operate with a certain degree of synchrony. At any instance, the network is filled with either forward traveling ants, or with backward ones, but not both. The random ants, on the other hand, are free to move about in an asynchronous manner.

Random Ant Movement: The information collected by the forward ants is used by the backward ants to update the local information stored at the nodes. However backward ants are restricted to follow the paths of the forward ants only. The purpose of the random ants is to share this information pertaining to a node potential with neighboring sensors.

The movement of random ants is not driven by heuristics. In order to move, these ants simply select a location drawn at random from the neighborhood list of their current node. When a random ant leaves any sensor i, it records the minimum node potential at that sensor as $min_k(\Psi_{k,i})$. When it hops from one sensor to another node j, the node potential of i recorded there is updated as,

$$\Psi_{i,j} = min\left(\Psi_{i,j}, dist(j,i) + min_k(\Psi_{k,i})\right) \tag{5}$$

The algorithm provided below depicts the movement of random ants.

Algorithm: Random-Ant

```
//random ant $m$
Random Ant($m$, $i$)
begin
        compute $min_k(\Psi_{i,k})$
        select $j$ from $nbd_i$
        move to $j$
        $\Psi_{i,j} = min\left(\Psi_{i,j}, dist(j,i) + min_k(\Psi_{k,i})\right)$
        $j = i$
end
```

4. Simulation Results

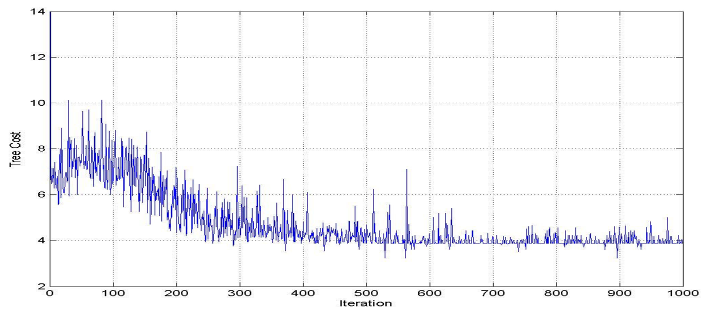
The algorithm was evaluated using discrete event simulation with Java as the programming language. The inputs to the simulation program are the following three files:

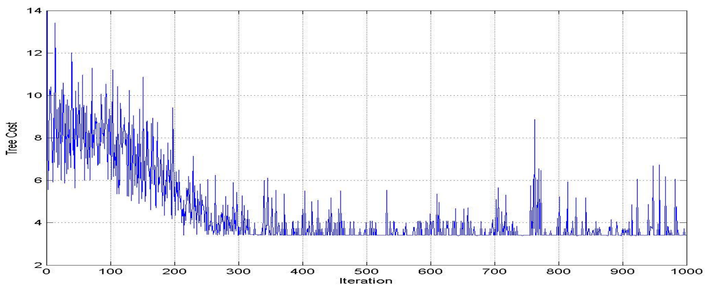
1. *Node.txt* contains N, the number of nodes, node number for the destination node, and the node number of the sources.
2. *Coordinate.txt* contains the coordinates for the nodes. This file is generated by randomly dispersing nodes over a certain range.
3. *Graph.txt* contains the egde costs.

The simulation program is an event-driven program where the program is initialized with events to start the movement of the forward ants at the source node and the random ants at randomly selected nodes. The program is subsequently driven by the events occurring in the algorithm. The simulation program simulates the local nature of the algorithm (that is, only local information is available to the ants and any data structure used are only for the purpose of collecting performance results). Initially, the algorithm was simulated for small sized graphs to enable manual checking of the results to ensure correctness of the program. Subsequently, the algorithm was simulated for larger graphs with the total number of nodes, N = 50, 100 and 200.

The nodes in the simulation were placed randomly using uniform distribution across a 1 X 1 sized area. Instead of creating a fully connected graph, nodes were connected to each other probabilistically, with the probability of connection being inversely proportional to the Eucilidean distance between the nodes. For nodes that were connected, the cost of the edge was simply equal to the distance between them. Exactly 20% of the nodes were selected as destination nodes. The trail evaporation rate ρ was set to 0.1, and the parameters γ and λ were set to 5 and 2 respectively.

The results for a 50-node and a 200-node network are shown in Figure 1 and Figure 2 respectively. The experiments were carried out with different number of random ants. For example, with 5% random ants, random ants are started at 5% of the nodes. As can be seen, as the number of random ants is increased, trees with lower costs are obtained. For the 50-node network, the tree cost converges to 3.87, 3.41 and 3.05 for 0%, 5% and 10% random ants respectively. For the 200-node network, the tree cost converges to 9.6, 8.57 and 7.7 for 0%, 5% and 10% random ants respectively. With random ants, we find that during the initial iterations, the tree cost varies significantly (in fact, some of the initial tree costs may be higher). This is due to the fact that random ants allow more paths to be explored, which may sometimes lead to higher cost trees. However, these paths are not taken in subsequent iterations.





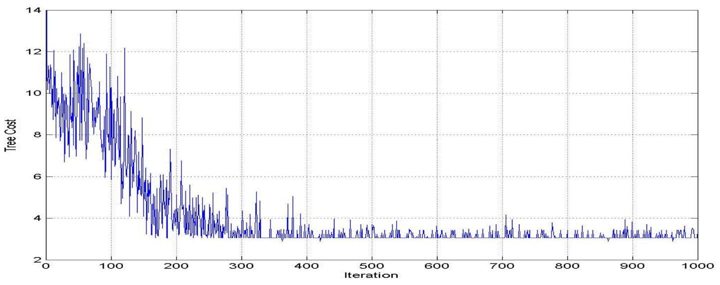
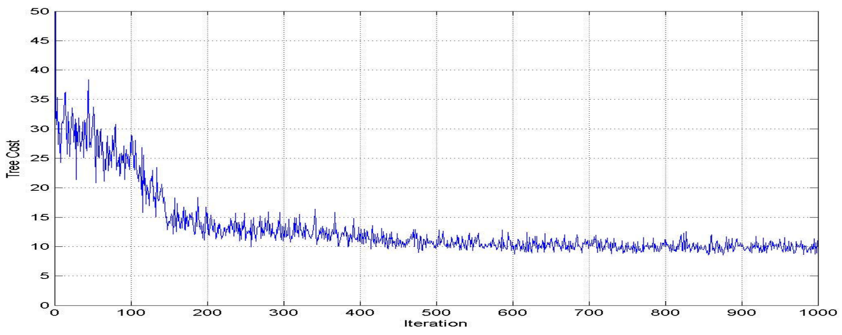


Figure 1: Simulation results for a 50 node network

5. Conclusion

In this paper, we proposed an application of ant colony algorithms to the problem of data centric routing in sensor networks. The problem involves establishing paths from multiple sources in a sensor network to one or more destinations. When only a single destination is involved, the set of optimal paths amount to a minimum Steiner tree, which is an NP-complete problem. We presented an online algorithm for data centric routing. The algorithm uses forward ants, backward ants and random ants. The purpose of the random ants is to spread node potential information among neighboring nodes. We show that this enables ants to explore new lower cost paths,



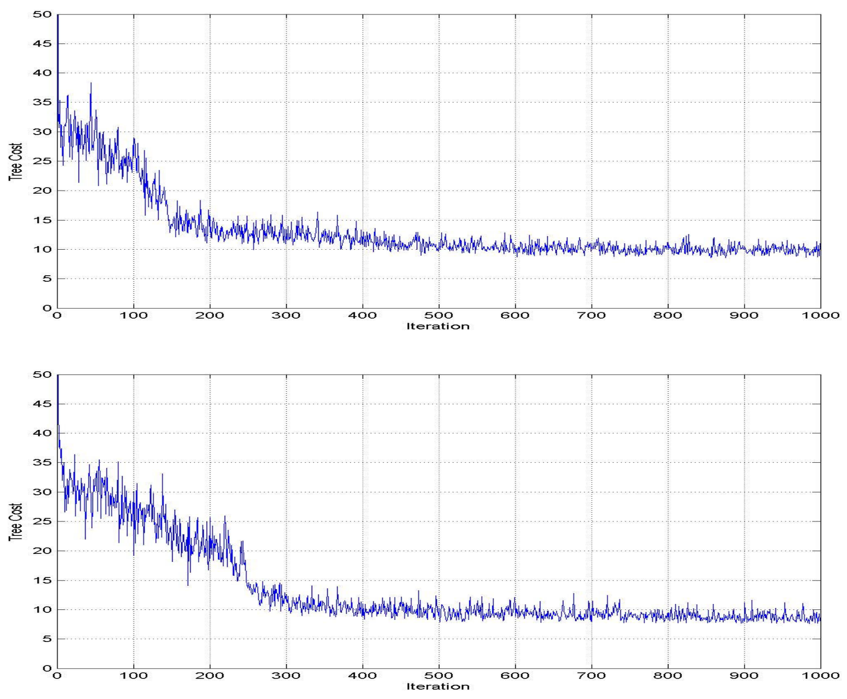


Figure 2: Simulation results for a 200-node network

resulting in more efficient distribution trees. The performance of the algorithm was evaluated using discrete event simulation.

References

1. B. Krishnamachari, D. Estrin, S. Wicker, "Modeling Data-centric routing in Wireless Sensor Networks", *Proceedings*, *IEEE INFOCOM*, 2002.
2. K. Lerman, A. Galstyan, "A General Methodology for Mathematical Analysis of Multiagent Systems", *Technical Report* ISI-TR-529, University of California, Information Sciences Institute, 2001.
3. D. P. Subramaniam, P. Druschel and J. Chen, "Ants and Reinforcement Learning: A Case Study in Routing in Dynamic Networks", *Proceedings, International Joint Conference on Artificial Intelligence,* 832-838, Palo Alto, CA 1997.
4. H. Matsuo, K. Mori, "Accelerated Ants Routing in Dynamic Networks", *2nd International Conference on Software Engineering,* Artificial Intelligence, Networking and Parallel Distributed Computing, Nogoya, Japan 2001.
5. Kassabalidis, M.A. El-Sharkawi, R.J. Marks II, P. Arabshahi, A. Gray, "Swarm Intelligence for Routing in Communication Networks", *IEEE Globecom'01,* San Antonio, Texas, 2001.
6. R. Schoonderwoerd, O. Holland, J. Bruten, L. Rothkrantz, "Ant-based Load Balancing in Telecommunications Networks", *Adaptive Behavior,* **5**(2), pp. 169-207, 1997.
7. S. Das, S. V. Gosavi, W. H. Hsu, S. A. Vaze, "An Ant Colony Approach for the Steiner Tree Problem", *Genetic and Evolutionary Computation Conference*, New York, 2002.

8. S. Das, I. Mohanty, D. Z. Yang, "An Ant Colony Algorithm for Multicast Routing in Communication Networks", *Proceedings, 6th International Joint Conference on Information Sciences*, Durham, North Carolina, 2002.
9. S. Das, G. Singh, S. Gosavi, S.Pujar, "Ant Colony Algorithms for Data-Centric Routing in Sensor Networks", *Proceedings, Joint Conference on Information Sciences*, Durham, North Carolina, 2003.
10. G. Di Caro, M. Dorigo, "Ant-Net: Distributed Stigmergetic Control for Communication Networks", Journal *for Artificial Intelligence Research*, 9:317-365, 1998.
11. G. Di Caro, M. Dorigo, "Ant Colonies for Adaptive Routing in Packet Switching Communication Networks", *Proceedings, Fifth International Conference on Parallel Solving for Nature,* K. A. Hawick, H. A. James (Eds.), 261-272, Springer-Verlag, 1998.
12. T. White, "Swarm Intelligence and Problem Solving in Telecommunications", *Canadian Artificial Intelligence Magazines,* Spring 1997.
13. T. While, B. Pagurek, "Towards Multi-swarm Problem solving in Networks", *Proceedings, Third International Conference on Multi-Agent Systems,*333-340, 1998.
14. S. Lipperts, B. Kreller, "Mobile Agents in Telecommunication Networks – A Simulative Approach to Load Balancing", *Proceedings, Fifth International Conference on Information Sciences, Analysis and Synthesis,* 1999.
15. K. Oida, M. Sekido, "Agent Based Routing for QoS Guarantees", *Proceedings, IEEE International Conferences on Systems, Man and Cybernetics,* 833-838, 1999.
16. G. Singh, S. Das, S. Pujar, S. Gosavi, "Ant Colony Algorithms for Steiner Trees: An application to Routing in Sensor Networks", Book Chapter in *Recent Developments in Biologically Inspired Computing, Eds. L. Nunes de Castro and F. Zuben, IGI press,* 2004.
17. A. Colorni, M. Dorigo, F. Maffioli, V. Maniezzo, G. Righini, M. Trubian, "Heuristics from Nature for Hard Combinatorial Problems", *International Transactions in Operational Research*, **3**(1), pp. 1-21, 1996.
18. M. Dorigo, L.M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", *IEEE Transactions on Evolutionary Computation*, **1**(1), pp. 53-66, 1997.
19. M. Dorigo, L.M. Gambardella, "Ant Colonies for the Traveling Salesman Problem", *BioSystems,* **43**, pp: 73-81, 1997.
20. M. Dorigo, G. Di Caro, "The Ant Colony Optimization Meta-Heuristic", in D. Corne, M. Dorigo and F. Glover (editors), *New Ideas in Optimization*, McGraw-Hill, pp. 11-32, 1999.
21. M. Dorigo, G. Di Caro, L. M. Gambardella, "Ant Algorithms for Discrete Optimization", *Artificial Life,* **5**(2), pp. 137-172, 1999.
22. A. Kapsalis, V. J. Rayward-Smith, G. D. Smith, "Solving the Graphical Steiner Tree Problem Using Genetic Algorithms", *Journal of the Operations Research Society*, 44(4): 397-406, 1993.