ELEKTROTEHNIŠKI VESTNIK 79(4): 153-156, 2012
ERK 2012 CONFERENCE ISSUE

Simulation Model for Ant-Based Control Algorithm in Wireless Mesh Networks

Erik Pertovt, Kemal Alič, Aleš Švigelj, Mihael Mohorčič

Institut "Jožef Stefan", Jamova cesta 39, 1000 Ljubljana, Slovenija
E-pošta: erik.pertovt@ijs.si

Abstract

In this paper, we present implementation of the ant-based control algorithm (ABC) for load balancing in connection-oriented routing in the wireless mesh network (WMN) using the OPNET Modeler simulation tool. We investigate the time required for the initialization phase of the ABC algorithm within which the shortest path is found as well as network topologies with a different number of nodes and different number of neighbours per node. The resuts show that the algorithm initialization time increases with the number of nodes and decreases with the number of neighbours per node.

1 Introduction

In communication networks, heuristic/agents-based routing algorithms can be used to implement the adaptive routing functionalities capable of making the routing aware of dynamic changes of the network rather than being oblivious to them [1]. In this context, several routing algorithms have been developed inspired by the behaviour of the ant colonies and most of them are summarized in [2, 3]. Ants are social insects that collaboratively perform complex tasks, which can not be performed by one particular ant itself. In simulations, ants/agents are produced in the form of packets, which are used for network optimization and management.

In this paper, we present the implementation of the main reference ant optimization algorithm for connection-oriented networks, named Ant-based control (ABC) [1, 4, 5], in OPNET Modeler [6] simulation environment with some additional functionalities that can be used as an option. The algorithm is implemented in our network coding simulation model, presented in [7, 8]. We demonstrate the performance of the simulation model through the investigation of the initialization phase of the algorithm in wireless mesh network (WMN), which represents an underlying study for our work on routing being aware of network coding opportunities in WMNs. In the initialization time, the algorithm has to find, as quick as possible, the shortest paths in the sense of the minimum hop count from all nodes to all nodes, as Dijkstra optimal routing algorithm does, which is used as a reference. Several simulation runs are performed with different network topologies to evaluate the performance of ABC algorithm.

2 Ant Algorithm

In [1, 4, 5], the ABC algorithm was introduced as the first ant optimization algorithm applied to a routing problem. It was used to balance the network traffic of telephone calls in a telecommunication network. The purpose of ants in the ABC algorithm is to find paths with little hops and nodes with spare capacity.The paths are then used as routes to establish new connections.

2.1 Algorithm description

Ants are generated and lauched from all the nodes in the network at every simulation time step of the simulation, synchronously. The destination of each ant is randomly selected. Ants select their next node based on the pheromone tables located in the nodes. In pheromone tables, pheromone values represent probabilities with which ants select the next node. Higher the probability is, higher the chance of the node selection.

When an ant traverses a link between nodes A and B in the direction from A to B, it "lays" on it pheromone, which will be used by ants moving in the opposite direction of the pheromone laying ant, i.e. from node B to A. Pheromone laying is applied in the algorithm as ants updating the pheromone tables. When the ant from source node S travelling from node A to the next node B, reaches node B, it updates that part of the pheromone table of node B that concerns ants with destination S travelling from node B to A. The updating is done in such a way that the probability of choosing node A as the next node by ants with destination S is increased, while the probabilities of choosing other possible next nodes from node B is decreased. The increase and decrease of pheromone values (applied in the form of probabilities) is done according to equations in [1, 4, 5]. In the equations, pheromone decay (which corresponds to the pheromone evaporation in reality) is applied as a function of ant hops representing the ant age. In correspondence to this, ants which traversed many hops, will deposit less pheromone resulting in lower increase of corresponding probabilities. It means that paths with more hops will be selected with lower probability as the routes for connections.

For considering the degree of node congestion, ants are virtually delayed at nodes as a function of the node spare capacity. Lower the percentage of spare capacity, greater the delay at the node and later the update of the pheromone table by ants, which has for the consequence that the node will be selected with lower probability as part of the routes for connections.

Routing tables are built based on the pheromone (ant-decision) tables. For a destination D, next node B is selected at node A if pheromone value for B in pheromone table in node A is greater than for other neighbours of A concerning destination D.

In one type of the algorithm, there is also a minority group of ants which select their next node randomly to prevent blocking problem by extreme node congestion or node failure, and to keep suddenly appearing better routes to be discovered more rapidly.

2.2 Initialisation

Before network operation begins and connection requests are placed in the network, routes from each source node to each destination node in the network have to be discovered by ants in the initialization phase.

At the beginning, all the pheromone tables are initialized with equal probabilities for neighbour nodes, meaning that each first ant in each node will select its next node among the neighbours with equal probability. Ants are not virtually delayed, as there is still no connection traffic in the network. The only influence on the routing are thus pheromone tables in the nodes. As pheromone tables are updated by ants as described above, the shortest paths (i.e. the minimum hop count) from sources to destinations will be discovered. As shown in [4], this scheme really tends to produce the shortest paths and gives a good enough result within the fixed period of the initialization phase time compared to a deterministic optimal algorithm as Dijkstra.

3 Implementation of the Ant Algorithm in OPNET

The ABC algorithm was implemented as ANT module in OPNET Modeler [6] simulation environment [7, 8] with all the algorithm functionalities. The state diagram of the implementation is presented in Figure 1. In addition, different types of the algorithm were introduced, which are desribed in Section 3.1.

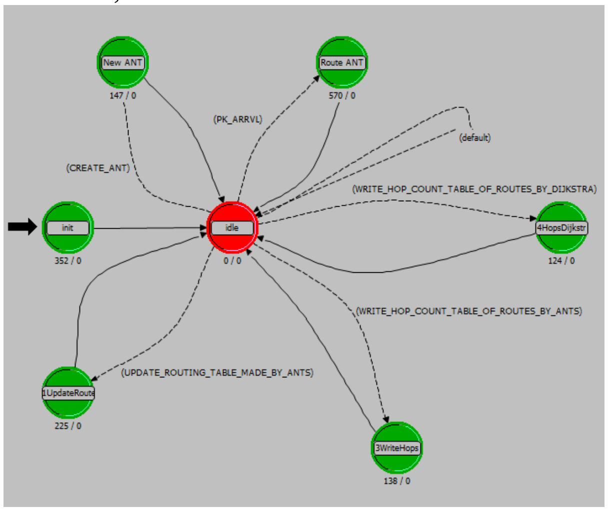

Figure 1. The state diagram of ABC OPNET Implementation.

In the ANT module, the red state represents the idle state, where the module waits for one of the interrupts. There are six main transitional states: (i) init state takes care of the algorithm parameters initialisation at the start of the simulation run; (ii) New ANT state creates and lauches new ants; (iii) Route ANT state receives ants coming from neighbouring nodes, updates pheromone tables based on the received ants pheromone information, virtually delays ants based on the percentage of the node's spare capacity, and selects the next node for ants; (iv) Update Route state updates routing tables based on pheromone tables every time pheromone tables are updated; (v) Write Hops state computes the instantaneous shortest paths from sources to destinations based on routing tables created by ants every time step during the algorithm initialisation phase; (vi) Hops Dijkstra state compares the shortest paths (i.e. optimal paths) found by Dijkstra to the corresponding shortest paths found by ants during the algorithm initialisation phase.

3.1 Propagation delay and ants dying

In our simulation, the ant movement from node to node is not determined by the simulation time steps. The time an ant is sent from a node to its neighbour is determined by the time the ant is waiting in the node's queue. In addition, we add to the present ABC algorithm some other functionalities as optional representing different types of the original algorithm.

In the first type of the ABC algorithm, we consider that an ant dies if it has performed a specific number of hops on the path. It means that ants which do not find their destination within a predefined number of hops are eliminated from the network. These ants are inefficient and do not really contribute to finding the shortest paths. All other functionalities are preserved from the ABC algorithm, presented above. We denote this type of the ABC algorithm as ABCdie.

In the second type of the ABC algorithm, besides all the ABC algorithm functionalities, we consider also propagation delay on the links between the nodes. This type of ABC algorithm considers the real network scenario case, where ants require the propagation time to travel on wireless links. We denote this type as ABCprop. The main property ABCprop introduces is that shorter links are preferable than longer links. On longer links, packets require more time to propagate through wireless medium and, therefore, there is more chance that they get lost, and vice versa.

The third type of the ABC algorithm comprises all the ABC algorithm functionalities and all additional functionalities introduced in ABCdie and ABCprop. It is denoted as ABCdie&prop.

4 Performance Evaluation of the Initialization Phase

We conduct several simulation runs of ABC algorithm in OPNET simulation model of WMN, presented in [7, 8]. In this section, we present selected parameters in simulations and simulation results. In particular, we are interested in the time required by the initialization phase of the ABC algorithm to find the shortest paths which correspond to the shortest paths found by Dijkstra in the number of hops on the paths. In [1, 4, 5], the time required for the initialization is assumed to be fixed. In ABC paper [4], authors only test if the initialization phase of the algorithm finds paths close to the minimum number of hops as Dijkstra does. Moreover, they performed simulation runs for a synchronized wired telecommunication network while, in our case, simulated networks are configured to present nonsynchronized WMNs.

4.1 Simulation parameters

In this paper, only the performance of the initialization part of the ABC algorithm is investigated, thus only the simulation parameters relevant for the initialization phase of the algorithm are discussed in the following.

We performed several simulation runs and investigate the required initialization time of ABC algorithm to find paths with the minimum number of hops in dependence of: (i) different number of nodes and different number of neighbours per node, and (ii) different types of the algorithm used during the simulation runs.

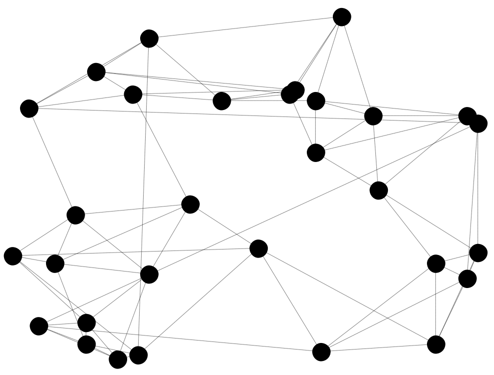

Figure 2. Network topology with 30 nodes and 5 neighbours per node.

We assume that all wireless network nodes are the same and have identical configuration, representing homogeneous network. Each node is given a random location within a given area. Node locations remain the same in all simulation scenarios. Neighbour selection is mainly based on the node positions. For the simulation purposes all the links are symmetrical 1Mbit/1Mbit and are ideal, meaning that no packets get lost during transmissions. In Figure 2, a network topology with 30 nodes each having 5 neighbours is presented. All other generated networks look similar. Wireless connections established between neighbours, which are represented in the network topology as wireless links, are graphically presented with dashed lines between nodes.

Table 1. Parameters of simulation runs.

Task	Type	Parameter	Attrib.
Ant creation	all	Time period	0.01 s
Ant pheromone laying config.	all	d	0.08
		c	0.005
Ants dying	ABCdie ABCdie&prop	10-30 nodes	15 hops
		40-70 nodes	20 hops

In Table 1, the used parameters for the corresponding task and type of ABC algorithm are summarised. The time period of ants creation in nodes is presented in seconds; this time has to be increased after the initialization phase to reduce the network overhead. This parameter is used in all types of the algorithm. Parameters for updating ants routing tables are presented without units and are also used in all types of the algorithm. The parameters affect equations, presented in [1, 4, 5], for calculating the amount of pheromone laying on a specific link. The number of hops required that an ant dies is presented in hops for networks with different number of nodes and is used only in ABCdie and ABCdie&prop.

4.2 Simulation results

In the following, the time required to find the shortest paths, in terms of the minimum number of hops, by ants is investigated. Four types of ABC algorithm implementations are investigated: (i) ABC, (ii) ABCprop, (iii) ABCdie, and (iv) ABCdie&prop. The results of the initialization time are presented for networks with 10, 20, 30, 40, 50, 60, and 70 nodes with 4, 5, 6, 7, and 8 neighbours per node. All the results are presented in Table 2. The time is presented in seconds.

Table 2. Times of ABC initialization phase.

Case	Neigh.	Nodes						
		10	*20*	*30*	*40*	*50*	*60*	*70*
ABC	*4*	0.8	18.4	29.5	54	77	323	352
	5	1.4	10.3	20.8	24	258	249	312
	6	1.2	16.5	30	50	62	149	142
	7	0.5	6.2	9.4	23	45	73	133
	8	0.8	9	9.9	24	38	68	167
ABC die	*4*	1.2	16.4	33.1	70	91	143	309
	5	1.1	6.8	19.1	48	70	123	196
	6	0.5	11.1	18.8	47	81	126	129
	7	0.5	7	14.7	32	38	98	180
	8	0.3	7.9	8.4	31	62	80	110
ABC prop	*4*	1.3	10.6	54	53	144	294	462
	5	2.9	11.5	15.9	56	201	113	136
	6	0.5	10.1	20.9	59	94	91	130
	7	0.5	5.5	13.6	32	55	90	105
	8	0.4	5.9	9.2	25	66	89	121
ABC die& prop	*4*	1.2	10.6	56.8	36	143	148	390
	5	1.1	8.2	30.2	34	54	127	237
	6	0.5	12.4	15.3	76	60	132	166
	7	0.3	4.8	11	27	57	85	110
	8	1.3	7.6	7.1	21	40	87	157

The initialization time in dependency of the number of nodes in the network for different number of neighbours per node for the ABC algorithm is depicted in Figure 3. The higher the number of nodes in the network, the greater the initialization time (as expected), and the lower the number of node neighbours, the greater the initialization time. The latter is due to the fact that in networks with less node neighbours the diameter of the network and average hop count to other nodes are greater. Thus, ants need to travel more time to achieve the desired results. Similar conclusions can be drawn for other types of ABC.

However, there are also cases where the initialization time for the network with more nodes is smaller than for the network with fewer nodes. Similar, the initialization time for the network with fewer neighbours per node can be in some cases smaller than for the network with more neighbours per node. This is because of the reinforcement learning algorithm used by

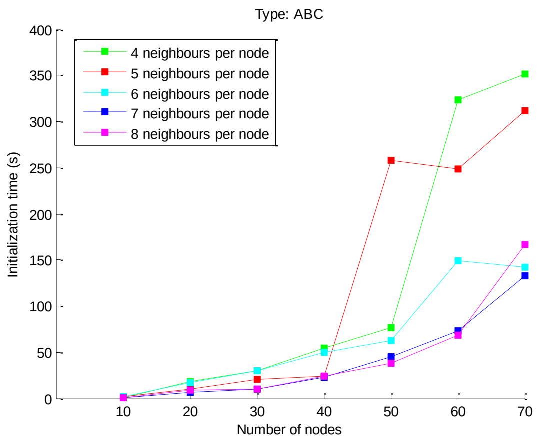

Figure 3. Initilization time in dependency of the number of nodes in the network for different number of neighbours per node for ABC algorithm.

ants, which comprises also the random part of pheromone laying. An ant selects the next hop (i.e. one of the neighbours of the node at which the ant is located at the moment) based on the probabilities in the pheromone table; higher the probability value of the neighbour, higher the chance to be selected as ant's next hop. But this also means that the node's neighbours with lower values in the pheromone table can be also selected as next hop, only that with lower probability. Thus, this randomness which is introduced through the probabilities is the cause of deviation in some cases from the expected behaviour.

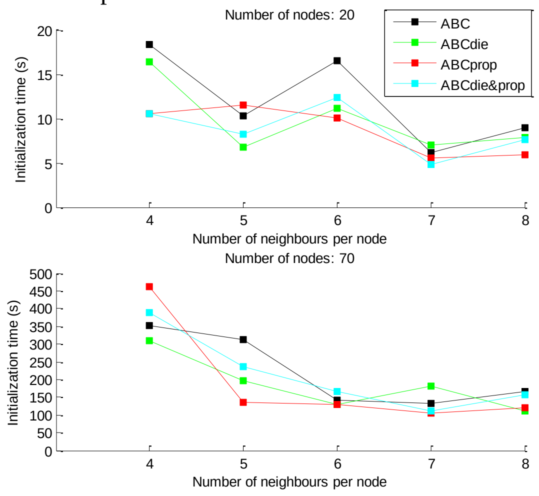

Figure 4. Initialization time in dependency of the number of neighbours per node for different types of ABC algorithm with 20 and 70 nodes in the network.

Similar as above, initialization time in dependency of the number of neighbours per node in the network for all four types of ABC algorithms for the network with 20 and 70 nodes is investigated in Figure 4. Please note different time scales in both graphs. It can be seen that times change when ABCprop is used compared with ABC. In the first case, ants require the propagation time to travel from a node to a node, while in the second case it is assumed that the time for travelling on links between nodes is zero. There is no tendency of the simulation time to be greater or lower in one of the cases. When ABCdie algorithm is used, times tend to be smaller compared with ABC, as the ants which are not efficient and do not find their destinations after 15 or 20 hops are eliminated from the network. However, there are also cases when ABCdie does not perform better. Again, this is because of some randomness of the reinforcement learning algorithm. In some cases, it is better to leave all ants in the network to learn but it is difficult to predict which cases are these.

5 Conclusion and future work

In this paper, we have described the implementation of the ABC [1, 4, 5] algorithm in the OPNET Modeler [6] simulation environment. We have implemented the algorithm in our network coding simulation model [7, 8] for WMNs. The time of the initialization phase required to find the shortest paths by ants, in terms of the minimum number of hops, is investigated for different types of ABC algorithm. The required initialization time increases with the number of nodes in the network and decreases with the number of neighbours per node, as expected. However, the additional functionalities of the ABCdie and the ABCprop tend to decrease the initialization time, which is favourable for routing.

As a further work, we will adapt the ABC algorithm for adaptive and dynamic routing that is aware of network coding opportunities in WMNs.

Literature

[1] R. Schoonderwoerd, "Collective intelligence for network control," M.S.c Thesis, Delft University of Technology, May 1996.

[2] M. Dorigo, G. Di Caro, L. M. Gambardella, "Ant Algorithms for Discrete Optimization," Artificial Life, 5(2), pp. 137-172, 1999.

[3] M. Farooq, G. A. Di Caro, "Routing Protocols for Next Generation Networks Inspired by Collective Behaviors of Insect Societies: An Overview," Swarm Intelligence (Natural Computing Series), Springer, pp. 101-160, 2008.

[4] R. Schoonderwoerd, O. Holland, J. Bruten, "Ant-Like Agents for Load Balancing in Telecommunication Networks," in Proceedings of the First Int. Conf. on Autonomous Agents, pp. 209-216, ACM Press, 1997.

[5] R. Schoonderwoerd, O. Holland, J. Bruten, L. Rothkrantz, "Ant-based load balancing in telecommunications networks," Adaptive Behavior, 5(2), pp. 169–207, 1996.

[6] OPNET web page, available at http://www.opnet.com/, retrieved July 2012.

[7] K. Alic, E. Pertovt, and A. Svigelj, "Simulation Environment for Network Coding," IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies 2011 (AEECT 2011) Amman, Jordan, 2011.

[8] K. Alic, E. Pertovt, A. Svigelj, "Network coding simulation model in OPNET Modeler," OPNETWORK 2012, Washnigton, NW, USA, August 2012.