An Intelligent Global Flooding Algorithm in Wireless Ad Hoc Networks Based on Distributed Learning Automata

J. Akbari Torkestani
Department of Computer Engineering, Islamic Azad University, Science and Research Branch, Tehran, Iran
j-akbari@iau-arak.ac.ir

M. R. Meybodi
Computer Engineering and IT Department, Amirkabir University of Technology, Tehran, Iran
Institute for Studies in Theoretical Physics and Mathematics (IPM), School of Computer Science, Tehran, Iran
mmeybodi@aut.ac.ir

ABSTRACT

In wireless ad hoc networks, due to the dynamic topology changes, multi hop communications and strict resource limitations, routing becomes the most challenging issue, and broadcasting is a common approach which is used to alleviate the routing problem. Global flooding is a straightforward broadcasting method which is used in almost all existing topology-based routing protocols and suffers from the notorious broadcast storm problem. The connected dominating set (CDS) formation is a promising approach for reducing the broadcast routing overhead in which the messages are forwarded along the virtual backbone induced by the CDS. In this paper, we propose an intelligent backbone formation algorithm based on distributed learning automata (DLA) in which a near optimal solution to the minimum CDS problem is found. Sending along this virtual backbone alleviates the broadcast storm problem as the number of hosts responsible for broadcast routing is reduced to the number of hosts in backbone. The proposed algorithm can be also used in multicast routing protocols, where the only multicast group members need to be dominated by the CDS. In this paper, the worst case running time and message complexity of the proposed backbone formation algorithm to find a $1/(1-\varepsilon)$ optimal size backbone are computed. It is shown that by a proper choice of the learning rate of the proposed algorithm, a trade-off between the running time and message complexity of algorithm with the backbone size can be made. The simulation results show that the proposed algorithm significantly outperforms the existing CDS-based backbone formation algorithms in terms of the network backbone size, and its message overhead is only slightly more than the least cost algorithm.

KEYWORD Wireless ad hoc networks, backbone formation, broadcast storm problem, connected dominating set, distributed learning automata

1. INTRODUCTION

A wireless ad hoc network is a multi hop wireless communication network supporting a collection of mobile hosts. There is no fixed infrastructure and no central administration and the mobile hosts can form a temporary network infrastructure in an ad hoc fashion. Two hosts can directly communicate when they are within transmission range of each other, and indirectly through relaying by the intermediate hosts. In an ad hoc network, each host assumes the role of a router and relays the packets toward the final destinations, if a source can not directly send the packets to a final destination due to the limitation of the radio transmission range. Since the wireless ad hoc networks exhibit severe resource constraints such as the bandwidth and power limitations, network topology changes, and the lack of the fixed infrastructures and consequently,

centralized administrations, to achieve good performance in ad hoc networks, the load on the hosts should be kept as low as possible[3, 22].

Dynamic network topology changes, resource constraints (e.g., bandwidth and power limitations) and lack of the fixed infrastructures and centralized administrations result in the optimal routing becomes the most challenging problem in wireless ad hoc networks [2-3]. Broadcasting is a common approach which is used to alleviate the routing problem [25], and a straightforward broadcasting by the global flooding, is a method usually employed by almost all existing topology-based routing protocols. The major drawback of the global flooding method is the notorious broadcast storm problem, in which the flooding may result in excessive redundancy, contention, and collision. The CDS formation is a promising approach for constructing a virtual network backbone which significantly alleviates the broadcast routing overhead by reducing the rebroadcasts (or redundant broadcasts), as the number of hosts responsible for routing is reduced to the number of hosts in the backbone [7, 33].

A wireless ad hoc network can be modeled as a unit disk graph $G = (V, E)$, where the nodes represent the individual hosts, and an edge connects two nodes if the corresponding hosts are within the transmission range of each other. A dominating set (DS) of a graph $G = (V, E)$ is a host subset $S \subseteq V$, such that every host $v \subseteq V$ is either in S or adjacent to a host of S. Each host in dominating set S is called a dominator host, otherwise it is called a dominatee host. A host of S is said to dominate itself and all adjacent hosts. Finding the dominating set is a well-known approach, proposed for clustering the wireless ad hoc networks [1, 3-4, 7-8, 32]. A minimum DS (MDS) is a DS with the minimum cardinality. A dominating set is also an independent dominating set, if no two hosts in the set are adjacent. A connected dominating set (CDS) S of a given graph G is a dominating set whose induced sub graph, denoted $< S >$, is connected, and a minimum CDS (MCDS) is a CDS with the minimum cardinality. A MCDS forms a virtual backbone [9-15, 23-26] in the graph by which the routing overhead can be significantly reduced, where the number of hosts responsible for the route discovery and data transmission can be reduced to the number of hosts in the backbone. A CDS is said to be Steiner CDS, if only a specified subset of the hosts has to be dominated by the CDS. Finding a Steiner CDS in the network, which dominates an arbitrary subset of hosts, forms a virtual backbone along which the multicast packets can be sent to the given hosts as the multicast members, and so can be effectively used in multicast routing protocols.

The closed neighborhood $N_G[v]$ of host v in graph G consists of the hosts adjacent to v as well as host v itself. The closed neighborhood $N_G[S]$ of the set S is the union $\bigcup_{v \in S} N_G[v]$. A dominating set S is a weakly connected dominating set (WCDS) of a graph G, if the graph $< S >_W = (N[S], E \cap (N[S] \times S))$ is a connected sub graph of G. In other words, the weakly induced sub graph $< S >_W$ contains the hosts of S, their neighbors, and all edges with at least one endpoint in S. The MDS and MCDS problems have been shown to be NP-Hard [5, 6], and even for a unit disk graph, the problem of finding a MCDS is still NP-Hard [6]. A sample UDG and one of its virtual backbones induced by the CDS have been shown in Figures 1.A and 1.B, respectively.
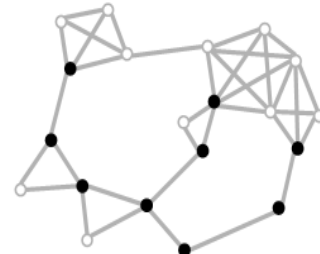

Figure 1.A. A sample unit disk graph

Figure 1.B. The virtual backbone induced by the CDS

In this paper, a DLA-based backbone formation algorithm is proposed to alleviate the notorious broadcast storm problem by reducing the rebroadcasts. The aim of the proposed algorithm is to form a virtual backbone for the wireless ad hoc networks by finding a near optimal solution to the minimum CDS problem. To implement this approach, a network of the learning automata, isomorphic to the unit disk

graph of the ad hoc network, is initially formed by equipping each host to a learning automaton. Then, at each stage, the learning automata randomly choose one of their actions in such a way that a solution to the CDS problem can be found. The constructed CDS is evaluated by the random environment, and the action probability vectors of the learning automata are then updated depending on the response received from the environment. Finally, the learning automata, in an iterative process, converge to a common policy that constructs a minimum size virtual backbone for the network graph. We compute the worst case running time and message complexity of the proposed backbone formation algorithm to find a $1/(1-\varepsilon)$ optimal size backbone and show that by a proper choice of the learning rate of the proposed algorithm, a trade-off between the computational and message complexities of algorithm with the backbone size can be made. We compare the results of our proposed algorithm with those of the best-known CDS-based backbone formation algorithms and the results show that our algorithm always outperforms the others in terms of the backbone (CDS) size, and its message overhead is only slightly more than the least cost algorithm.

The rest of the paper is organized as follows. The next section reviews the related work. Section 3 describes the learning automata, distributed learning automata and variable action set learning automata. In Section 4, the proposed DLA-based backbone formation algorithm is presented. In Section 5, the worst case running time and message overhead of the proposed algorithm are computed. The performance of the proposed algorithm is evaluated through the simulation experiments and comparison with the best-known CDS-based algorithms in Section 6, and Section 7 concludes the paper.

2. RELATED WORK

Finding the minimum CDS problem in an arbitrary graph is a NP-Hard problem [5, 6], and so a host of approximation algorithms have been proposed to find a near optimal solution to this problem in a reasonable time. Guha and Khuller [27] proposed two centralized greedy heuristic algorithms with bounded performance guarantees for connected dominating set formation. In the first algorithm, the connected dominating set is grown from one node outward, and in the second algorithm, a weakly connected dominating set is constructed, and then the intermediate nodes are selected to create a CDS. Guha and Khuller [27] also proposed an approximation algorithm to solve the Steiner CDS problem, in which only a specified subset of vertices has to be dominated by a CDS. Butenko et al.[13] also proposed a prune-based heuristic algorithm for constructing the small connected dominating sets. In this algorithm, the connected dominating set is initialized to the vertex set of the graph and each node is then examined to determine whether it should be removed or retained. If removing a given node disconnects the induced sub graph of the connected dominating set, then it is retained and otherwise removed. The algorithm proposed by Wu and Li [30] first finds a CDS and then prunes certain redundant nodes from the CDS. Alzoubi et al. [9, 28] proposed two distributed heuristic algorithms for constructing CDS in a wireless ad hoc network. These algorithms first employ the distributed leader election algorithm [28] to construct a rooted spanning tree from the original network topology. Then, an iterative labelling strategy is used to classify the nodes in the tree to be either dominator or dominatee, based on their ranks. The first heuristic uses the ID-based approach for ranking the nodes, and the second heuristic uses the level-based approach. Alzoubi et al. [31] proposed another distributed algorithm for approximating the minimum CDS in a wireless ad hoc network. The proposed algorithm first constructs a maximum independent set (MIS), and then forms the CDS by adding more intermediate nodes into the MIS. Li et al. [14] proposed a greedy MIS-based algorithm for constructing CDS in wireless networks. The first step of the proposed algorithm focuses on the forming a MIS of the network topology graph. At the second step, a greedy approximation is employed for finding a Steiner tree with the minimum number of Steiner nodes to interconnect the nodes in the MIS. Xie et al. [15] proposed a novel distributed approximation algorithm to construct the MCDS in wireless sensor networks in which the network is modeled as a hierarchical graph. In this algorithm, at each level of the hierarchical graph, a selected set of nodes is served as the message hubs (to route the messages) for the other nodes in the next level of the hierarchical graph. The proposed algorithm uses a competition-based strategy to select the nodes at each level. Wu and Dai [29] proposed a prune-based heuristic algorithm for constructing the virtual backbone of the wireless ad hoc networks by finding the dominator nodes. In this method, the connected dominating set is initialized to the set of all vertices having two unconnected neighbors. Then, the connected dominating set is modified by removing the nodes whose neighbors are also the neighbors of the other nodes in the CDS.

Learning automata have been found to be useful in systems where incomplete information about the environment, in which those systems operate, exists. Learning automata are also proved to perform well in

dynamic environment of the wireless, ad hoc and sensor networks. Haleem and Chandramouli [34] used learning automata to address a cross-layer design for joint user scheduling and adaptive rate control for downlink wireless transmission. The proposed method tends to ensure that user defined rate requests are satisfied by the right combination of transmission schedules and rate selections. Nicopolitidis et al. [35] proposed a bit rate control mechanism based on learning automata for broadcasting data items in wireless networks. A learning automaton is used in the server which learns the demand of wireless clients for each data item. As a result of this learning, the server is able to transmit more demanded data items by the network more frequently. The same authors [36] proposed a learning automata based polling protocol for wireless LANs in which the access point uses a learning automaton to assign to each station a portion of the bandwidth proportional to the station's need. Nicopolitidis et al. [37] proposed an ad hoc learning automata-based protocol for wireless LANs. In this protocol, the data transmission permission is granted by means of the learning automata to the stations. The proposed protocol is capable of operating efficiently under the bursty traffic conditions. They showed the proposed protocol outperforms TDMA in all cases. Nicopolitidis et al. [38] also proposed a carrier-sense-assisted adaptive learning MAC protocol for wireless LANs, in bursty traffic wireless networks with unreliable channel feedback. The self adaptive proposed protocol utilizes carrier sensing in order to reduce the collisions that are caused by different decisions at the various mobile stations due to the unreliable channel feedback. Ravana and Morthy [39] proposed Learning-TCP, a novel learning automata based reliable transport protocol for wireless networks, which efficiently adjusts the congestion window size and thus reduces the packet losses.

Learning automata are also used in cellular radio networks [40-43]. Beigy and Meybodi [40] proposed two learning automata based decentralized dynamic guard channel algorithms for cellular mobile networks. The proposed algorithms use learning automata to adjust the number of guard channels to be assigned to the cells in the network. In [40] they also introduced a new model for nonstationary environments under which the proposed algorithms work. They [41] also introduced a multi-threshold guard channel policy, and proposed a prioritized channel assignment algorithm for multi-cells cellular networks to minimize the probability of blocking the calls with lowest level of QoS subject to constraints on the blocking probabilities of other calls. The same authors [42] also proposed an adaptive and autonomous call admission algorithm for cellular mobile networks in which a new continuous action set learning automaton is used to minimize the blocking probability of the new calls.

Recently, a few attempts are made for applying learning automata to the sensor networks [44- 46]. In [44], Gholipour and Meybodi proposed a learning automata-based mobicast routing protocol for wireless sensor networks. The proposed protocol uses learning automata to adaptively determine the location and the shape of the forwarding zone in such a way that the number of activated nodes in each forwarding hop along the routing path is almost the same. A clustering algorithm using cellular learning automata was proposed by Esnaashari and Meybodi in [45]. In this clustering algorithm, each node is equipped by a learning automaton. Learning automaton of each node in cooperation with the learning automata of the neighboring nodes determines the role of the node to be a cluster head or a cluster member. Same authors in [46] proposed a learning automata-based scheduling solution to the dynamic point coverage problem. In the proposed scheduling algorithm, the learning automaton of each node learns the sleep duration of that node based on the movement pattern of a single moving target point in the network.

3. Learning Automata, Distributed Learning Automata, and Variable Action Set Learning Automata

3.1. Learning Automata

A learning automaton [16-21] is an adaptive decision-making unit that improves its performance by learning how to choose the optimal action from a finite set of allowed actions through repeated interactions with a random environment. The action is chosen at random based on a probability distribution kept over the action set and at each instant the given action is served as the input to the random environment. The environment responds the taken action in turn with a reinforcement signal. The action probability vector is updated based on the reinforcement feedback from the environment. The objective of a learning automaton is to find the optimal action from the action set so that the average penalty received from the environment is minimized [16].

The environment can be described by a triple $E \equiv \{\underline{\alpha}, \underline{\beta}, \underline{c}\}$, where $\underline{\alpha} \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ represents the finite set of inputs (actions), $\underline{\beta} \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ denotes the set of

values can be taken by the reinforcement signal, and $\underline{c} = \{ f(\alpha) \mid \alpha \in \underline{\alpha} \}$ denotes the set of the penalty probabilities (probability distributions over $\underline{\beta}$), $f(\alpha)$ is called *penalty function*. If the penalty probabilities are constant, the random environment is said to be a stationary random environment, and if they vary with time, the environment is called a non stationary environment. The environments depending on the nature of the reinforcement signal β can be classified into P-model, Q-model and S-model. The environments in which the reinforcement signal can only take two binary values 0 and 1 are referred to as P-model environments. Another class of the environment allows a finite number of the values in the interval [0, 1] can be taken by the reinforcement signal. Such an environment is referred to as Q-model environment. In S-model environments, the reinforcement signal lies in the interval $[a, b]$. The relationship between the learning automaton and its random environment has been shown in Figure 2.
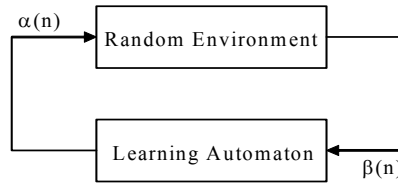

Figure 2. The relationship between the learning automaton and its random environment

Learning automata can be classified into two main families [16-20]: fixed structure learning automata and variable structure learning automata. Variable structure learning automata are represented by a triple $< \underline{\beta}, \underline{\alpha}, T >$, where $\underline{\beta}$ is the set of inputs, $\underline{\alpha}$ is the set of actions, and T is learning algorithm. The learning algorithm is a recurrence relation which is used to modify the action probability vector. Let $\alpha(k)$ and $\underline{p}(k)$ denote the action chosen at instant k and the action probability vector on which the chosen action is based, respectively. The recurrence equation shown by (1) and (2) is a linear learning algorithm by which the action probability vector $\underline{p}$ is updated. Let $\alpha_i(k)$ be the action chosen by the automaton at instant k. The action probabilities are updated as given in equation (1), when the chosen action is rewarded by the environment (i.e., $\beta(n) = 0$). When the taken action is penalized by the environment, the action probabilities are updated as defined in equation (2) (i.e., $\beta(n) = 1$).

$$p_j(n+1) = \begin{cases} p_j(n) + a[1 - p_j(n)] & j = i \\ (1-a)p_j(n) & \forall j \quad j \neq i \end{cases} \tag{1}$$

$$p_j(n+1) = \begin{cases} (1-b)p_j(n) & j = i \\ (\frac{b}{r-1}) + (1-b)p_j(n) & \forall j \quad j \neq i \end{cases} \tag{2}$$

where r is the number of actions can be chosen by the automaton, $a(k)$ and $b(k)$ denote the reward and penalty parameters and determine the amount of increases and decreases of the action probabilities, respectively. If $a(k) = b(k)$, the recurrence equations (1) and (2) are called linear reward-penalty (L_{R-P}) algorithm, if $a(k) >> b(k)$ the given equations are called linear reward-ε penalty ($L_{R-\varepsilon P}$), and finally if $b(k) = 0$ they are called linear reward-Inaction (L_{R-I}). In the latter case, the action probability vectors remain unchanged when the taken action is penalized by the environment. In the remaining of this section, some convergence results of the learning automata are summarized.

DEFINITION 2.1 The average penalty probability $M(n)$, received by a given automaton is defined as

$$M(n) = E[\beta(n) \mid \zeta_n]$$
$$= \int_{\alpha \in \underline{\alpha}} \zeta_n(\alpha) f(\alpha)$$

where $\zeta : \underline{\alpha} \to [0,1]$ specifies the probability of choosing each action $\alpha \in \underline{\alpha}$, and $\zeta_n(\alpha)$ is called the action probability.

If no priori information is available about f, there is no basis for selection of action. So, all the actions are selected with the same probabilities. This automaton is called *pure chance automaton* and its average penalty is equal to

$$M_0 = E[f(\alpha)]$$

DEFINITION 2.2 A learning automaton operating in a *P*-, *Q*-, or *S*-model environment is said to be *expedient* if

$$\lim_{n \to \infty} E[M(n)] < M_0$$

   Expediency means that when automaton updates its action probability function, its average penalty probability decreases. Expediency can also be defined as a closeness of $E[M(n)]$ to $f_l = \min_\alpha f(\alpha)$. It is desirable to take an action by which the average penalty can be minimized. In such case, the learning automaton is called *optimal*.

DEFINITION 2.3 A learning automaton operating in a *P*-, *Q*-, or *S*-model environment is said to be *absolutely expedient* if

$$E[M(n+1) \mid \underline{p}(n)] < M(n)$$

for all n and all $p_i(n)$.

Absolute expediency implies that $M(n)$ is a super martingale and $E[M(n)]$ is strictly decreasing for all n in all stationary environments. If $M(n) \leq M_0$, absolute expediency implies expediency.

DEFINITION 2.4 A learning automaton operating in a *P*-, *Q*-, or *S*-model environment is said to be *optimal* if

$$\lim_{n \to \infty} E[M(n)] = f_l$$

   Optimality implies that asymptotically the action for which penalty function attains its minimum value is chosen with probability one. While optimality appears a very desirable property, certain conditions in a given situation may preclude its environment. In such cases, a suboptimal performance is desirable. Such property is called ε-optimality and is defined in the following definition

DEFINITION 2.5 A learning automaton operating in a *P*-, *Q*-, or *S*-model environment is said to be ε-*optimal* if

$$\lim_{n \to \infty} E[M(n)] < f_l + \varepsilon$$

can be obtained for any $\varepsilon > 0$ by a proper choice of the parameters of the learning automaton. ε-optimality implies that the performance of the learning automaton can be made as close to the optimal as desired.

3.2. DISTRIBUTED LEARNING AUTOMATA

   A Distributed learning automata (DLA) [21] is a network of the learning automata which collectively cooperate to solve a particular problem. Formally, a DLA can be defined by a quadruple $< A, E, T, A_0 >$, where $A = \{A_1, ..., A_n\}$ is the set of learning automata, $E \subset A \times A$ is the set of the edges in which edge $e_{(i,j)}$ corresponds to the action α_j of the automaton A_i, T is the set of learning schemes with which the

learning automata update their action probability vectors, and A_0 is the root automaton of DLA from which the automaton activation is started . An example of a DLA has been shown in Figure 3.
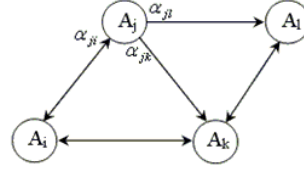

Figure 3. Distributed learning automata

The operation of a DLA can be described as follows: At first, the root automaton randomly chooses one of its outgoing edges (actions) according to its action probabilities and activates the learning automaton at the other end of the selected edge. The activated automaton also randomly selects an action which results in activation of another automaton. The process of choosing the actions and activating the automata is continued until a leaf automaton (an automaton which interacts to the environment) is reached. The chosen actions, along the path induced by the activated automata between the root and leaf, are applied to the random environment. The environment evaluates the applied actions and emits a reinforcement signal to the DLA. The activated learning automata along the chosen path update their action probability vectors on the basis of the reinforcement signal by using the learning schemes. The paths from the unique root automaton to one of the leaf automata are selected until the probability with which one of the paths is chosen is close enough to unity. Each DLA has exactly one root automaton which is always activated, and at least one leaf automaton which is activated probabilistically.

3.3. VARIABLE ACTION SET LEARNING AUTOMATA

A variable action set learning automaton is an automaton in which the number of actions available at each instant changes with time. It has been shown in [18] that a learning automaton with a changing number of actions is absolutely expedient and also ε -optimal, when the reinforcement scheme is L_{R-I}. Such an automaton has a finite set of n actions, $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$. $A = \{A_1, A_2, ..., A_m\}$ denotes the set of action subsets and $A(k) \subseteq \alpha$ is the subset of all the actions can be chosen by the learning automaton, at each instant k. The selection of the particular action subsets is randomly made by an external agency according to the probability distribution $q(k) = \{q_1(k), q_2(k), ..., q_m(k)\}$ defined over the possible subsets of the actions, where $q_i(k) = prob[A(k) = A_i \mid A_i \in A, 1 \le i \le 2^n - 1]$. $\hat{p}_i(k) = prob[\alpha(k) = \alpha_i \mid A(k), \alpha_i \in A(k)]$ is the probability of choosing action α_i, conditioned on the event that the action subset $A(k)$ has already been selected and also $\alpha_i \in A(k)$. The scaled probability $\hat{p}_i(k)$ is defined as

$$\hat{p}_i(k) = p_i(k) / R(k) \tag{3}$$

where $R(k) = \sum_{\alpha_i \in A(k)} p_i(k)$ is the sum of the probabilities of the actions in subset $A(k)$, and $p_i(k) = prob[\alpha(k) = \alpha_i]$.

The procedure of choosing an action and updating the action probabilities in a variable action set learning automaton can be described as follows. Let $A(k)$ be the action subset selected at instant k. Before choosing an action, the probabilities of all the actions in the selected subset are scaled as defined in equation (3). The automaton then randomly selects one of its possible actions according to the scaled action probability vector $\hat{p}(k)$. Depending on the response received from the environment, the learning automaton updates its scaled action probability vector. Note that the probability of the available actions is only updated. Finally, the probability vector of the actions of the chosen subset is rescaled as $p_i(k+1) = \hat{p}_i(k+1) \cdot R(k)$, for all $\alpha_i \in A(k)$. The absolute expediency and ε - optimality of the method described above have been proved in [18].

4. DLA-BASED VIRTUAL BACKBONE FORMATION

It is assumed that, the ad hoc network comprises a group of wireless hosts communicating through a common broadcast channel using omnidirectional antennas and all hosts have the same transmission range. That is, the corresponding topology graph is a unit disk graph in which each host H_i corresponds to a given vertex v_i, and every two hosts are connected and said to be neighbors, if there exists a direct bidirectional communication channel connecting them. Therefore, the network graph is assumed to be undirected. Scheduling of transmissions is the responsibility of the MAC layer, and like many existing approaches, we are not concerned with the issues of using a shared wireless channel to send the messages avoiding the collisions and contentions. Each host has a unique ID (e.g., IP address) and also needs to know its neighbors' ID.

In this section, a distributed learning automata-based algorithm called DLA-BF is proposed for virtual backbone formation in wireless ad hoc networks, which focuses on finding a near optimal solution to the minimum CDS problem in the network graph. In this approach, each host (e.g., H_i) is equipped by a learning automaton (e.g., A_i) and so a network of learning automata isomorphic to the network graph (which is a UDG) is formed. The resulting network of learning automata can be described by a duple $< \underline{A}, \underline{\alpha} >$, where $\underline{A} = \{A_1, A_2, ..., A_m\}$ denotes the set of the learning automata corresponding to the vertex set (or the set of hosts), and $\underline{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_m\}$ denotes the set of actions in which $\alpha_i = \{\alpha_{i,1}, \alpha_{i,2}, ..., \alpha_{i,ri}\}$ defines the action set can be taken by learning automata A_i.

4.1. ACTION-SET FORMATION METHOD

In the proposed algorithm, to form the action set of each learning automaton A_i, its corresponding host (i.e., H_i) propagates locally a message to its one-hop neighbors. The hosts which are within the transmission range of the sender host, upon receiving the message, reply it and return their action set information. The sender forms its action set on the basis of the received replies, so that each host H_i by which the message is replied is associated with action $\alpha_{i,j}$ in the action set of automaton A_i. Action $\alpha_{i,j}$ corresponds to the selection of host H_i as a dominator host by learning automaton A_i. Therefore, the action set size of each learning automaton is strongly dependent on the degree of its corresponding host, and consequently, on the network density. Assuming that the hosts are uniformly distributed in the network, the average number of actions of each learning automaton is $n\pi R^2 / A$, each of initial probability $A / n\pi R^2$, where n denotes the number of hosts, A denotes the size of the square area, and R denotes the radio transmission range.

The problem with the action set defined above is that the number of actions is fixed and does not vary (with time) as the algorithm proceeds. This may result in a host to be chosen many times, the virtual backbone contains loops and suffers from the redundant dominators by which no more hosts can be spanned. Therefore, the fixed action set decreases the convergence speed of algorithm and increases the virtual backbone size also. To overcome these shortcomings, we propose the learning automata with changing number of actions, and introduce the following rules for pruning the action set of such learning automata.

RULE I. To avoid choosing the same dominators (by different hosts), each activated learning automaton is allowed to prune its action set by disabling the actions corresponding to the dominator hosts selected earlier. This rule increases the convergence speed, and consequently, decreases the running time of the proposed algorithm.

RULE II. To avoid the loops and the redundant dominator hosts by which no more (dominatee) hosts can be spanned, the proposed algorithm prunes the action set as follows. As mentioned earlier, when host H_i is going to form the action set of its automaton, it receives some messages from its neighboring hosts which include the action set information of these hosts. Depending on received information, activated

automaton A_i updates its action set by disabling the actions corresponding to the hosts whose one-hop neighbors all have been spanned (or added to the dominatee set) before (see Figures 5(a)-5(f)), if any. This rule reduces the dominator set (backbone) size, decreases the running time and improves the convergence rate of algorithm.

In these rules, the action probability vector of each activated learning automaton is scaled as described in Section 3.3, on the variable action set learning automata, in such a way that the probability of choosing the hosts corresponding to the disabled actions is temporarily set to zero. At the end of each iteration, the disabled actions of each activated learning automaton must be enabled for the next iteration.

4.2. BACKBONE FORMATION PROCESS

Each host included in the CDS (virtual backbone) is called a dominator host, otherwise a dominatee host. Indeed, a dominatee host is a one-hop neighbor of at least one host in the CDS, if it is not included in the CDS. At each iteration, the dominator hosts, which are selected as dominators, form the virtual network backbone. The learning automata iteratively construct the virtual backbones and update the action probability vectors until they find a near optimal solution to the MCDS problem that guarantees the optimality of the formed virtual backbone. Each host requires to support the following data structures to participate in the backbone formation process:

- *MAX_ITERATION*, a stopping condition for the algorithm as a maximum number of iterations.
- *DOMINATOR_SET,* the set of chosen dominator hosts.
- *DOMINATEE_SET*, a set of hosts in which each member is a one-hop neighbor of at least one dominator host in the *CDS*.
- *PCDS*, a threshold required for termination the backbone formation process as the product of the probability of choosing the dominator hosts in the *CDS*.
- *PROB_VECTOR*, a vector of the probability of choosing the members of the *CDS*.
- *MIN_SIZE*, a dynamic threshold contains the cardinality of the smallest *CDS* which has been selected yet. This is initially set to the network size.
- *ITERATION_NUM*, a counter which keeps the number of constructed *CDS*.

A *READY* message, which contains *MAX_ITERATION* and *PCDS*, is initially flooded within the network to inform the hosts of the virtual backbone formation start. Each host, upon receiving the *READY* message, calls the *DOMINATION* procedure. During the backbone formation procedure, each host may receive from or send to the other hosts the following messages: *INITIALIZATION, ACTIVATION*, *REWARDING*, *PENALIZING* and *BACKBONE* message. After the *READY* message is sent, one of the hosts is randomly chosen and denoted as the initial dominator host H_i. An *INITIALIZATION* message is then sent to the initial dominator host and the backbone formation process is continued by receiving the *INITIALIZATION* message in dominator host H_i as described below.

4.2.1. INITIALIZATION MESSAGE

When a given host H_i receives an *INITIALIZATION* message, it activates its corresponding automaton and updates the *DOMINATOR_SET* by adding its ID number. The initial dominator host then adds its ID and its one-hop neighbors' ID to the *DOMINATEE_SET*. It applies rules I and II, and chooses one of its actions, if there exist more actions and size of *DOMINATEE_SET* is smaller than the network size according to its action probability vector. The probability with which learning automaton L_i chooses its action is added to the *PROB_VECTOR*. If there exist more actions and size of *DOMINATEE_SET* is smaller than the network size, an *ACTIVATION* message is sent to the host corresponding to the chosen action. The ACTIVATION message is described below.

4.2.2. ACTIVATION MESSAGE

An *ACTIVATION* message includes *DOMINATEE_SET*, *DOMINATOR_SET*, *MIN_SIZE*, *PROB_VECTOR*, and *ITERATION_NUM*. When a given host H_i receives an *ACTIVATION* message, it activates its corresponding learning automaton and inserts its ID as a new dominator host into the *DOMINATOR_SET*. To update the *DOMINATEE_SET* it adds its ID and its one-hop neighbors' ID to this set. The action set of learning automaton L_i is updated by disabling the actions as described in subsection 4.1. Now, if there exist actions to be chosen by learning automaton L_i, and the size of the

DOMINATEE_SET is still smaller than the network size, learning automaton L_i chooses one of its actions as a new dominator host, updates *PROB_VECTOR* by adding the probability of choosing the action, and sends an *ACTIVATION* message to the chosen dominator host. Else, the cardinality of the *DOMINATOR_SET* is calculated. If this cardinality is less than the dynamic threshold *MIN_SIZE*, and *DOMINATEE_SET* size equals to the network size, the dynamic threshold *MIN_SIZE* is set to the cardinality of the selected set (for future iterations) and all the chosen actions of the activated automata (corresponding to the dominator hosts) are rewarded by sending back a *REWARDING* message, otherwise, they are penalized by sending back a *PENALIZING* message.

To verify the *stopping condition* of the backbone formation process, the probability of choosing the *DOMINATOR_SET* needs to be calculated as the product of the probability of choosing the dominator hosts based on the information contained in *PROB_VECTOR*. If this probability is greater than the certain threshold *PCDS* or *ITERATION_NUM* exceeds a per-specified threshold *MAX_ITERATION*, dominator host H_i generates a *BACKBONE* message including the last selected *DOMINATOR_SET* and broadcasts it in the network. Otherwise, it increments the *ITERATION_NUM* and starts a new iteration by randomly choosing one of its actions. It sends an *INITIALIZATION* message to the initial dominator host corresponding to the taken action. In what follows, we describe the *BACKBONE, PENALIZING* and *REWARDING* messages which are used in an *ACTIVATION* message.

4.2.3. BACKBONE MESSAGE

A *BACKBONE* message includes the *DOMINATOR_SET* selected during the last iteration. When host H_i receives a *BACKBONE* message, it is noticed that the backbone formation process has been completed, and so accepts the set of dominator hosts, contained in the *BACKBONE* message, as a new virtual backbone. It then assumes the role of a dominator host, if it finds its ID number in the *BACKBONE* message. Otherwise it acts as a dominatee host. It will then terminate the *DOMINATION* procedure.

4.2.4. REWARDING MESSAGE

When dominator host H_i receives a *REWARDING* message, it updates its action probability vector by rewarding chosen action $\alpha_{i,j}$ as

$$p_{i,j}(n+1) = p_{i,j}(n) + a[1 - p_{i,j}(n)], \tag{4}$$

where $p_{i,j}$ is the probability with which host H_i chooses host H_i as a domintor, and penalizing the other actions $\alpha_{i,k}$, for all $k \neq j$, as

$$p_{i,k}(n+1) = (1-a)p_{i,k}(n) \qquad \forall k \ \ k \neq j. \tag{5}$$

After rewarding the chosen action, the scaled action probability vector must be updated once again (or rescaled) by enabling all the disabled actions according to the rescaling method described in Section 3.3 on the variable action set learning automata.

4.2.5. PENALIZING MESSAGE

Since the reinforcement scheme by which the learning automata update their action probability vectors is L_{R-I}, the action probabilities of the activated learning automata (corresponding to the dominator hosts) remain unchanged when they receive a *PENALIZING* message. In this case, the disabled actions of each activated learning automaton are enabled again. Figure 4 shows the flowchart of the *DOMINATION* procedure which is run at host H_i.
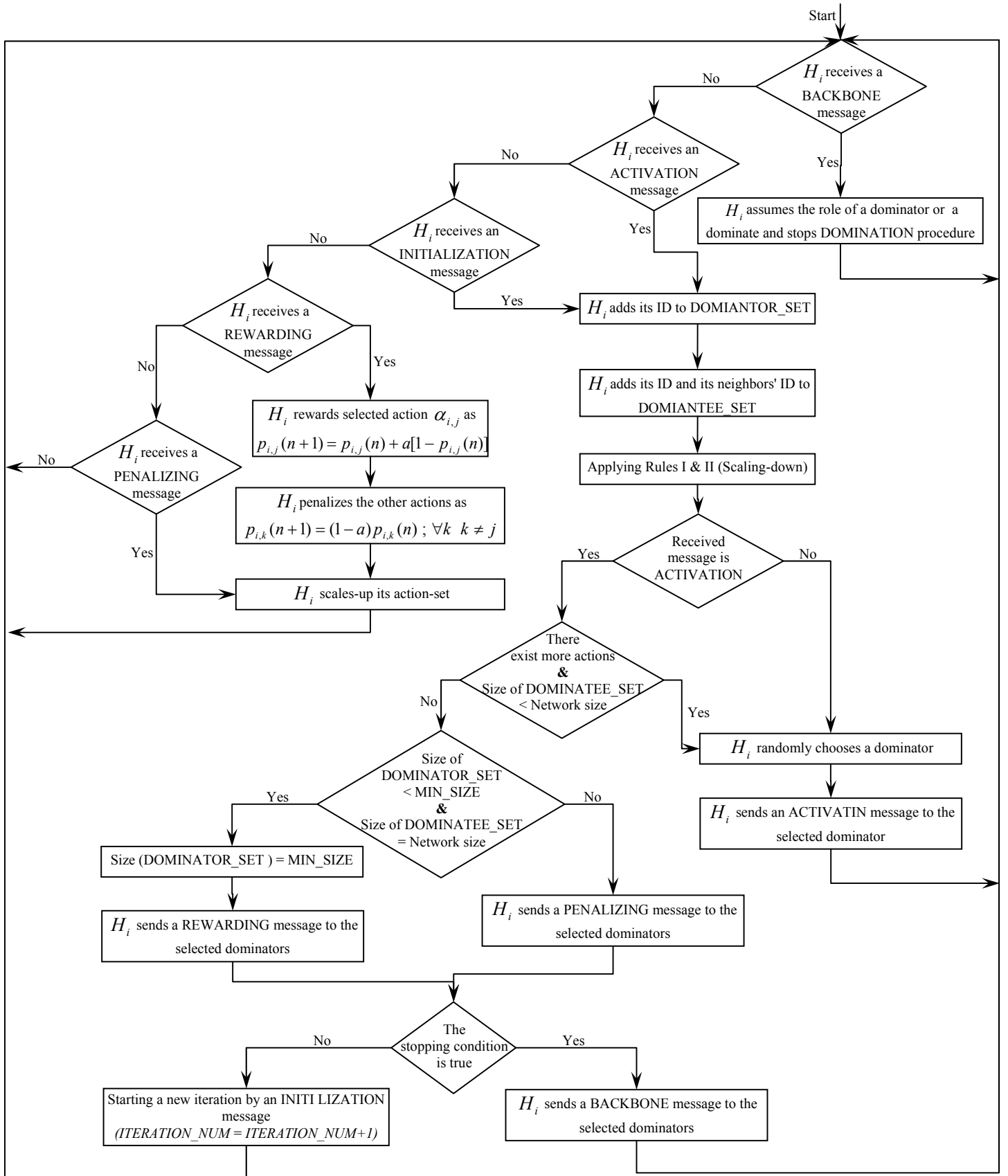
Figure 4. Flowchart of the *DOMINATION* procedure which is run at host H_i

In ad hoc networks, hosts are free to join or leave the network at any time. The proposed algorithm divides the hosts as dominators and dominatees. Each dominator host takes the responsibility of forwarding the messages to its neighboring hosts, while the dominatee hosts only receive the messages. When a host is going to join the network, it sends a *JOIN_REQUEST* message to its neighboring hosts and then waits for a certain period of time. Each dominator host, upon receiving a *JOIN_REQUEST*, acknowledges it by means of a *JOIN_ACK* message. If the newly joining host receives one or more *JOIN_ACK* messages, it finds out that it is within the transmission range of at least one dominator host, and so it will receive the messages. Otherwise, if it receives no *JOIN_ACK* messages, it selects one of its (dominatee) neighboring hosts (e.g., the host with highest ID number) to change its role as a dominator host. When a host decides to leave the network, no control message is required, if it is a dominatee host, otherwise, when a dominator host is going to leave the network, a backbone reformation process needs to be initiated so as to modify the network backbone.

4.3. An Example

Figures 5(a)-5(i) illustrate the step-by-step backbone formation process in our proposed algorithm for an example ad hoc network. The ad hoc network graph has 11 nodes (hosts) and 18 edges. In Figures 5(a)-5(i), a black circle represents a dominator host and the number beside it represents the host ID. An encircled node represents an initial dominator host. The set near a host represents the action set of the learning automaton corresponding to it. For instance in Figure 5(b), since hosts 2, 3 and 5 are within the transmission range of host 1, the action set of the learning automaton corresponding to host 1 contains three actions for choosing hosts 2, 3 and 5 as dominators. A crossed action represents a disabled action which can not be temporarily chosen by the learning automaton (e.g., action 3 in Figure 5(b)). A grey circle represents a dominatee host, and so a white circle represents a host that it is neither a dominator nor a dominatee. An arrow represents a message and the text beside it represents the type of the message. In what follows, three possible execution scenarios are discussed to construct the network backbone in the first three iterations of the proposed backbone formation algorithm.

- As shown in Figure 5(a), let us assume that host 1 receives an *INITIALIZATION* message and starts the first iteration of the backbone formation process. It adds its ID to the *DOMINATEE_SET* and *DOMINATOR_SET*. Hosts 2, 3 and 5 form the initial action set of the learning automaton (i.e., $\{2,3,5\}$) corresponding to host 1, and so their ID are added to the *DOMINATEE_SET*. Applying rule II to its action set, host 3 is disabled, and host 1 randomly chooses one of its remaining actions, according to the scaled action set $\{2,5\}$. Let us assume that host 5 to be chosen as the second dominator host. Host 1 sends an *ACTIVATION* message to host 5 (see Figure 5(b)).

- Host 5 receives the *ACTIVATION* message and adds its ID to the *DOMINATOR_SET* and *DOMINATEE_SET* too. It adds its one-hop neighbors' ID to the *DOMINATEE_SET* also. In this stage, the dominator and dominatee sets are updated to $\{1,5\}$ and $\{1,2,3,4,5,8\}$, respectively. As described earlier, the action set of the current dominator host is scaled down by applying the action disabling rules. Rules I and II disable the actions corresponding to the (selected) dominator hosts 1 and hosts 3, respectively. Then, host 5 chooses the third dominator host among hosts 4 and 8 at random. It then sends an *ACTIVATION* message to this host (see Figure 5(c)).

- Assume that host 5 chooses host 4, host 4 chooses host 7, host 7 chooses host 9, and finally it chooses host 11(see Figures 5(d)-5(f)). So the *DOMINATOR_SET* is $\{1,4,5,7,9,11\}$.

- Host 11 receives the *ACTIVATION* message, updates the *DOMINATOR_SET*, *DOMINATEE_SET*, and its action set. It terminates the current iteration, when it finds out that there are no more actions to be chosen and the *DOMINATEE_SET* size is equal to the network size too.

- Host 11 compares the cardinality of the *DOMINATOR_SET* with the dynamic threshold *MIN_SIZE*. Assume that this is the first iteration of algorithm, and so the *MIN_SIZE* is equal to the network size (i.e., 11). Therefore, the *DOMINATOR_SET* size (i.e., 6) is less than the *MIN_SIZE*, and *MIN_SIZE* is set to the size of the *DOMINATOR_SET*. Host 11 generates a *REWARDING* message and sends it back to the dominator hosts (floods it) (see Figure 5(g)). Note that in Figures 5(g)-5(i), the arrows only show the shortest paths to the dominator hosts for the flooded *REWARDING* message.

- After receiving the *REWARDING* message, all activated learning automata (corresponding to the dominator hosts) reward their chosen actions by a L_{R-I} reinforcement scheme and then enable the

disabled actions. This increases the probability of choosing the selected *DOMINATOR_SET* (i.e., {1,4,5,7,9,11}) for future iterations.

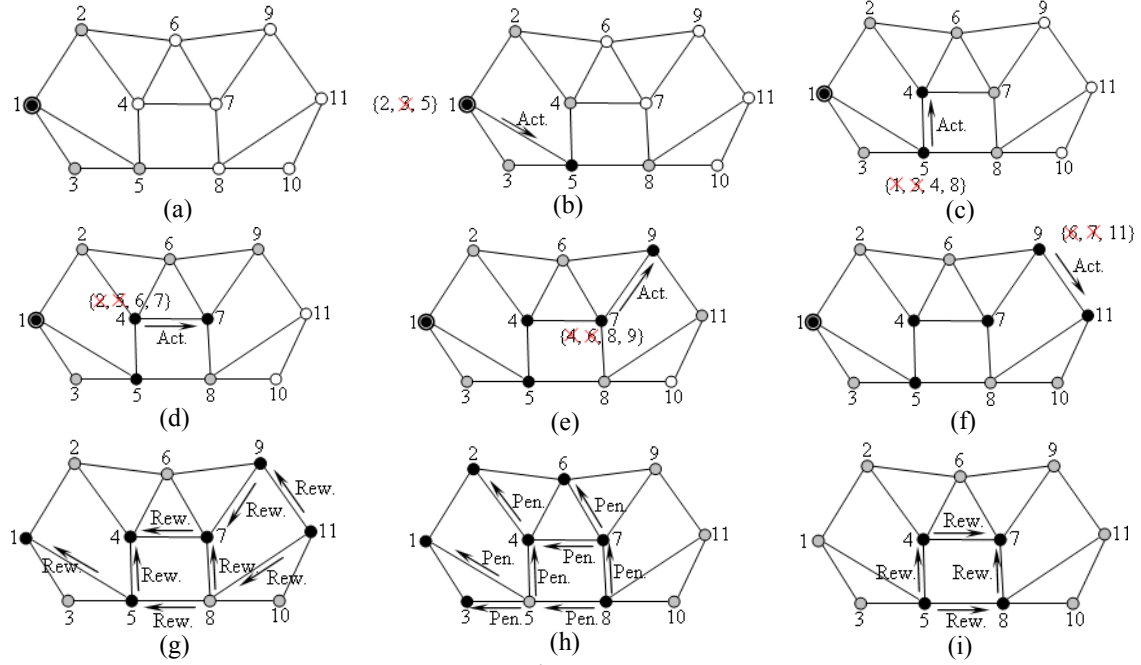- Host 11 checks the stopping condition and finds out it is false, so resumes a new iteration.

Figure 5.

At the beginning of each iteration, *DOMINATOR_SET, DOMINATEE_SET* and *PROB_VECTOR* are reset.

- Host 11 starts the second iteration by choosing a new initial dominator host, and sends an *INITIALIZATION* message to it. Assume that host 3 to be chosen as the initial dominator host, it chooses host 1, host 1 chooses host 2, host 2 chooses host 4, host 4 chooses host 6, host 6 chooses host 7, and finally host 7 chooses host 8 as the dominator hosts. The dominator set size is 7 and larger than dynamic threshold *MIN_SIZE*, so host 8 generates a *PENALIZING* message and sends it back to the selected dominator hosts (see Figure 5(h)). As described earlier, the dominator hosts penalize their selected actions and then enable the disabled actions. Thus, the probability of choosing this dominator set decreases. The stopping condition is false and so host 8 starts a new iteration.
- At the third iteration, we assume that host 4 to be chosen as the initial dominator host, it chooses host 7, host 7 chooses host 8, and eventually host 8 chooses host 5. The backbone size is 4 and smaller than the dynamic threshold *MIN_SIZE*. Therefore, the chosen actions are rewarded (see Figure 5(i)).
- The process of constructing the virtual backbones (dominator sets) is repeated until the stopping condition is satisfied. At each iteration, the dominators are rewarded, if they form a set smaller than the dynamic threshold *MIN_SIZE*, and penalized otherwise. The last dominator set which is selected before the algorithm stops is chosen as the minimum size virtual backbone and broadcast through the network by sending *a BACKBONE* message.

As described earlier, a CDS is said to be Steiner CDS, if only a particular subset of the nodes needs to be dominated by the CDS. A Steiner CDS of a given graph is a subset of its CDS. That is, a CDS is a Steiner CDS, where all nodes must be dominated. Therefore, the proposed algorithm is capable of solving the minimum Steiner CDS problem as well. On the other hand, finding a Steiner CDS in the network, which dominates an arbitrary subset of hosts, forms a virtual backbone along which the multicast packets can be sent to the given hosts as the multicast members. Hence, the proposed backbone formation algorithm can be effectively used in multicast routing protocols. To implement the proposed algorithm in multicast routing protocols, the *ACTIVATION* message needs to be changed so that each iteration of

algorithm terminates when *DOMINATEE_SET* includes all the multicast members or the current host has no more actions.

5. Complexity Analysis

Li et al.'s greedy algorithm [14] is known for constructing the smallest connected dominating sets among existing MIS-based algorithms. The message complexity and computation time of this algorithm are $O(n\Delta^2)$ and $O(\Delta)$, respectively, where Δ is the maximum degree in the graph. Butenko et al.'s algorithm [13] is also a prune-based algorithm which performs even much better than Li et al.'s algorithm to construct the small CDSs. The running time and message complexity of Butenko et al.'s algorithm [13] are $O(n\log^3(n))$ and $O(n^2\log^3(n))$, respectively, where n is the number of nodes in the network graph. Therefore, the performance superiority of Butenko et al.'s algorithm over Li et al.'s algorithm is due to the high message and computation complexities. Xie et al.'s algorithm [15] uses a hierarchical graph to construct the CDS, and selects a set of nodes at each hierarchical level to route the messages for the other nodes in the next hierarchical level. The computation complexity and message complexity of Xie et al.'s algorithm are $O(\theta^2)$ and $O(\theta)$, respectively, where θ is the maximum number of child nodes in the hierarchical graph. In [15], the authors claim that the maximum number of child nodes in a hierarchical graph is less than or equal to the maximum degree in the graph (i.e., $\theta \le \Delta$), and so their algorithm outperforms Li et al.'s algorithm in terms of the message complexity. The results given in [15] show that, in addition to the lower message complexity of Xie et al.'s algorithm, the size of the connected dominating sets constructed by this algorithm is even slightly better than Butenko et al.'s algorithm.

The aim of the proposed algorithm is minimizing the network backbone size. For this purpose, the learning rate of algorithm must be chosen as small as possible. On the other side, in ad hoc networks, where the hosts suffer from the strict resource limitations, the communication and processing overheads should be kept as low as possible. Since the computational and message complexities of the proposed algorithm are inversely proportional to the learning rate, to alleviate the network overheads, the learning rate must be chosen as large as possible. Therefore, to make a trade-off between the computational and message complexities of algorithm with the backbone size (backbone formation optimality) the learning rate should be selected carefully. It can be chosen empirically (like our approach in Section 6) or theoretically. Narendra and Thathachar [16], using Martingale convergence theorems, determined a lower bound on the probability of convergence to the desired (optimal) action for each learning rate, when the learning automaton uses a L_{R-I} learning scheme. They also showed that by a proper choice of the learning rate, the lower bound on the convergence probability to the desired action is close to unity. Beigy and Meybodi [22] also showed for the distributed learning automata that for every error parameter $\varepsilon > 0$, there exists a learning rate $a^*(\varepsilon) \in (0,1)$ such that for all learning rate $a \in (0, a^*)$, the probability of convergence to the optimal solution is greater than $1-\varepsilon$. It can be shown that the same results hold true for our proposed DLA-based algorithm. In the remaining of this section, we estimate the worst case running time (Theorem 1) and message complexity (Theorem 2) of the proposed backbone formation algorithm to find a $1/(1-\varepsilon)$ optimal size backbone on the basis of a pre-specified learning rate.

Theorem 1. Let OPT denotes the size of the smallest backbone (or CDS) in network graph G, and $\underline{q}(k) = \{q_1(k), \ldots, q_r(k)\}$ is updated according to the proposed backbone formation algorithm (DLA-BF). The time required for finding a $\dfrac{1}{1-\varepsilon} \cdot |OPT|$ (for $0 < \varepsilon < 1$) size backbone (e.g., ω_i) in DLA-BF is not longer than

$$[2/(1-\varepsilon+q_i^g)] \cdot \log_{1-a}^{\frac{1-\sqrt[m]{1-\varepsilon}}{1-\sqrt[m]{q_i^g}}} + r$$

where $q_i(k)$ denotes the probability of choosing backbone ω_i at stage k, $\varepsilon \in (0,1)$ is the error parameter of algorithm denoted as $1-PCDS$ in DLA-BF, a denotes the learning rate of algorithm, n is the number of nodes in the network graph, and r is the number of backbones.

PROOF. Let $q_i = \prod_{h_j \in \omega_i} p_j$ be the initial probability of choosing backbone ω_i, and p_j be the initial

probability of choosing host H_j as a dominator host. The worst case occurs when all the other backbones to be chosen before the smallest backbone ω_i. In this case, the learning process can be divided into two distinct phases. In the first phase, called *shrinking phase,* it is assumed that all the other backbones to be chosen, from the largest to the smallest, and so rewarded before ω_i. Such an ordered (backbone) selection procedure decreases the probability of choosing ω_i no more than that given in inequality (6). The second phase called *growing phase* is started when the backbone ω_i is chosen for the first time. According to the proposed algorithm, during the growing phase, the probability of penalizing ω_i, and rewarding the other backbones is zero. Therefore, the conditional expectation of $q_i(k)$ (i.e., the probability of choosing backbone ω_i at stage k), increases only when ω_i is rewarded. In other words, during the growing phase, the changes in the conditional expectation of $q_i(k)$ is always non-negative and given in equation (7). As described in the proposed algorithm, the growing phase is continued until the probability of choosing backbone ω_i is greater than or equal to $1-\varepsilon$ (or $PCDS$). Let q_i^s denotes the probability of choosing backbone ω_i at the beginning of the shrinking phase, and a denotes the learning rate of the proposed algorithm. Therefore, during the shrinking phase, the probability of choosing backbone ω_i changes as

$$q_i^s(r) \geq q_i^s(r-1) \cdot (1-a)^m \qquad (6)$$

where r is the number of all possible network backbones, and m denotes the average backbone size. Substituting recurrence function $q_i^s(r-1)$ in inequality above, we have

$$q_i^s(r) \geq q_i^s(r-2) \cdot (1-a)^{2m}$$

By repeatedly applying inequality (6) $(r-1)$ times, we obtain

$$q_i^s(r-1) \geq q_i^s \cdot (1-a)^{(r-1) \cdot m}$$

where $q_i^s(r-1)$ denotes the probability of choosing backbone ω_i at the end of the shrinking phase. For the sake of simplicity in notation, $q_i^s(r-1)$ is substituted by q_i^g, where $q_i^g \geq \prod_{h_j \in \omega_i} p_j (1-a)^{(r-1)}$ is the

probability of choosing ω_i at the beginning of the growing phase and $p_j(1-a)^{(r-1)}$ is substituted by ρ_j. As mentioned earlier, in the growing period, q_i^g increases, if ω_i is rewarded, and remains unchanged otherwise. Thus, during this phase, the probability of choosing backbone ω_i increases as

$$q_i^g(1) \geq \prod_{h_j \in \omega_i} \rho_j + a \cdot (1 - \rho_j)$$

$$q_i^g(2) \geq \prod_{h_j \in \omega_i} \rho_j(1) + a \cdot (1 - \rho_j(1))$$

$$= \prod_{h_j \in \omega_i} \rho_j(1) \cdot (1 - a) + a$$

$$\vdots$$

$$q_i^g(k-1) \geq \prod_{h_j \in \omega_i} \rho_j(k-2) + a \cdot (1 - \rho_j(k-2)) \tag{7}$$

$$= \prod_{h_j \in \omega_i} \rho_j(k-2) \cdot (1-a) + a$$

$$q_i^g(k) \geq \prod_{h_j \in \omega_i} \rho_j(k-1) + a \cdot (1 - \rho_j(k-1))$$

$$= \prod_{h_j \in \omega_i} \rho_j(k-1) \cdot (1-a) + a$$

where $q_i^g(k)$ (i.e., *PCDS* in DLA-BF) denotes the probability with which backbone ω_i is chosen at the end of the growing phase, which according to the theorem, it is assumed to $1 - \varepsilon$. After some algebraic simplification, we have

$$q_i^g(k) \geq \prod_{h_j \in \omega_i} \rho_j(k-1) \cdot (1-a) + a$$

$$= \prod_{h_j \in \omega_i} \rho_j(k-2) \cdot (1-a)^2 + a \cdot (1-a) + a$$

$$= \prod_{h_j \in \omega_i} \rho_j(k-3) \cdot (1-a)^3 + a \cdot (1-a)^2 + a \cdot (1-a) + a \tag{8}$$

$$\vdots$$

$$= \prod_{h_j \in \omega_i} \rho_j(1) \cdot (1-a)^{k-1} + a \cdot (1-a)^{k-2} + \cdots + a \cdot (1-a) + a$$

$$= \prod_{h_j \in \omega_i} \rho_j \cdot (1-a)^k + a \cdot (1-a)^{k-1} + \cdots + a \cdot (1-a) + a$$

Hence, we have

$$q_i^g(k) \geq \prod_{h_j \in \omega_i} [\rho_j \cdot (1-a)^k + a \cdot (1-a)^{k-1} + \cdots + a \cdot (1-a) + a] \tag{9}$$

where k denotes the number of times backbone ω_i must be selected until

$$q_i^g(k) = 1 - \varepsilon \tag{10}$$

From inequality (9), it follows that, the running time (the number of iteration) of the proposed algorithm is computed, if we find a value of k (for a given learning rate a) under which the condition given in equation (10) to be satisfied. From equation (9) we have

$$q_i^g(k) \geq \prod_{h_j \in \omega_i} \left[\rho_j \cdot (1-a)^k + a \cdot \left(1 + (1-a) + (1-a)^2 + \cdots + (1-a)^{k-1} \right) \right] \tag{11}$$

and so

$$q_i^g(k) \geq \prod_{h_j \in \omega_i} \left[\rho_j \cdot (1-a)^k + \sum_{i=0}^{k-1} a \cdot (1-a)^i \right] \tag{12}$$

The second term on the right hand side of equation (12) is a geometric series that sums up to $a \cdot \left(1 - (1-a)^k / 1 - (1-a)\right)$, where $|1-a| < 1$. Therefore, we have

$$q_i^g(k) \geq \prod_{h_j \in \omega_i} \left[\rho_j \cdot (1-a)^k + a \cdot \left(\frac{1 - (1-a)^k}{1 - (1-a)} \right) \right] \tag{13}$$

Substituting ρ_j with $p_j (1-a)^{(r-1)}$ and some simplification, we have

$$q_i^g(k) \geq \prod_{h_j \in \omega_i} \left[(1-a)^k \left(p_j \cdot (1-a)^{(r-1)} - 1 \right) + 1 \right]$$

For simplicity, but without lose of generality, we assume that the host are uniformly distributed within the network. Therefore, the network density is uniform and the hosts have the same degree. That is, all learning automata have the same number of actions, and the initial probability p_j equals to c, for all h_j. Hence, we have,

$$q_i^g(k) \geq \left[(1-a)^k \left(c \cdot (1-a)^{(r-1)} - 1 \right) + 1 \right]^m$$

From equation (10) and (13) we have

$$\left[(1-a)^k \left(c \cdot (1-a)^{(r-1)} - 1 \right) + 1 \right]^m = 1 - \varepsilon$$

$$(1-a)^k \left(c \cdot (1-a)^{(r-1)} - 1 \right) + 1 = \sqrt[m]{1 - \varepsilon}$$

Since $\left[c \cdot (1-a)^{(r-1)} \right]^m = q_i^g$, we have

$$(1-a)^k \cdot \left(\sqrt[m]{q_i^g} - 1 \right) + 1 = \sqrt[m]{1 - \varepsilon}$$

Hence, we have

$$(1-a)^k = \frac{1 - \sqrt[m]{1 - \varepsilon}}{1 - \sqrt[m]{q_i^g}} \tag{14}$$

Taking $\log_{1-a}$ of both sides of equation (14), we derive

$$\log_{1-a}^{(1-a)^k} = \log_{1-a}^{\frac{1 - \sqrt[m]{1-\varepsilon}}{1 - \sqrt[m]{q_i^g}}}$$

and thus the number of times ω_i is rewarded during the growing phase, apart from penalizing the other backbones, is obtained as

$$k = \log_{1-a}^{\frac{1 - \sqrt[m]{1-\varepsilon}}{1 - \sqrt[m]{q_i^g}}} \tag{15}$$

Since during the growing phase, q_i^g remains unchanged when the other backbones are penalized, k does not include the number of times the other backbones are chosen and this should be separately calculated based on k. Let q_i^g be the probability of choosing backbone ω_i at the beginning of the growing phase, and reaches $1 - \varepsilon$ (or $PCDS$ in DLA-BF) after k iterations. On the other hand, the probability of choosing the other backbones is initially $1 - q_i^g$, and reaches ε after the same number of iterations. Thus, the number of times the other backbones are chosen (before satisfying the condition given in equation (10)) is obtained as

$$\left(\frac{1 + \varepsilon - q_i^g}{1 - \varepsilon + q_i^g} \right) \cdot k$$

After some algebraic manipulations, the total number of iterations required in the growing phase of the proposed algorithm (i.e., K) to satisfy the condition given in equation (10) is obtained as

$$K = [2/(1 - \varepsilon + q_i^g)] \cdot k$$

By substituting k from equation (15) and we have

$$K = [2/(1 - \varepsilon + q_i^g)] \cdot \log_{1-a}^{\frac{1 - \sqrt[m]{1-\varepsilon}}{1 - \sqrt[m]{q_i^g}}} \tag{16}$$

From equation (16), the running time of the growing phase can be estimated. Since the worst case of the algorithm occurs when (in the shrinking phase) all the other backbones to be chosen before ω_i, the running time of the shrinking phase is always less than $O(r)$. Therefore, we have

$$T(K) \leq [2/(1 - \varepsilon + q_i^g)] \cdot \log_{1-a}^{\frac{1 - \sqrt[m]{1-\varepsilon}}{1 - \sqrt[m]{q_i^g}}} + r \tag{17}$$

where $q_i^g = \left[c \cdot (1-a)^{(r-1)}\right]^m$, which completes the proof of this theorem.   □

THEOREM 2. The message complexity of the proposed backbone formation algorithm for finding a $\dfrac{1}{1-\varepsilon} \cdot |\text{OPT}|$ (for $0 < \varepsilon < 1$) size backbone is at most

$$m \cdot ([2/(1 - \varepsilon + q_i^g)] \cdot \log_{1-a}^{\frac{1 - \sqrt[m]{1-\varepsilon}}{1 - \sqrt[m]{q_i^g}}} + r) \tag{18}$$

where OPT denotes the size of the minimum backbone (optimal solution to the MCDS problem), $q_i(k)$ denotes the probability of choosing backbone ω_i at stage k, $\varepsilon \in (0,1)$ is the error parameter of algorithm, a denotes the learning rate of algorithm, n is the number of hosts in the network graph, and r is the number of backbones.

PROOF. As proved in Theorem 1, the running time (number of iterations) of the proposed backbone formation algorithm to find a $\dfrac{1}{1-\varepsilon} \cdot |\text{OPT}|$ size backbone for the network graph is at most

$$[2/(1 - \varepsilon + q_i^g)] \cdot \log_{1-a}^{\frac{1 - \sqrt[m]{1-\varepsilon}}{1 - \sqrt[m]{q_i^g}}} + r$$

Furthermore, as described in Section 4, at each iteration of the backbone formation algorithm, the number of messages needs to be sent to form a backbone is equal to the number of hosts in the selected backbone. Hence, the message complexity of the backbone formation algorithm is smaller than

$$m \cdot ([2/(1 - \varepsilon + q_i^g)] \cdot \log_{1-a}^{\frac{1 - \sqrt[m]{1-\varepsilon}}{1 - \sqrt[m]{q_i^g}}} + r)$$

where m is the average backbone size, and hence the proof of the theorem is completed.   □

The backbone size is directly proportional to the learning rate (i.e., a) of algorithm so that the backbone size increases as the learning rate increases, and it decreases when the learning rate of algorithm decreases. On the other hand, as proved in theorems 1 and 2, the computational and message complexities of the proposed algorithm are inversely (and logarithmically) proportional to the learning rate of algorithm. Computational and message complexities increase as the learning rate decreases, and they decrease as the learning rate increases. Therefore, to make a trade-off between the running time and message complexity of the proposed algorithm with the backbone size (backbone formation optimality), the learning rate of algorithm should be carefully selected. The main superiority of the proposed DLA-based backbone formation algorithm over the existing methods is the capability of adjusting the learning rate in such a way that a desired performance (near optimal backbone size) for each application of ad hoc network can be achieved in a reasonable running time and message complexity.

6. Numerical Results

To study the performance of the proposed backbone formation algorithm, we have conducted several simulation experiments (Experiments 1-7) in two groups. The first group of experiments is concerned with investigating the impact of the learning rate of algorithm on the backbone (CDS) size, and message overhead of algorithm, and the second group evaluates the results of the proposed algorithm in comparison with those of the best-known CDS-based backbone formation algorithms.

Li et al.'s greedy algorithm [14] is known for constructing the smallest connected dominating sets among existing MIS-based algorithms. Butenko et al.'s algorithm [13] is also a prune-based algorithm which performs even much better than Li et al.'s algorithm to construct the small CDSs. The performance superiority of Butenko et al.'s algorithm over Li et al.'s algorithm is due to the high message and computation complexities. Xie et al.'s algorithm [15] uses a hierarchical graph to construct the CDS, and selects a set of nodes at each hierarchical level to route the messages for the other nodes in the next hierarchical level. Although the message complexity of Xie et al.'s algorithm [15] is much less than Butenko et al.'s algorithm, the size of the backbones constructed in Xie's algorithm is even slightly better than Butenko's algorithm. Therefore, comparing our proposed backbone formation method with these algorithms will determine how much our method performs well.

In simulation experiments, IEEE 802.11 is used as the MAC layer protocol, and wireless hosts communicate through a common broadcast channel using omnidirectional antennas and all hosts have the same radio transmission range $R = 20, 30$, respectively. For simplicity, we assume that all links are bidirectional and symmetric. In our simulations, *MAX_ITERATION* is set to 200, *PCDS* is set to 0.9, network size ranges from 60 to 180, and learning rate changes from 0.06 to 0.6. The hosts are randomly distributed in a two-dimensional area of size 100×100 and 120×120, respectively, and the backbone is formed only on the connected graphs. The results shown in figures 6 to 15 are averaged over 100 connected graphs. In these figures, the points along the curves show the average value and the error bars represent 95% confidence interval. The simulation square area size, network size, and radio transmission range of hosts are three correlated parameters that affect the network backbone size. In this section, we study how these three parameters impact the backbone (CDS) size constructed by different algorithms.

Experiment I. The virtual backbone formation can be extensively used so as to design the (broadcast and multicast) routing protocols in wireless ad hoc networks in which the dominator hosts are responsible for relaying the messages as the intermediate hosts. Therefore, finding a virtual backbone including a small number of dominator hosts is the aim of the backbone formation algorithms. In this experiment, to study the impact of the learning rate of the proposed backbone formation method on the network backbone (CDS) size, the learning rate is initialized to 0.06 and the backbone size is measured as the network size (number of hosts in network) ranges from 60 to 180 with increment step of 20. The results are depicted in Figure 6. Then, the learning rate is varied to 0.08, 0.1, 0.2, 0.4 and 0.6 and the experiment is repeated. In these simulation studies, it is assumed that the radio transmission range for each host is 20, and a given number of hosts are randomly and uniformly distributed in a square simulation area of size 100×100 units. As shown in Figure 6, the network backbone size decreases as the learning rate of algorithm decreases. It can be seen that the backbone size rarely increases and remains unchanged in most cases as the network size increases. Note that the simulation results shown in Figures 6-15 are obtained as an average over the results of 100 connected graphs.
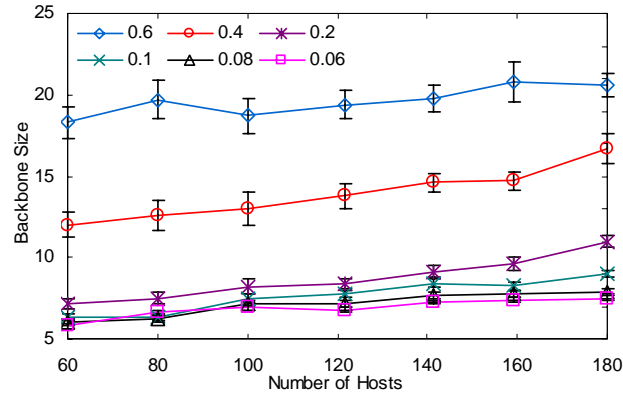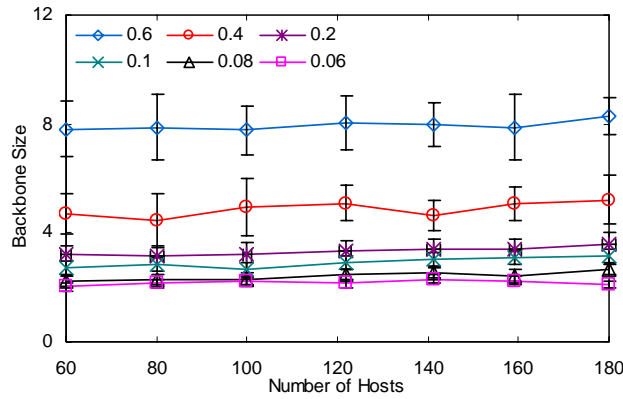
Figure 6. The average backbone size in proposed algorithm when the radio transmission range is 20 and the square area size is 100×100

EXPERIMENT II. To study the effect of the radio transmission range of host on the backbone size, in this experiment, we increased it to 30 and repeated the same studies as those in experiment I. The results are shown in Figure 7 and show that the backbone size decreases as the radio transmission range increases. Like Figure 6, the results shown in Figure 7 confirm that the backbone size decreases as the learning rate of algorithm decreases, and it slightly increases as the network size increases. From the error bars shown in Figures 6 and 7, we conclude that the variances in the results increases as the learning rate increases.

Figure 7. The average backbone size in proposed algorithm when the radio transmission range is 30 and the square area size is 100×100

EXPERIMENT III. From Theorem 2, it is clear that the message overhead of the proposed backbone formation algorithm is inversely proportional to the learning rate, and it logarithmically decreases as the learning rate increases. In this experiment, to examine the correctness of theorem 2, we have conducted some simulations to study the impact of the learning rate on the message overhead of algorithm. In this study, the message overhead of the proposed algorithm is measured (like the previous experiments) as the learning rate ranges from 0.06 to 0.6, the network size varies from 60 to 180, and the radio transmission range is set to 30 and 20, respectively. The results are shown in Figures 8 and 9. These results conform to those theoretically proved in Section 5, and show the message complexity significantly increases as the learning rate of algorithm decreases.
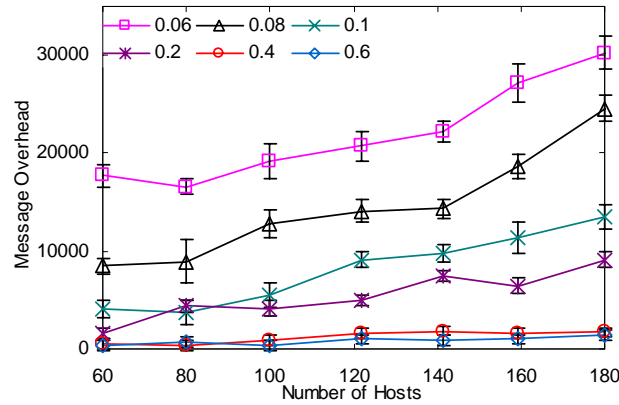
Figure 8. The message overhead of the proposed algorithm when the radio transmission range is 30 and the square area size is 100×100
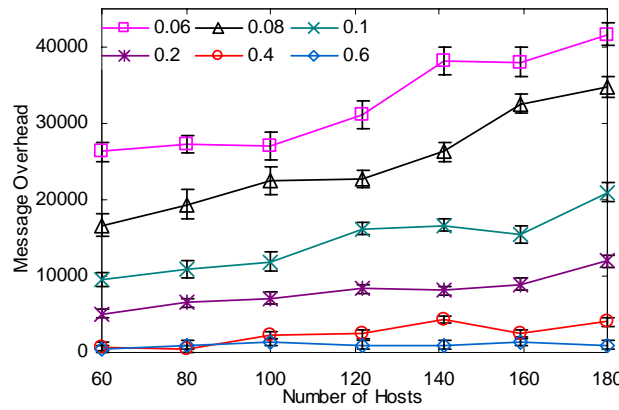

Figure 9. The message overhead of the proposed algorithm when the radio transmission range is 20 and the square area size is 100×100

Comparing the results given in Figures 6 and 7 with those given in Figures 8 and 9, we conclude that the message complexity is inversely proportional and the network backbone size is directly proportional to the learning rate. Therefore, to obtain a trade-off between the network backbone size and message overhead of algorithm, the learning rate must be carefully selected. Comparing the results given in Figures 6, 7, 8 and 9, we observe that a desired backbone size can be achieved in a reasonable message overhead when the learning rate is set to 0.2. Therefore, we choose it for further experiments.

In the second group of simulation studies, we compare the results of the proposed backbone formation algorithm with those of Butenko et al.'s algorithm [13], Li et al.'s algorithm [14], and Xie et al.'s algorithm [15] in terms of the network backbone size and message overhead. At each iteration of the simulation scenario which is used in our experiments, the initial dominator hosts is randomly chosen. Then it chooses the second dominator host according to its action probability vector. Each host forms its action set on occasion. This process continues until the network backbone to be formed. In all simulations, the constructed backbone is rewarded or penalized with a learning rate of 0.2. The backbone formation process terminates (or simulation stops) when the probability of choosing the constructed virtual backbone exceeds 0.90.

EXPERIMENT IV. In this experiment, the radio transmission range is set to 20, and the hosts are randomly and uniformly distributed in a square simulation area of size100×100 units. The proposed backbone formation algorithm is run and the network backbone (CDS) size is measured as the number of

hosts changes from 60 to 180 with increment step of 20. The results of different algorithms are shown in Figure 10. The results reveal that our proposed algorithm always significantly outperforms the other algorithms. Although the size of the backbone constructed by Xie et al.'s algorithm is much smaller than Butenko et al.'s algorithm and Li et al.'s algorithm, the size of the backbone constructed by Xie et al.'s algorithm is at least three times larger than that of our proposed algorithm.
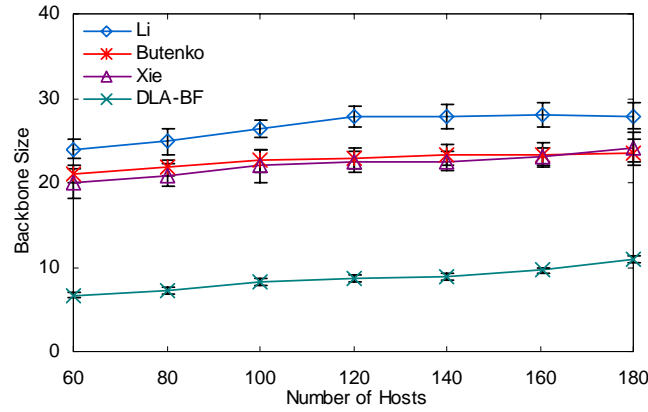

Figure 10. Comparison of the backbone size for the CDS-based backbone formation algorithms, when the radio transmission range is 20 and the square area size is 100×100

EXPERIMENT V. To investigate the impact of the radio transmission range on the backbone size of algorithms, in this experiment, we increase the radio transmission range of each host to 30 and the square area size remains unchanged. The same experiments are repeated as those in experiment V, and the backbone size of the given algorithms is measured as the network size changes from 60 to 180. The obtained results are depicted in Figure 11. Comparing the results given in Figure 11 with Figure 10, we observe that the size of the backbone constructed by all the studied algorithms becomes smaller when the radio transmission range of the hosts increases. The reason for this reduction is that a dominator host with a larger radio transmission range can cover more hosts and so the whole network can be covered by a less number of the dominators.
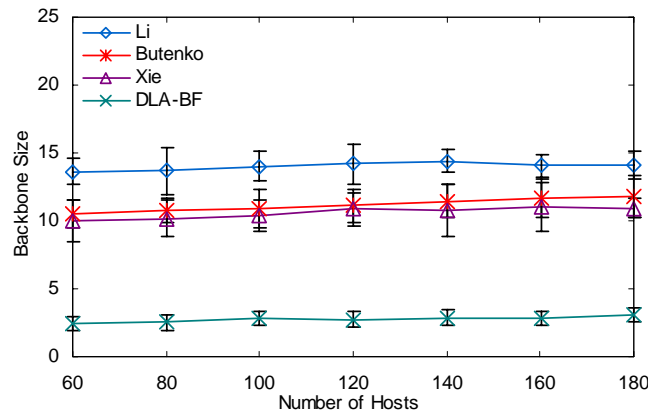

Figure 11. Comparison of the backbone size for the CDS-based backbone formation algorithms, when the radio transmission range is 30 and the square area size is 100×100

EXPERIMENT VI. In this experiment, we studied the impact of the square area size on the network backbone size. For this purpose, we increased the square area size to 120×120, and repeated the same experiments as those in experiments IV and V. The results were shown in Figures 12 and 13. Comparing the results given in Figures 10 and 11 with Figures 12 and 13, it is obvious that the size of the backbone

constructed by all the studied algorithms becomes larger as the square area size increases. This is reasonable because a dominator host can only cover the hosts which are within its radio transmission range, and so, a larger number of hosts must be chosen as dominators when the hosts are distributed in a larger area. Comparing the results shown in Figure 6 with Figure 10 and Figure 7 with Figure 11, we find that, even for the largest learning rate (i.e. a =0.6), the size of backbone constructed by our proposed algorithm is always smaller than that constructed by the best existing algorithms.
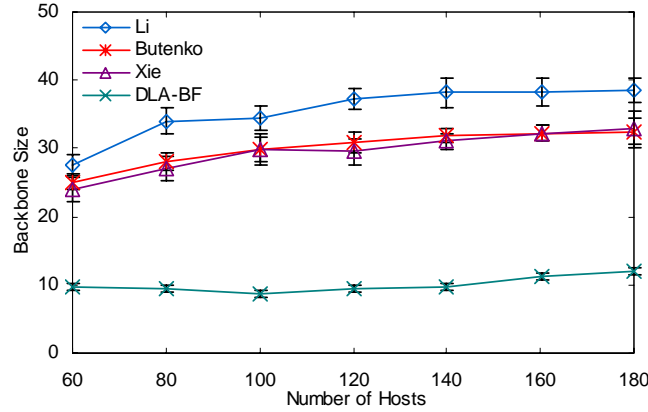

Figure 12. Comparison of the backbone size for the CDS-based backbone formation algorithms, when the radio transmission range is 20 and the square area size is 120×120
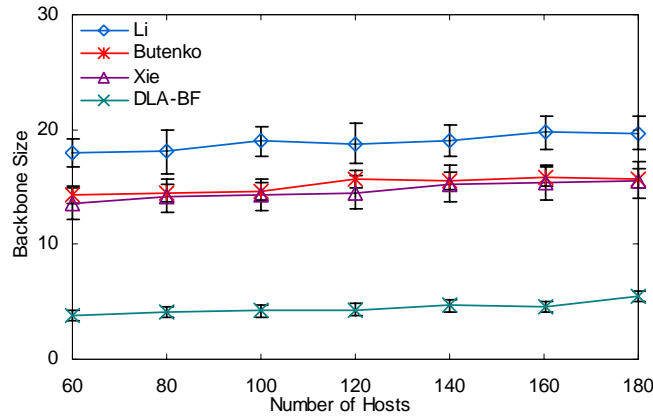

Figure 13. Comparison of the backbone size for the CDS-based backbone formation algorithms, when the radio transmission range is 30 and the square area size is 120×120

The results given in Figures 10-13 show the performance superiority of the proposed backbone formation algorithm over the other studied algorithms in terms of the network backbone size. As shown in these figures, in most cases, Xie et al.'s algorithm slightly outperforms Butenko et al.'s algorithm and so is ranked below our proposed algorithm. Furthermore, it can be seen that the size of the backbone constructed in Butenko et al.'s algorithm is much smaller than that constructed in Li et al.'s algorithm, and therefore Li et al.'s algorithm performs worst among all algorithms. For instance, in Figure 10, when the network size is 100, the average backbone size in Li et al.'s algorithm is 26.4, in Butenko et al.'s algorithm is 22.6, and in Xie et al.'s algorithm is 22, whereas in our algorithm it is 8.2.

EXPERIMENT VII. In this experiment, we compare the results obtained from our proposed algorithm with the results of Butenko et al.'s algorithm [13], Li et al.'s algorithm [14], and Xie et al.'s algorithm [15] in terms of the message overhead. In ad hoc networks, where the hosts suffer from the strict resource limitations, the communication overheads should be kept as low as possible. For this purpose, the learning rate must be chosen as large as possible. This leads to the large backbone size. Therefore, to obtain a trade-

off between the message complexity and backbone size, the learning rate should be carefully chosen. Based on the results obtained from the first group of experiments (i.e., Experiments I, II and III), we set it to 0.2. The simulation square area size is fixed at 100×100, and the radio transmission range of each host is set to 30. The message overhead of the above mentioned algorithms is measured as the number of hosts changes from 60 to 180, and shown in Figure 14. As expected, the results given in Figure 14 show that Xie et al.'s algorithm outperforms all the others in terms of the message overhead, and Butenko et al.'s algorithm has the most message overhead. The message overhead of our proposed algorithm is slightly more than Li et al.'s algorithm [14], and so our proposed algorithm is ranked the third least cost algorithm for the message overhead. Comparing the results given in Figure 8 with Figure 14 and Figure 6 with Figure 10, we observe that choosing the larger learning rates for our proposed algorithm leads to a less message overhead than the other algorithms despite the superiority of our algorithm in terms of the backbone size. That is, choosing the learning rates larger than 0.2, the proposed algorithm outperforms the others both in terms of the backbone size and message overhead. However, the aim of choosing learning rate 0.2, in our experiments, is to achieve a solution as close to the optimal as possible in a reasonable message overhead for ad hoc networks.

To study the effect of the radio transmission range of host on the message complexity of algorithms, we decreased it to 30 and repeated the same experiments. The obtained results are given in Figure 15. As shown in this figure, the algorithms have the same order as those in Figure 14. Xie et al.'s algorithm is always superior to the other algorithms. Li et al.'s algorithm is ranked below Xie et al.'s algorithm, though it only slightly outperforms our proposed algorithm. Butenko et al.'s algorithm lags far behind our algorithm.
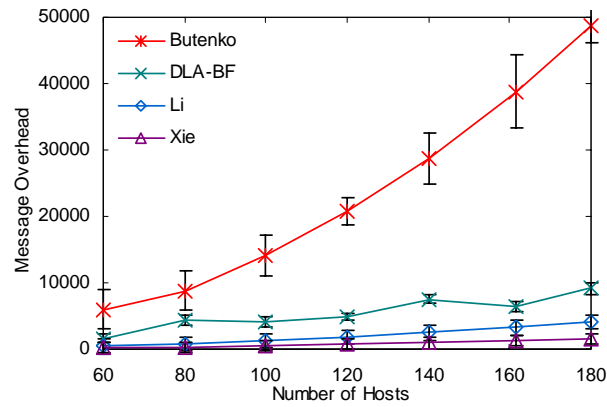

Figure 14. Comparison of the message overhead for the CDS-based backbone formation algorithms, when the radio transmission range is 30 and the square area size is 100×100
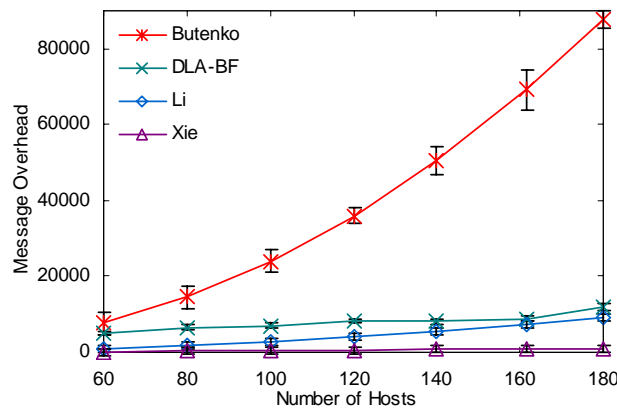

Figure 15. Comparison of the message overhead for the CDS-based backbone formation algorithms, when the radio transmission range is 20 and the square area size is 100×100

7. Conclusion

In this paper, we proposed a DLA-based backbone formation algorithm in which by finding a near optimal solution to the minimum CDS problem, a minimum size network backbone is formed. Reducing the rebroadcasts due to sending the messages along the virtual backbone alleviates the notorious broadcast storm problem in ad hoc networks. The proposed algorithm could be also used in multicast routing protocols, where the only multicast group members need to be dominated by the CDS. We computed the worst case running time and message complexity of the proposed backbone formation algorithm to find a $1/(1-\varepsilon)$ optimal size backbone and showed that by a proper choice of the learning rate of the proposed algorithm, a trade-off between the running time and message complexities of algorithm with the backbone size could be made. We compared the results of our proposed algorithm with those of the best-known CDS-based backbone formation algorithms and showed that our algorithm always outperforms the others in terms of the backbone (CDS) size. Although Xie et al's algorithm has the least message overhead, its backbone size is much larger than Butenko et al.'s algorithm, and Butenko et al.'s algorithm has the most message overhead. The experimental results showed that our proposed algorithm significantly outperforms Butenko et al.'s algorithm in terms of the backbone size, while its message overhead is slightly more than Xie et al's algorithm.

References

[1] Y. P. Chen, A. L. Liestman, "Maintaining Weakly-Connected Dominating Sets for Clustering Ad Hoc Networks," *Ad Hoc Networks*, Vol. 3, pp. 629–642, 2005.

[2] P. Gupta, P.R. Kumar, "The Capacity of Wireless Networks," *IEEE Transaction on Information Theory*, Vol. 46, No. 2, pp. 388–404, 2000.

[3] B. Han, W. Jia, "Clustering Wireless Ad Hoc Networks with Weakly Connected Dominating Set," *Journal of Parallel and Distributed Computing*, Vol. 67, pp. 727 – 737, 2007.

[4] R. Rajaraman, "Topology Control and Routing in Ad Hoc Networks: A Survey," *SIGACT News*, Vol. 33, No. 2, pp. 60–73, 2002.

[5] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit Disk Graphs," *Discrete Mathematics*, Vol. 86, pp. 165-177. 1990.

[6] M.V. Marathe, H. Breu, H.B. Hunt III, S.S. Ravi, D.J. Rosenkrantz, "Simple Heuristics for Unit Disk Graphs," *Networks*, Vol. 25, pp. 59–68, 1995.

[7] Y.Z. Chen, A.L. Liestman, "Approximating Minimum Size Weakly Connected Dominating Sets for Clustering Mobile Ad hoc Networks," *Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'2002)*, pp. 157–164, 2002.

[8] Y.P. Chen, A.L. Liestman, "A Zonal Algorithm for Clustering Ad Hoc Networks," *International Journal of Foundations of Computer Science*, Vol. 14, No. 2, pp. 305–322, 2003.

[9] K.M. Alzoubi, P.-J. Wan, O. Frieder, "Maximal Independent Set, Weakly Connected Dominating Set, and Induced Spanners for Mobile Ad Hoc Networks," *International Journal of Foundations of Computer Science*, Vol. 14, No. 2, pp. 287-303, 2003.

[10] P. J. Wan, K. Alzoubi, O. Frieder, "Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks," in Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002), Vol. 3, pp. 1597–1604. 2002.

[11] J. Wu, B. Wu, and I. Stojmenovic, "Power-Aware Broadcasting and Activity Scheduling in Ad Hoc Wireless Networks Using Connected Dominating Sets," *Journal of Wireless Communications and Mobile Computing*, Vol. 3, pp. 425-438, 2003.

[12] O. Dousse, F. Baccelli, and P. Thiran, "Impact of Interferences on Connectivity in Ad hoc Networks," *IEEE/ACM Transactions on Networking*, Vol. 13, No. 2, pp. 425-436, 2005.

[13] S. Butenko, X. Cheng, C. Oliveira, P. M. Pardalos, "A New Heuristic for the Minimum Connected Dominating Set Problem on Ad Hoc Wireless Networks," *In Recent Developments in Cooperative Control and Optimization, Kluwer Academic Publishers*, pp. 61–73, 2004.

[14] Y. Li, M. T. Thai, F. Wang, C. W. Yi, P. J. Wang, D. Z. Du, "On Greedy Construction of Connected Dominating Sets in Wireless Networks," *Special issue of Wireless Communications and Mobile Computing (WCMC)*, 2005.

[15] R. Xie, D. Qi, Y. Li, and J. Z. Wang, "A Novel Distributed MCDS Approximation Algorithm for Wireless Sensor Networks," *Journal of Wireless Communications and Mobile Computing*, 2007.

[16] K. S. Narendra and K. S. Thathachar, "Learning Automata: An Introduction," New York, *Printice-Hall*, 1989.

[17] M. A. L. Thathachat, P. S. Sastry, "A Hierarchical System of Learning Automata That Can Learn the Globally Optimal Path," *Information Science*, Vol.42, pp.743-766, 1997.

[18] M. A. L. Thathachar and B. R. Harita, "Learning Automata with Changing Number of Actions," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMG17, pp. 1095-1100, 1987.

[19] M. A. L. Thathachat, V.V.Phansalkar, "Convergence of Teams and Hierarchies of Learning Automata in Connectionist Systems," *IEEE Transactions on Systems, Man and Cybernetics*, 1995, Vol. 24, pp. 1459-1469.

[20] S. Lakshmivarahan and M. A. L. Thathachar, "Bounds on the Convergence Probabilities of Learning Automata," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-6, pp. 756-763, 1976.

[21] K. S. Narendra, and M. A. L. Thathachar, "On the Behavior of a Learning Automaton in a Changing Environment with Application to Telephone Traffic Routing," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-l0, No. 5, pp. 262-269, 1980.

[22] H. Beigy, M. R. Meybodi, "Utilizing Distributed Learning Automata to Solve Stochastic Shortest Path Problems," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol.14, pp. 591-615, 2006.

[23] S. Basagni, D. Bruschi, I. Chlamtac, "A Mobility-Transparent Deterministic Broadcast Mechanism for Ad Hoc Network," *IEEE/ACM Transactions on Networking*, Vol. 7, No. 6, pp-799-807, 1999.

[24] I. Chlamtac, S. Kutten, "Tree-Based Broadcasting in Multihop Radio Networks," *IEEE Transactions on Computing*, Vol. 36, No. 10, pp. 1209-1223, 1987.

[25] H. Lim, and C. Kim, "Flooding in Wireless Ad Hoc Networks," *Journal of Computer Communications*, Vol. 24, pp. 353-363, 2001.

[26] X. Cheng, M. Ding, D. Hongwei, and X. Jia, "Virtual Backbone Construction in Multihop Ad Hoc Wireless Networks," *Journal of Wireless Communications and Mobile Computing*,Vol. 6, pp. 183-190, 2006.

[27] S. Guha and S. Khuller, "Approximation Algorithms for Connected Dominating Sets," *Algorithmica*, Vol. 20, No. 4, pp. 374-387, 1998.

[28] K.M. Alzoubi, P.J. Wan and O. Frieder, "Distributed Heuristics for Connected Dominating Sets in Wireless Ad Hoc Networks," *Journal of Communications and Networks*, Vol. 4, No. 1, pp. 22-29, 2002.

[29] F. Dai, and J. Wu, "An Extended Localized Algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks," *to appear in IEEE Transactions on Parallel and Distributed Systems*, 2004.

[30] J. Wu, and H. Li, "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks," *in Proceedings of the Third ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (ACM DIALM'1999)*, pp. 7–14, 1999.

[31] K. M. Alzoubi, X. Y. Li, Y. Wang, P. J. Wan, and O. Frieder, "Geometric Spanners for Wireless Ad Hoc Network," *IEEE Transactions on Parallel and Distributed Systems,* Vol. 14, No. 4, pp. 408–421, 2003.

[32] S. Basagni, M. Mastrogiovanni, C. Petrioli, "A Performance Comparison of Protocols for Clustering and Backbone Formation in Large Scale Ad Hoc Network," *Proceedings of the First IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS'2004)*, pp. 70–79, 2004.

[33] J. Wu, F. Dai, M. Gao, and I. Stojmenovic "On Calculating Power-Aware Connected Dominating Sets for Efficient Routing in Ad Hoc Wireless Networks," *Journal of Communications and Networks*, Vol.4, No.1, 2002.

[34] M. Haleem and R. Chandramouli, "Adaptive Downlink Scheduling and Rate Selection: A Cross Layer Design," Special issue on Mobile Computing and Networking, *IEEE Journal on Selected Areas in Communications*, Vol. 23, No. 6, 2005.

[35] P. Nicopolitidis, G. I. Papadimitriou and A. S. Pomportsis, "Exploiting Locality of Demand to Improve the Performance of Wireless Data Broadcasting," *IEEE Transactions on Vehicular Technology*, Vol. 55, No. 4, pp. 1347-1361, 2006.

[36] P. Nicopolitidis, G. I. Papadimitriou and A. S. Pomportsis, "Learning-Automata-Based Polling Protocols for Wireless LANs," *IEEE Transactions on Communications*, Vol. 51, No. 3, pp. 453-463, March 2003.

[37] P. Nicopolitidis, G. I. Papadimitriou and A. S. Pomportsis, "Distributed Protocols for Ad-Hoc Wireless LANs: A Learning-Automata-Based Approach," *Ad Hoc Networks*, Vol. 2, No. 4, pp. 419-431, 2004.

[38] P. Nicopolitidis, G. I. Papadimitriou, M. S. Obaidat and A. S. Pomportsis, "Carrier-sense-assisted Adaptive Learning MAC Protocol for Distributed Wireless LANs," *International Journal of Communication Systems*, Wiley, Vol. 18, No. 7, pp. 657-669, 2005.

[39] B. V. Ramana and C. S. R. Murthy, "Learning-TCP: A Novel Learning Automata Based Congestion Window Updating Mechanism for Ad hoc Wireless Networks," *12th IEEE International Conference on High 13 Performance Computing*, pp. 454-464, 2005.

[40] H. Beigy and M. R. Meybodi, "Learning Automata-based Dynamic Guard Channel Algorithms," *Journal of High Speed Networks*, 2008, to appear.

[41] H. Beigy and M. R. Meybodi, "A General Call Admission Policy for Next Generation Wireless Networks," *Computer Communications*, Vol. 28, pp. 1798–1813, 2005.

[42] H. Beigy, and M.R. Meybodi, "An Adaptive Call Admission Algorithm for Cellular Networks," *Computers and Electrical Engineering*, Vol. 31, pp. 132–151, 2005.

[43] H. Beigy, and M. R. Meybodi, "A Learning Automata-based Dynamic Guard Channel Scheme," *Lecture Notes on Information and Communication Technology*, Vol. 2510 Springer Verlag, pp. 643-650, 2002.

[44] M. Golipour and M. R. Meybodi, "LA-Mobicast: A Learning Automata based Mobicast Routing Protocol for Wireless Sensor Networks," *Sensor Letters,* Vol. 6, No.2, pp. 305-311, 2008.

[45] M. Esnaashari and M. R. Meybodi, "A Cellular Learning Automata based Backbone formation algorithm for Wireless Sensor Networks," *Sensor Letters*, Vol. 6, pp.1-13, 2008.

[46] M. Esnaashari and M. R. Meybodi, "Dynamic Point Coverage in Wireless Sensor Networks: A Learning Automata approach," *Lecture Notes in Computer Science Springer Verlag*, Kish Island, Iran, pp. 758-762, 2008.

Javad Akbari Torkestani received the B.S. and M.S. degrees in Computer Engineering from Arak University in Iran, in 2001 and 2004, respectively. He is currently pursuing the Ph.D. degree in Computer Engineering at Science and Research University, Tehran, Iran. He joined the faculty of Computer Engineering Department at Arak Azad University, Arak, Iran, in 2004. His research interests include wireless networks, mobile ad hoc networks, fault tolerant systems, learning systems, parallel algorithms, and soft computing.

Mohammad Reza Meybodi received the B.S. and M.S. degrees in Economics from Shahid Beheshti University in Iran, in 1973 and 1977, respectively. He also received the M.S. and Ph.D. degree from Oklahoma University, USA, in 1980 and 1983, respectively in Computer Science. Currently, he is a full professor in Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an assistant professor at Western Michigan University, and from 1985 to 1991 as an associate professor at Ohio University, USA. His research interests include wireless networks, fault tolerant systems, learning systems, parallel algorithms, soft computing and software development.