Wireless Pers Commun
DOI 10.1007/s11277-010-9936-4

Weighted Steiner Connected Dominating Set and its Application to Multicast Routing in Wireless MANETs

Javad Akbari Torkestani · Mohammad Reza Meybodi

Abstract In this paper, we first propose three centralized learning automata-based heuristic algorithms for approximating a near optimal solution to the minimum weight Steiner connected dominating set (WSCDS) problem. Finding the Steiner connected dominating set of the network graph is a promising approach for multicast routing in wireless ad-hoc networks. Therefore, we present a distributed implementation of the last approximation algorithm proposed in this paper (Algorithm III) for multicast routing in wireless mobile ad-hoc networks. The proposed WSCDS algorithms are compared with the well-known existing algorithms and the obtained results show that Algorithm III outperforms the others both in terms of the dominating set size and running time. Our simulation experiments also show the superiority of the proposed multicast routing algorithm over the best previous methods in terms of the packet delivery ratio, multicast route lifetime, and end-to-end delay.

Keywords Steiner connected dominating set · Multicast routing · Learning automata · Wireless mobile Ad-hoc networks

1 Introduction

Dynamic network topology changes, resource constraints (e.g., bandwidth and power limitations), lack of the fixed infrastructures and centralized administrations result in the optimal

J. Akbari Torkestani (✉)
Department of Computer Engineering, Islamic Azad University, Arak Branch, Arak, Iran
e-mail: j-akbari@iau-arak.ac.ir

M. R. Meybodi
Department of Computer Engineering and IT, Amirkabir University of Technology, Tehran, Iran
e-mail: mmeybodi@aut.ac.ir

M. R. Meybodi
School of Computer Science, Institute for Studies in Theoretical Physics and Mathematics (IPM), Tehran, Iran

multicast routing becomes the most challenging problem in wireless mobile ad hoc networks. A wireless mobile ad-hoc network (MANET) is a self-organizing and self-configuring multi hop wireless communication network supporting a collection of the mobile hosts. There is neither a fixed infrastructure nor a central administration in these networks, and mobiles can form a temporary network infrastructure in an ad-hoc fashion. In ad hoc networks, two hosts can directly communicate when they are within transmission range of each other, and indirectly through relaying by the intermediate hosts. In such environments, each host assumes the role of a router to relay the packets toward the final destinations, and so the messages are forwarded through the multi hop communications, if the source and destination are not within the transmission range of each other. In addition to the multi-hop nature of the wireless ad-hoc networks and the lack of a fixed infrastructure, these networks inherit the traditional problems of the wireless and mobile communications. Frequent and hard to predict topology changes are the most important issues that must be taken into consideration in mobile networking.

Multicast routing is an effective way to facilitate the group communications in which the messages need to be sent from a transmitting node to multiple receivers. Due to the broadcast nature of the wireless channels, a single transmission can be received by all neighbors of the transmitting node [1]. Therefore, the multicast routing in wireless ad-hoc networks significantly differs from the traditional multicasting in wired networks. In wired networks, the multicast packets are forwarded along the tree edges, and so the multicast routing problem can be defined as a Steiner tree problem (ST), where the multicast group members are the terminals (leaf nodes) in the Steiner tree. While in wireless networks, owing to the broadcast nature of the omnidirectional antennae, the Steiner connected dominating set (SCDS) problem [2] is a promising approach for modeling the multicast routing problem, where the only multicast group members need to be dominated. Due to the fundamental natural differences, the protocols designed for multicast routing in traditional wired networks can not be applied to the wireless mobile ad-hoc networks.

The minimum Steiner connected dominating set problem was first proposed by Guha and Khuller [2], as the generalization of the well-known minimum connected dominating set problem, where only a specified subset of the nodes should be dominated. Using the Steiner connected dominating set for modeling the multicast routing in ad-hoc networks was also proposed by Wu et al. [3] for the first time. They proposed a method in which taking advantage of the Steiner connected dominating set a virtual multicast backbone (VMB) is formed along which the multicast messages can be sent. The proposed method aims at minimizing the number of nodes which are responsible for relaying the multicast packets. In SCDS-based multicast routing protocols, a subset of the nodes is chosen as dominators to construct a route from the multicast source to each of the multicast receivers. That is, the Steiner connected dominating set forms a virtual backbone by which not only all the multicast group members are dominated but also the number of hosts responsible for broadcasting is reduced to the number of hosts in the backbone. Nevertheless, a few SCDS-based algorithms have been designed for wireless ad-hoc networks, and the notorious global flooding method [4], in which all hosts broadcast the multicast messages, is still a common approach in most of the multicast routing protocols.

In this paper, three centralized approximation algorithms are first proposed for solving the minimum weight Steiner connected dominating set problem proposed by Guha and Khuller. In these algorithms, it is assumed that a random weight is associated with each node. Each proposed algorithm consists of a number of stages, and at each stage, a subset of the nodes is randomly chosen as a Steiner connected dominating set. The weight of the selected Steiner connected dominating set is evaluated through the random environment, and depending on the response received from the environment, the selected Steiner connected dominating set is rewarded or penalized. After a number of iterations, the algorithm learns how to find the Steiner connected dominating set with the minimum weight. The proposed centralized algorithms are compared with each other, and the obtained results show that the third proposed algorithm outperforms the others both in terms of the Steiner connected dominating set size and running time of algorithm. Then, this algorithm is compared with the well-known previous algorithms and the results show its superiority over the others. Due to the lack of the central administration in mobile ad-hoc networks, the proposed centralized algorithms are not applicable to such environments. Therefore, in this paper, a distributed version of the last proposed centralized algorithm (*Algorithm III*) is presented for multicast routing in wireless mobile ad-hoc networks. The proposed multicast routing algorithm is also compared with the best multicasting methods, and the simulation experiments show that the proposed multicast routing algorithm is superior to the existing methods in terms of the packet delivery ratio, multicast route lifetime, and end-to-end delay.

The rest of the paper is organized as follows. The next section describes related work on Steiner connected dominating set, multicast routing in wireless mobile ad-hoc networks, and learning automata. Section 3 summarizes some preliminaries on the dominating set and learning automata. The centralized Steiner connected dominating set algorithms are proposed in Sect. 4, and the proposed multicast routing algorithm is presented in Sect. 5. Section 6 shows the efficiency of the proposed algorithms through the simulation experiments, and Sect. 7 concludes the paper.

2 Related Work

In this section, we give a brief overview of the existing literature on the Steiner connected dominating problem, mobility-based multicast routing algorithms in ad-hoc networks, and applications of learning automata in a variety of network environments.

2.1 Steiner Connected Dominating Set

Guha and Khuller [2] first introduced the Steiner connected dominating set problem as a generalization of the CDS problem, where only a specified subset of the nodes, $R \subseteq V$, should be dominated. They also showed that the SCDS is an NP-hard problem in general graphs and even in unit disk graphs (UDG). The centralized algorithm proposed by Guha and Khuller [2] is a polynomial time greedy algorithm with approximation ratio $[(c+1)\cdot H(\delta)+c-1]\cdot \mathrm{OPT}$, where c is the Steiner approximation ratio for graphs (currently, $c \approx 1.55$ [5]),H is the harmonic function, $\delta \leq \min(\Delta, |R|)$is the size of the largest subset of R, adjacent to a vertex in the graph, Δ is the maximum node degree, and OPT denotes the optimal solution to the Steiner connected dominating set problem. The time complexity of the proposed algorithm is at most $O(|V|^2)$, where $|V|$denotes the number of vertices in the graph. Wu et al. [3] proposed two approximation algorithms based on maximal independent set (MIS) for solving the minimum SCDS (MSCDS) problem. Their former algorithm is a one-hop method for approximating the MSCDS of a unit disk graph with a constant approximation ratio at most 10. This algorithm exploits the properties of the MIS and minimum ST to form the MSCDS. The proposed algorithm first finds the MIS of the graph induced by only the vertices in R. Then, the Steiner tree algorithm proposed in [6] is applied to connect the vertices of the constructed MIS. The size of the SCDS constructed by the proposed one-hop algorithm is at most $10 \times \mathrm{OPT} + 1$. The time complexity of this algorithm is $O(D \cdot |V|)$, where D denotes the graph diameter. They also proposed a d-hop algorithm in which a d-hop graph is initially constructed, where the terminals form the vertex-set of the graph. In this method, every two vertices of the graph are connected by an edge, if they are d-hop neighbors. Now, like the one-hop algorithm, an MIS of the d-hop graph is computed, and than a Steiner tree algorithm [6] is applied to connect the vertices of MIS. The d-hop algorithm computes a SCDS of subset R, whose size is at most $(8d^2 + 8d + 2 + 4/d) \cdot \text{OPT} - 1$, if d is even, and $[(16d^3+16d^2+4d+6)\cdot\text{OPT}+4d^3+4d^2-2d+3]/(2d-1)$, if is $d$odd. The d-hop algorithm can be implemented in a fully distributed manner, and the message and time complexity of the distributed d-hop algorithm are $O(|V|^2)$ and $O(D \cdot |V|)$, respectively. They showed the distributed d-hop algorithm can be effectively used for multicast routing in ad-hoc networks by constructing a VMB with a small number of forwarding nodes. Muhammad [7,8] also proposed a distributed MIS-based Steiner connected dominating set algorithm for multicast routing in wireless ad-hoc networks. The first step of this algorithm constructs a maximal independent set in $O(|V|)$time, whose size is at most $|MIS| = 4 \times \text{OPT} + 1$. The second step uses a distributed Steiner tree with message complexity $O(2\,|V|)$ and time complexity $O\ ((|V| - 1) \cdot |E| \cdot \log |V|)$ for connecting the MIS members. The total size of the SCDS constructed by Muhammad's algorithm is smaller than $2(|MIS|+\text{OPT}-1)+1$, and the message and time complexity of this algorithm are $O(2\,|V|)$ and $O\ ((|V| - 1) \cdot |E| \cdot \log |V|)$, respectively. Aggarwal et al. [9] proposed an algorithm for approximating the MSCDS in a dominating pair graph. The time complexity of the proposed algorithm is $O(|V|^8 \cdot |R|)$, where $|V|$and $|R|$denotes the cardinality of the vertex set and terminal-set, respectively. The authors show that the proposed algorithm computes the Steiner connected dominating set in $O(|V|^4 \cdot |R|)$time, if the distance between the dominating pair vertices is greater than 8. The cardinality of the Steiner connected dominating set constructed by Aggarwal et al.'s algorithm is smaller than $\text{OPT} + 2dt(G)$, where $dt(G)$is the dominating target number of graph G.

2.2 Multicast Routing in Ad-Hoc Networks

ODMRP [10] applies on-demand routing techniques to avoid channel overhead and improve scalability. It uses the concept of forwarding group [11], a set of nodes which is responsible for forwarding multicast data on the shortest paths between any member pairs to build a forwarding mesh for each multicast group. By maintaining and using a mesh, ODMRP avoids drawbacks of multicast trees in mobile wireless networks (for example, intermittent connectivity, traffic concentration, frequent tree reconfiguration, non-shortest path in a shared tree). ODMRP is a reactive (on-demand) protocol that delivers packets to destination(s) on a mesh topology using scoped flooding of data. ODMRP takes a soft-state approach to maintain multicast group members. No explicit control message transmission is required to leave the group. ODMRP establishes and maintains group membership and multicast routes by the source on demand. The major strengths of ODMRP are its simplicity and scalability. Su et al. [12] proposed two mobility prediction mechanisms for mobile ad-hoc networks. The former mobility prediction method utilizes the location and mobility information provided by the global positioning system (GPS) to estimate the link expiration time. In this method, the time interval during which two neighboring hosts remain within the transmission range of each other, is determined based on their mobility information (speed and direction) and radio range transmission. Since GPS may not work properly in certain situations, we may not always be able to accurately predict the link expiration time for a particular link. Therefore, Su et al. [12] proposed an alternative method to predict the link expiration time based on a more realistic propagation model. In this method, the transmission power samples are measured periodically from packets received from a neighboring host. Based on this information, the mobile can compute the rate of change for a particular neighbor's transmission power level. Therefore, the time that the transmission power level drops below the acceptable value can be computed. They applied the proposed mobility prediction method to ODMRP and showed it superiorities over ODMRP. An and Papavassiliou [13] proposed a mobility-based hybrid multicast routing protocol for mobile ad-hoc wireless networks. In mobile ad-hoc networks, communications are often among teams that tend to coordinate their movements. Therefore, the relative mobility of each host with respect to its peers is the mobility metric upon which the proposed multicast routing algorithm is based. In this method, the network is dynamically and adaptively partitioned into several groups, each with its own mobility behaviors. Then, a group-based hierarchical multicast routing is supported in each group. Guo and Yang [14] proposed two distributed multicast routing algorithms for achieving the maximum-lifetime in mobile ad-hoc networks. The former algorithm is a basic energy efficient multicast routing algorithm called BEEM which can construct and maintain a multicast tree in a distributed manner. It uses beaconing to allow periodical transmission power adjustment to the minimal level at each transmitting node in the tree such that it could significantly save energy compared to those multicast algorithms for mobile ad-hoc networks which only apply single level of transmission power. They also proposed a distributed maximum lifetime multicast routing algorithm called DMLM, in which an extra localized operation called lifetime enhancement operation is used to prolong the tree lifetime. In this method, the hosts make decisions based solely on the mobility information of and distances to its neighbors.

2.3 Learning Automata Applications

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 [15] 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. [16] 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 [17] 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. [18] 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. [19] 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 [20] 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 [21–24]. Beigy and Meybodi [21] 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 [21] they also introduced a new model for non-stationary environments under which the proposed algorithms work. They [22] 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 [23] 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.

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 environments. It has been shown in [36,45,46] that the learning automata are capable of solving the NP-hard problems. Recently, several learning automata-based protocols have been also designed for improving the performance of the wireless ad hoc networks [25–27,45]. For instance in [45], Akbari Torkestani and Meybodi proposed a learning automata-based cluster formation algorithm for ad hoc networks. In comparison with the best existing clustering methods, the proposed algorithm reduces the number of cluster-heads as well as the message overhead. In [25], the same authors proposed a CDMA/TDMA channel assignment scheme for mobile ad hoc networks. In the proposed channel assignment scheme, each host is assigned a fraction of TDMA frame proportional to its needs (traffic load). In [26], Akbari Torkestani and Meybodi also proposed a backbone formation algorithm based on distributed learning automata. By the proposed algorithm, the overheads of the multicast routing and broadcasting in wireless ad hoc networks can be significantly reduced. This is due to the fact that the proposed algorithm decreases the number of hosts which are responsible for relaying the message to the number of hosts in backbone.

3 Preliminaries

In this section, to provide a sufficient background for the remainder of the paper, we present a brief overview of the dominating set problems and their applications in communication networks, as well as some preliminaries on learning automata theory and variations of learning automata.

3.1 Dominating Set

The dominating set problems are a class of the combinatorial optimization problems in graph theory which are widely used in wireless ad-hoc networks [28–32]. In wireless networks, message broadcasting, network clustering, multicast routing, and network backbone formation are some networking issues in which the dominating set plays an important role. In what follows, the different optimization problems dealing with the domination sets, and their applications in wireless ad-hoc networks are summarized.

A dominating set S of graph $G = (V, E)$ is a subset of V, such that every vertex $v \in V$ is either in S or adjacent to a vertex of S. A vertex of S is said to dominate itself and all adjacent vertices. Finding the dominating set is a well-known approach, proposed for clustering the wireless ad-hoc networks [32]. A minimum DS (MDS) is a DS with the minimum cardinality. A dominating set is also an independent dominating set, if no two vertices in the set are adjacent.

A connected dominating set S of a given graph G is a dominating set whose induced sub-graph, denoted by $<S>$, is connected, and a minimum CDS (MCDS) is a CDS with the minimum cardinality. A MCDS forms a virtual backbone in the network 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 vertices in the MCDS of the network topology graph. Finding the MCDS is a promising approach to send the broadcast messages [28,30]. The MDS and MCDS problems have been shown to be NP-Hard [33,34], and even for a unit disk graph, the problem of finding a MCDS is still NP-Hard [34].

A weakly connected dominating set (WCDS) S of a given graph G is a dominating set of G, where the graph $< S >_W = (N[S], E \cap (N[S] \times S))$ is a connected sub-graph of G. The closed neighborhood of a given host v, $N_G[v]$, consists of the hosts adjacent to$v$ and host v itself, and closed neighborhood of set S, $N_G[S]$, is the union $\bigcup_{v \in S} N_G[v]$. That is, the weakly induced sub-graph $< S >_W$contains the hosts of S, their neighbors, and all edges with at least one endpoint in S. Finding the WCDS was first suggested for clustering the wireless networks by Chen and Listman [35].

The Steiner connected dominating set S of a given graph G is a connected dominating set by which only a given subset $R \subseteq V$ (hereafter referred to as terminal-set) must be dominated. Each member of this subset is referred to as a terminal. Indeed, in SCDS problem, a specified subset, R, of the vertices has to be dominated by the a connected dominating set. Finding the SCDS of the network graph is a well-known approach proposed for solving the multicast routing problem in wireless ad-hoc networks [2,3,6–9], where subset R comprises the multicast source and the multicast receivers. In this method, the SCDS includes the intermediate nodes by which the massage sent out by the multicast source is relayed.

In most of the multicast routing protocols, it is assumed that all the nodes have the same weights, and so these protocols try to minimize the number of forwarding nodes for optimizing the multicast routes. In these methods, the multicast routing problem is defined as finding the connected dominating set with the minimum cardinality. However, in many applications of the wireless ad-hoc networks, the assumption above can not hold true, and reducing the number of forwarding nodes is not sufficient. In such networks, due to the host heterogeneity, host mobility, and strict resource limitations, each wireless host may have a different cost in the multicast tree. Therefore, in the following, we present the concept of the weighted Steiner connected dominating set. The (node)-weighted Steiner connected dominating set problem (WSCDS) is the generalization of the Steiner connected dominating set problem to the case where the vertices have weight, and the minimum WSCDS problem aims at finding the Steiner connected dominating set with a minimum weight. The weight of a set is assumed to be sum of the weight of the elements contained in it.

3.2 Learning Automata

A learning automaton [36–43] 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.

The environment can be described by a triple $E \equiv \{\alpha, \beta, c\}$, where $\alpha \equiv \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents the finite set of the inputs, $\beta \equiv \{\beta_1, \beta_2, \ldots, \beta_m\}$ denotes the set of the values can be taken by the reinforcement signal, and $c \equiv \{c_1, c_2, \ldots, c_r\}$ denotes the set of the penalty probabilities, where the element c_i is associated with the given action α_i. 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 $\underline{\beta}$ 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 $[a, b]$.

Learning automata can be classified into two main families [37]: 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.

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

when the taken action is rewarded by the environment (i.e. $\beta(n) = 0$) and

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

When the taken action is penalized by the environment (i.e. $\beta(n) = 1$). $r$is the number of actions which 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 Eqs. (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 multicast routing algorithm presented in this paper, each learning automaton uses a linear reward-inaction learning algorithm to update its action probability vector.

3.3 Distributed Learning Automata

A learning automaton is by design a simple unit by which simple things can be done. The full potential of the learning automata will be realized when a cooperative effort is made by a set of interconnected learning automata to achieve the group synergy. A Distributed learning automata (DLA) [36] is a network of interconnected 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 $A = \{A_1, \ldots, A_n\}$ 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.

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.

3.4 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. Variable action-set learning automata have been found to be useful in many applications. For instance, Akbari Torkestani and Meybodi employed variable action-set learning automata for solving vertex coloring problem [46], connected dominating set problem [26,36,45], and stochastic minimum spanning tree problem [43]. It has been shown in [36,39,43] 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, $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$. $A = \{A_1, A_2, \ldots, A_m\}$ denotes the set of action subsets and $A(k) \subseteq \alpha$ is the subset of all the actions that 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 $q(k) = \{q_1(k), q_2(k), \ldots, q_m(k)\}$ defined over the possible subsets of the actions, where $q_i(k) = prob[A(k) = A_i | A_i \in A, 1 \le i \le 2^n - 1]$. $\hat{p}_i(k) = prob\,[\alpha(k) = \alpha_i | A(k), \alpha_i \in A(k)]$ 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

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

where $K(k) = \sum_{\alpha_i \in A(k)} p_i(k)$ 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 Eq. (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 $p_i(k+1) = \hat{p}_i(k+1) \cdot K(k)$, for all $\alpha_i \in A(k)$. The absolute expediency and ε-optimality of the method described above have been proved in [39].

```
Algorithm I The First SCDS Formation Algorithm
Step 1-    <SCDS FORMATION>
    For all learning automata do in parallel
        Each automaton $A_i$ chooses one of its two actions according to its action probability vector.
        If (The chosen action declares vertex $v_i$ as a dominator)
        Then
            Vertex $v_i$ is added to the set of dominator set being selected in this stage.
            Weight $w_i$ associated with vertex $v_i$ is added to the weight of the selected dominator set.
            Vertex $v_i$ and all its neighbors are added to the set of dominatee set (if they have not been added yet).
    End for
Step 2-    <DOMINATION & CONNECTIVITY CHECKING>
    If (Terminal-set $R$ is a subset of dominatee set and the sub-graph induced by selected dominators is
    connected)
    Then
        Go to step 3
    Else
        Repeat step 1
Step 3-    <COMPARING WITH A DYNAMIC THRESHOLD>
    Let dynamic threshold $T_k$ (which is initially set to a large value) denotes the cardinality of the minimum size
    SCDS constructed until stage $k$.
    Dynamic threshold $T_k$ is updated to the cardinality of the constructed SCDS, if $T_k$ is larger than the cardinality
    of constructed SCDS.
Step 4-    <UPDATING THE ACTION PROBABILITY VECTOR>
    Depending on the result of the comparison in step 3, all the learning automata (only the activated automata in
    Algorithm III) reward their chosen actions if the cardinality of the constructed SCDS is less than or equal to the
    dynamic threshold $T_k$, and penalize them otherwise. Each learning automaton then updates its action
    probability vector using a $L_{R-I}$ reinforcement scheme.
Step 5-    <STOP CONDITION>
    The process of constructing the SCDSs and updating the action probabilities are repeated until the product of
    the probability of choosing the vertices of the constructed SCDS is greater than a certain threshold or the
    number of constructed SCDS exceeds a pre-specified threshold. The SCDS which is formed last before
    stopping the algorithm is the SCDS with the minimum cardinality among all SCDSs.
End Algorithm
```

Fig. 1 Algorithm I

4 The WSCDS Algorithms

In this section, we propose three centralized approximation algorithms based on learning automata for finding a near optimal solution to the minimum weighted Steiner connected dominating set problem as described in Sect. 3.1. In these algorithms, a learning automaton is first assigned to each vertex of the graph $G(V, E)$, and so a network of the learning automata isomorphic to the graph is formed. The resultant network of learning automata can be described by a triple $< \underline{A}, \underline{\alpha}, \underline{W} >$, where $\underline{A} = \{A_1, A_2, \ldots, A_n\}$ denotes the set of the learning automata, $\underline{\alpha} = \{\underline{\alpha_1}, \underline{\alpha_2}, \ldots, \underline{\alpha_n}\}$ denotes the set of actions in which $\underline{\alpha_i} = \{\alpha_i^1, \alpha_i^2, \ldots, \alpha_i^{ri}\}$ defines the action-set of learning automaton A_i, for each $\alpha_i \in \underline{\alpha}$, and $\underline{W} = \{w_1, \ldots, w_n\}$ denotes the set of weights such thatw_i is the weight associated with automaton A_i. Since the proposed algorithms associate a learning automaton with each host, hereafter a host may be referred to as its corresponding learning automaton and vice versa. In the first proposed algorithm, which we call it Algorithm I, the action-set of each learning automaton A_i (i.e., $\underline{\alpha_i}$) contains only two actions α_i^0 and α_i^1. Choosing action α_i^0 by learning automaton $A_i$declares vertex v_i as a dominatee vertex and choosing action α_i^1 declares it as a dominator vertex. Algorithm I consists of a number of stages, and kth stage of this algorithm is shown in Fig. 1.

The first proposed algorithm has two disadvantages which may increase the running time of algorithm. The first weakness is that some members of subset R may not be dominated by the selected *dominator set*, and the second is that the sub-graph induced by the constructed

```
Algorithm II The SCDS Formation Step of the Second Proposed Algorithm
Repeat
    For all learning automata do in parallel
        Each automaton $A_i$ chooses one of its actions (e.g. action $\alpha_i^j$) according to its action probability vector
        The vertex corresponding to the selected action (i.e., vertex $v_j$) is added to the dominator set
        The weight associated with vertex $v_j$ (i.e., weight $w_j$) is added to the weight of the dominator set which is
        being selected at stage $k$
    End for
Until ( The sub-graph induced by the selected dominator set is a connected sub-graph of $G$)
```

Fig. 2 The SCDS formation step (steps 1 and 2) in algorithm II

dominator set may be disconnected. In both cases, Algorithm I rejects the selected *dominator set*, and restarts step 1 for constructing a new *dominator set*.

The second algorithm we propose in this paper attempts to solve the first disadvantage of Algorithm I. This algorithm is very similar to Algorithm I, but differs in the action-set defined for each learning automaton, as well as the way of constructing the SCDS. In this algorithm which is called Algorithm II, the action-set of learning automaton A_i (corresponding to vertex v_i) includes an action for (associated with) each vertex v_j (for all $j \in \{1, \ldots, n\}$), where vertex v_j is adjacent to vertex v_i, as well as an action for vertex v_i itself. Indeed, in this algorithm, each vertex v_i, by choosing action $\alpha_i^j \in \underline{\alpha_i}$ declares vertex v_j as its dominator and adds it to the SCDS, if it has not yet been added. The first two steps of Algorithm II at each stage k are described as shown in Fig. 2. The remaining steps (*steps* 3, 4, *and* 5) of Algorithm II are the same as those described in Algorithm I.

Although the *dominator set* constructed by Algorithm II dominates all members of terminal-set R, it does not guarantee a *connected* dominating set to be formed at each stage of algorithm. This problem significantly increases the number of *dominator set*s that must be constructed until formation of a Steiner connected dominating set.

To solve the addressed problem, we propose another algorithm based on DLA called Algorithm III by which the formation of a SCDS at each stage of algorithm is guaranteed. In this algorithm, the action-set of each learning automaton A_i includes an action for each vertex v_j (for all $j \in \{1, \ldots, n\}$), where vertex v_j is adjacent to vertex v_i. At each stage of Algorithm III, the first dominator (e.g., vertex v_i) is randomly chosen using a separate learning automaton whose action-set contains the set of all the vertices of graph G. The chosen vertex is added to the SCDS, and its weight is added to the weight of SCDS. Vertex v_i and its neighbors are also added to the *dominatee set*. The learning automaton corresponding to vertex v_i (i.e., A_i) is activated. To improve the convergence speed of the learning automaton, some actions may be disabled in the action-set of the currently active automaton. To do so, the actions corresponding to the selected vertices (or dominators) are temporarily removed from the action-set of the active automaton. This avoids choosing a dominator many times. Furthermore, in Algorithm III, the currently active learning automaton updates its action-set by disabling the actions corresponding to the dominators by which no more vertices can be dominated. This avoids redundant dominators, and so considerably decreases the cardinality of the constructed SCDS. Now, the active learning automaton randomly chooses one of its actions according to its updated action probability vector. The learning automaton corresponding to the selected action is activated and does the same as the previous activated automata did. This sequential activation process is either repeated until formation of a SCDS (all terminals to be dominated) or no more actions can be taken by the currently active learning automaton. In the first case, the current iteration is over, and in the second case, the algorithm traces the path induced by the activated automata back for finding a learning

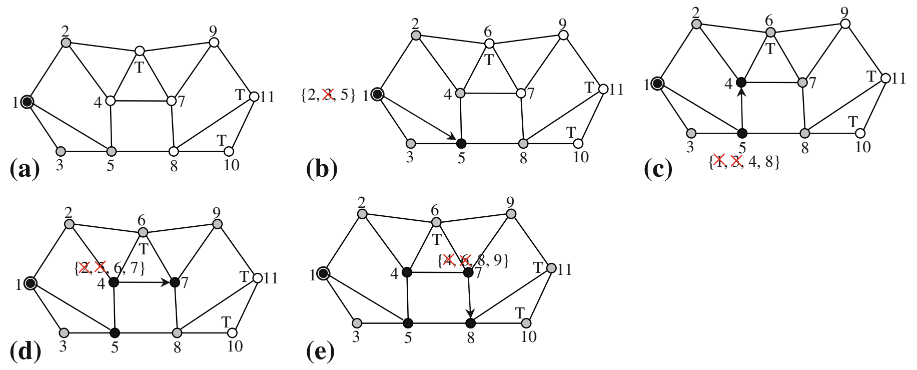

Fig. 3 The steps of SCDS formation in algorithm III

automaton with available actions. The action-set of such an automaton (found in backtracking phase) is updated by disabling its last chosen action and the current iteration is resumed as described earlier. In Algorithm III, each learning automaton may activate more than one of its neighbors at each stage. That is, more than one action can be chosen by each learning automaton. This algorithm can be implemented in a fully distributed manner and so we use it for multicast routing in ad-hoc networks. The SCDS formation step of Algorithm III has been shown in Fig. 3 for a sample graph. The remaining steps (*steps* 3, 4, *and* 5) of Algorithm III are the same as those described in Algorithm I.

The sample graph shown in Fig. 3 has 11 vertices and 18 edges. In these figures, a black circle represents a dominator, an encircled vertex represents the first chosen dominators, and a grey circle represents a dominatee. Therefore, a white circle represents a host that it is neither a dominator nor dominatee. The number beside each vertex represents its ID and the weight associated with that vertex too. The set near a dominator represents the action-set of the learning automaton corresponding to it. For instance in Fig. 3b, since vertices 2, 3 and 5 are the neighbors of dominator vertex 1, the action-set of the learning automaton corresponding to vertex 1 contains three actions for choosing vertices 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 Fig. 3b). Letter "T" near a vertex represents that this vertex is a terminal and must be dominated. In this example, vertices 6, 10 and 11 form the terminal-set. An arrow from vertex v_i to v_j means that the action corresponding to vertex v_j has been chosen by automaton A_i, and so automaton A_j should be activated. As shown in Fig. 3a, let us assume that vertex 1 is chosen as the first dominator. It is added to the SCDS and *dominatee set* also. Its neighboring vertices are added to the*dominatee set*. Its weight is added to the weight of SCDS. Vertices 2, 3 and 4 form the initial action-set of the learning automaton (i.e.,{2, 3, 5}) corresponding to vertex 1. As described earlier, the action corresponding to vertex 3 must be disabled, since it dominates no more actions. Therefore, vertex 1 randomly chooses one of its two remaining actions. Let us assume that vertex 5 to be chosen as the second dominator. Vertex 5 is activated and does the same as vertex 1 did. Let us assume that vertex 5 chooses vertex 4 (by which terminal 6 is dominated), vertex 4 chooses vertex 7, and vertex 7 finally chooses vertex 8 (by which terminals 10 and 11 are dominated). At this point the current iteration is over, and the selected SCDS contains {1, 4, 5, 7, 8}. The weight of the selected SCDS is calculated as the sum of the weight of its members. According to assumptions, this weight is 25 for this example. Like Algorithm I, in step 3, the weight of the selected SCDS is compared with the dynamic threshold T_k. If this weight is less than or equal to the dynamic threshold T_k, in step 4, the activated automata are rewarded, and they are penalized otherwise. The action probability vector of the activated automata is updated using a L_{R-I} reinforcement scheme. The stop condition is verified and a new iteration begins, if the stop condition is false. Due to the distributed nature of DLA, Algorithm III can be also implemented in a fully distributed manner, and so in the next section, it will be used for multicast routing in wireless mobile ad-hoc networks.

5 DLA-based Multicast Routing Algorithm

In this section, a distributed learning automata-based multicast routing algorithm called DLAMRA is proposed for wireless mobile ad-hoc networks. The multicast routing algorithm that we are going to propose in this paper is a distributed version of the last WSCDS algorithm proposed in Sect. 4 (i.e., Algorithm III), where the relative speed of each host is considered as its weight. In this approach, the multicast messages are forwarded along the MVB by which the multicast source is connected to each of the multicast receivers. This backbone can be found by solving the WSCDS problem in the network graph where subset$R$comprises the multicast source and the multicast receivers. In this method, each host (e.g., h_i) is equipped by a learning automaton (e.g., A_i) whose learning algorithm is linear reward-inaction. The resulting network of learning automata is isomorphic to the network graph and can be described by a triple $< \underline{A}, \underline{\alpha}, \underline{W} >$ as described in Sect. 4, where w_i represents the mobility speed associated with host h_i.

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_j by which the message is replied is associated with action α_i^j in the action-set of automaton A_i. Action α_i^j corresponds to the selection of host h_j 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. Due to the frequent topology changes in mobile ad-hoc networks, the action-set size of the learning automata is always changing.

The problem with the action-set defined above is that the number of actions is fixed and does not vary with time. 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 dominated. 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 rule for pruning the action-set of such learning automata.

Pruning Rule To avoid the loops and the redundant dominator hosts by which no more (dominatee) hosts can be dominated and to avoid choosing the same dominators (by different hosts), the proposed algorithm prunes the action-set as follows. As mentioned earlier, when host h_i is forming 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 the 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 dominated earlier, if any. By this rule, the action probability vector of each activated learning automaton is scaled as described in Sect. 3.4, on the variable action-set learning automata. At the end of each iteration, the disabled actions of each activated learning automaton must be re-enabled for the next iteration.

5.1 Multicast Routing Procedure

In this algorithm, each mobile host immediately broadcasts its mobility information (mobility speed and movement direction) to its neighboring hosts, when it experiences a new epoch (or its mobility characteristics change). Each mobility epoch is considered as a (short) period of time in which both the mobility speed and the movement direction of a mobile host are constant. Then, each host calculates its relative mobility, on the basis of the recently received information from its neighbors. The relative mobility is defined as the mobility degree a mobile host exhibits with respect to its neighbors. The relative mobility of each host (with respect to all its neighbors) is considered as a criterion to measure the mobility degree of the host, and to classify the hosts for finding the more stable multicast routes.

To compute the relative mobility, the mobility profile (mobility speed and movement direction) of each host must be exchanged with its neighboring hosts. Since the mobility characteristics of a mobile host vary with time, the host relative mobility changes as the host moves. Let s_i^k denotes the mobility speed of host $\boldsymbol{h}_i$ at instant k, and α_i^k denotes the movement direction of host $\boldsymbol{h}_i$ at instant k . Thus, the relative mobility between two mobile hosts $\boldsymbol{h}_i$ and $\boldsymbol{h}_i$ at instant k is defined as

$$\mathcal{R}_{ij}^k = \sqrt{s_i^{k^2} + s_j^{k^2} - 2\alpha_i^k \alpha_j^k \cos\left(\alpha_i^k - \alpha_j^k\right)}$$

Then, the relative mobility of a host with respect to all its neighbors can be achieved as follows.

$$\mathbf{R}_i^k = \frac{\sum_{\forall h_j \in \mathcal{N}_1} \mathcal{R}_{ij}^k}{|\mathcal{N}_{\mathbf{i}}|} \tag{4}$$

where $\mathcal{N}_{\mathbf{i}} = \{\boldsymbol{h}_j | \boldsymbol{h}_j$ is a neighbor of $\boldsymbol{h}_i\}$ denotes the set of all the neighbors of host $\boldsymbol{h}_i$, and $|\mathcal{N}_{\mathbf{i}}|$ represents the cardinality of $\mathcal{N}_{\mathbf{i}}$. Relative mobility $\mathbf{R}_i^k$ is periodically calculated and assigned to each host $\boldsymbol{h}_i$ as a weight. The relative mobility speed of each host, in the proposed multicast routing method, corresponds to the weight associated with each vertex in equivalent WSCDS problem. A host with a high relative mobility is more prone to the erratic mobility behaviors compared to the stable hosts, and so more stable routes can be founded on the hosts with lower relative mobility.

Each host included in the Steiner connected dominating set (or multicast route) 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 SCDS, if it is not included in the SCDS. In this method, upon receiving a multicast message, the dominator hosts re-broadcast it, while the dominatee hosts only receive the message. That is, the dominator hosts assume the role of the (intermediate) relay hosts. At each iteration of algorithm, the hosts which are selected as dominators form a route from the multicast source to each of the multicast receivers. The process of choosing the dominators is described later. The learning automata iteratively construct the multicast routes and update the action probability vectors until they find a near optimal solution to the WSCDS problem that guarantees the stability of the formed multicast route. Each host requires the following data structures to participate in the multicast routing process: Let Max_itr denotes the stopping condition of algorithm as a maximum number of iterations, Dominator_set denotes the set of dominator hosts by which the WSCDS is formed, Dominatee_set be the set of hosts in which each member is a one-hop neighbor of at least one dominator host in the Dominator_set, MRP denotes the threshold required for terminating the multicast routing process as the product of the probability of choosing the dominator hosts in the Dominator_set, Prob_vct denotes the vector of the probability of choosing the members of the Dominator_set, Itr_num be a counter which keeps the number of constructed Dominator_set, Multicast_grp includes the set of hosts to which the multicast message must be forwarded, Multicast_src denotes the host by which the multicast message is sent out, Trshld be a dynamic threshold including the weight of the minimum (weight) multicast routes (dominator sets) constructed yet which is initially set to a large value, and Dom_set_wgt denotes the weight associated with the chosen Dominator_set.

When a multicast source decides to initiates a multicast session (or to send a multicast message to a multicast group), it inserts its one-hop neighbors' ID to the Dominatee_set, activates its corresponding learning automaton, disables some actions according to the pruning rule described earlier, chooses an action according to its action probability vector, generates an Activation message, and finally sends it to the mobile host corresponding to the chosen action. Each Activation message includes Dominatee_set, Dominator_set, Multicast_grp, Multicast_src, Trshld, Dom_set_wgt, Prob_vct, and Itr_num. The multicast group members are included in the Multicast_grp, and multicast source inserts the probability of the chosen action to the Prob_vct. The Activation message is described below.

When a given host h_i receives an Activation message, it 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 relative speed (or weight) of the activated host is added to the Dom_set_wgt. Now, host h_i checks whether all multicast receivers are included in Dominatee_set or not. If so, it checks the stopping condition. Otherwise, host h_i updates its action-set using pruning rule, and determines whether there exist any available actions or not.

If there exist any actions can be chosen by learning automaton L_i, learning automaton L_i is activated and chooses one of its actions as a new dominator host. Otherwise, the path induced by the activated learning automata (selected dominator hosts) is traced back for finding a learning automaton with available actions. The action-set of such a learning automaton (found in backtracking phase) is updated by disabling its previous chosen action and the current iteration is resumed by choosing a new action. The probability of the chosen action is inserted in Prob_vct, and an Activation message is sent to the chosen dominator host.

To verify the stopping condition of the multicast routing process, the probability of choosing the recently selected Dominator_set is computed. This probability is defined as the product of the probability of choosing the dominator hosts contained in the Dominator_set. If this probability is greater than the certain threshold MRP or Itr_num exceeds a per-specified threshold Max_itr, dominator host h_i generates a MULTIICAST message including the last selected Dominator_set (or multicast route) and broadcasts it within the network. Otherwise (i.e., when the stopping condition is false), the relative speed of the chosen Dominator_set is calculated as the sum of the relative speed of the dominators contained in Dom_set_wgt. If the weight of the selected Dominator_set is less than the dynamic threshold Trshld, then dynamic threshold Trshld is updated to the weight of the selected Dominator_set 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. Rewarding and Penalizing messages are binary reinforcement signals. Multicast source starts a new iteration when it receives a Rewarding or a Penalizing message.

As our algorithm proceeds, the value of Trshld tends to the weight of the most stable multicast route, and so the probability of choosing the more stable routes increases as the probability of penalizing the unstable routes increases. Finally, the probability of choosing the most stable multicast route (i.e., the multicast route with the minimum relative speed) converges to one. In what follows, we describe the Multicast, Penalizing and Rewarding messages which are used in an Activation message.

A Multicast message includes the multicast route selected during the last iteration. When host h_i receives a Multicast message, it is noticed that the multicast routing process has been finished, and it thereafter uses the multicast route contained in the Multicast message to send the multicast packets to the given multicast members. It will then terminate the multicast routing procedure.

Let α_i^j denotes the chosen action by learning automaton A_i. When dominator host h_i receives a Rewarding message, it updates its action probability vector using the learning algorithm given in Eq. (5), under which the chosen action (i.e., α_i^j) is rewarded, and the other actions ($\alpha_{i,k}$, for all $k \neq j$) are penalized.

$$\begin{aligned} p_{i,j}(n+1) &= p_{i,j}(n) + a[1 - p_{i,j}(n)], \\ p_{i,k}(n+1) &= (1-a)p_{i,k}(n) \quad \forall k \neq j. \end{aligned} \tag{5}$$

where $p_{i,j}$ is the probability with which host h_i chooses host h_j as a dominator host.

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 Sect. 3.4 on the variable action-set learning automata. In this case, the multicast source starts a new iteration as described earlier.

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, and the multicast source starts a new iteration upon receiving a Penalizing message.

6 Experimental Results

In this paper, we first proposed three learning automata-based algorithms for approximating a near optimal solution to the minimum weighted Steiner connected dominating set problem (Sect. 4). Then, we proposed a multicast routing algorithm for wireless mobile ad-hoc networks based on distributed learning automata (Sect. 5). Therefore, we have conducted two groups of simulation experiments. In the first group of our experiments, we compare the centralized algorithms proposed (in Sect. 4) for solving the minimum WSCDS problem, and in the second group, we study the performance of the proposed multicast routing algorithm in comparison with the best existing methods.

6.1 WCDS Algorithms

The simulation experiments conducted in this section are concerned with investigating the efficiency of the centralized approximation algorithms proposed for solving the minimum WSCDS problem. The results of these algorithms are compared with those of the previous well-known algorithms like Guha and Khuller's algorithm [2], Wu et al.'s algorithm [3], and Muhammad's algorithm [7].

In all experiments presented in this paper, the reinforcement scheme used for updating the action probability vectors is L_{R-I}, and the learning rate is 0.1. To generate the random graphs, a number of vertices are uniformly distributed in a two-dimensional simulation area of size 1,000 × 1,000 units at random. We assume that a link to be established between two vertices, if the distance between them is not longer than 200 units. Each algorithm is tested only on the connected graphs, and the reported results are averaged over 100 runs. Each algorithm is terminated when the probability of choosing the SCDS approaches 0.95, or the number of selected SCDSs exceeds 10,000.

In these experiments, 100 vertices are randomly distributed in the simulation area. To study the impact of the terminal-set size on the performance of the proposed algorithms, the size of the terminal-set ranges from 10 to 100. The average size of the Steiner connected dominating set constructed by the proposed algorithms is given in Fig. 4, and the running time of each algorithm is given in Fig. 5.

From the results given in Fig. 4, it is clear that, for all algorithms the size of the Steiner connected dominating set increases as the terminal-set size increases. For instance, in Guha and Khuller's algorithm, the SCDS size is 7.35 when the terminal-set size is 10, and it approaches 19.10 when the terminal-set size increases to 100.

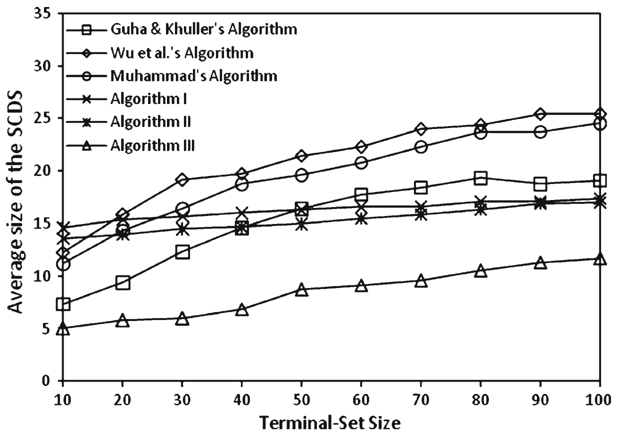

Fig. 4 The average size of the Steiner connected dominating set constructed by different algorithms

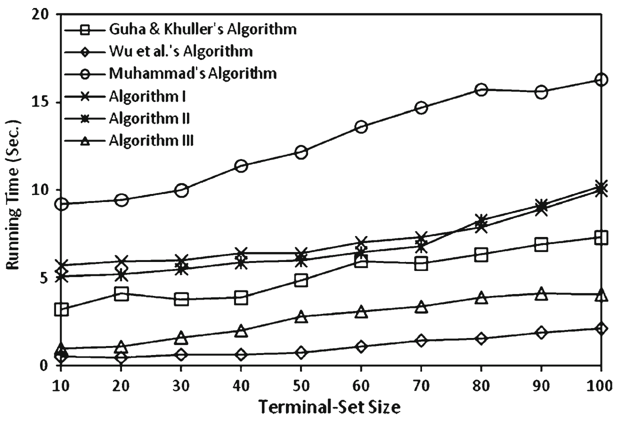

Fig. 5 The average time taken by different algorithms (in seconds) to construct the Steiner connected dominating set

The results given in Figs. 4 and 5 show that, as expected, Algorithm III outperforms the others, and Algorithm I performs worst among the algorithms presented in this paper. As described earlier, in Algorithm II, each vertex chooses its dominator itself, and so unlike Algorithm I, it guarantees to form a dominating set at each iteration. Therefore, it is expected that Algorithm II will improve the running time of Algorithm I. However, from Fig. 5, it is clear that the running time of Algorithm1 is very close to that of Algorithm II. This is because the action-set size of each learning automaton in Algorithm II is much larger than that of Algorithm I, and this prolongs the convergence of learning automaton to its optimal action, and increases the running time of algorithm. The results given in Figs. 4 and 5 also show that Algorithm III significantly outperforms Algorithms I and II both in terms of the SCDS size and running time. This is due to the fact that at each stage of this algorithm a Steiner connected dominating set is formed. Furthermore, taking advantage of the variable action-set learning automata, Algorithm III avoids the redundant dominators by which no more vertices can be dominated.

From the results given in Figs. 4 and 5, it can be concluded that Algorithm III has the best performance both in terms of the running time and dominating set size compared to the first two algorithms presented in this paper (Algorithms I and II). Therefore, to evaluate our findings on constructing the optimal Steiner connected dominating set, we compare the results of Algorithm III with the best existing algorithms such as Guha and Khuller's algorithm [2], Wu et al.'s algorithm [3], and Muhammad's algorithm [7].

Comparing the results of the previous algorithms given in Fig. 4, we find that, in most cases, Guha and Khuller's algorithm forms the Steiner connected dominating set with the smallest number of dominators, and Muhammad's algorithm and Wu et al.'s algorithm lag far behind. Though Wu et al.'s algorithm and Muhammad's algorithm both are MIS-based algorithms, it can be seen that the size of the Steiner connected dominating set constructed by Muhammad's algorithm is smaller compared to Wu et al.'s algorithm. The results show that our last proposed algorithm (Algorithm III) significantly outperforms the existing algorithms in terms of the SCDS size, especially for the larger terminal-sets.

Figure 5 shows the average time taken (in seconds) by each algorithm to construct the Steiner connected dominating set of the random graphs. Comparing the results of algorithms given in Fig. 5, it is observed that Wu et al.'s algorithm outperforms the others in terms of the average time required for constructing the SCDS. It can be also seen that Muhammad algorithm has the longest running time. From the results given in Fig. 5, it is clear that, the running time of all algorithms is directly proportional to the number of terminals, and it increases as the terminal-set size increases. The obtained results also show that our last proposed algorithm considerably outperforms Guha and Khuller's algorithm and Muhammad's algorithm, but has a longer running time in comparison with Wu et al.'s algorithm.

Comparing the results shown in Fig. 4 with Fig. 5, it can be concluded that our proposed algorithm (Algorithm III) generates much smaller Steiner connected dominating sets compared to the other algorithms in a reasonable time (comparable to the fastest method), and so has the best overall performance. The size of the SCDS constructed by Muhammad's algorithm is as large as that constructed by Wu et al.'s algorithm, whereas Wu et al.'s algorithm has the shortest running time. Therefore, Muhammad's algorithm has the worst overall performance and Wu et al.'s algorithm is ranked below Algorithm III. Guha and Khuller's algorithm is also ranked lower than Wu et al.'s algorithm but very higher than Muhammad's algorithm.

6.2 Multicast Routing Algorithm

To study the performance of the multicast routing algorithms, we have conducted several simulation experiments in two sets. In the first set of experiments, we investigate the impact of the host mobility on the performance of algorithms. In these simulation experiments, the multicast group size is fixed at 10, and the host speed varies from 10 to 70 km/h. The second set of the simulation experiments aims at evaluating the scalability of the multicast routing algorithms, and so in these experiments, the host mobility speed is fixed at 15 km/h and the multicast group size changes from 5 to 30. In these experiments, the performance of the various multicast routing algorithms is evaluated in terms of the following metrics.

- Packet delivery ratio. This metric is defined as the number of data packets delivered to the multicast receivers over the number of data packets supposed to be received by multicast receivers. This ratio represents the efficiency of routing in our proposed method.
- End-to-end delay. The time elapsed between the instant when the source has data packet to send and the instant when the destination receives the data. Note that if no multicast route is available, the time spent for building a route (route acquisition latency) is also included in the end-to-end delay. In this case, this metric is defined as the time required for multicast route creation as well as the time required for transmitting the multicast packets.
- Multicast route Lifetime. The time interval during which the multicast routes remain connected. In mobile ad hoc networks, the network topology changes, caused by the host movement, shortens the lifetime of the links. To find the stable routes, these movements should be exactly estimated. This metric represents the efficiency of each algorithm to predict the realistic mobility behavior of a host.

To show the efficiency of our proposed multicast routing algorithm, we compare its obtained results with those of the mobility-based hybrid multicast routing protocol proposed by An and Papavassiliou [13], hereafter referred to as AP, two enhanced versions of the ODMRP proposed by Su et al. [12], hereafter referred to as SLG-1 and SLG-2, and two distributed multicast routing algorithms proposed by Guo and Yang [14], hereafter referred to as GY-1 and GY-2.

In our simulation scenarios, a mobile ad hoc network consisting of 50 mobile hosts is modeled in which the mobiles are randomly and uniformly distributed within a square simulation area of size 1,000 m $\times$ 1,000 m. Each host is modeled as an infinite-buffer, store-and forward queuing station, and is assumed to be aware of its mobility information with the aid of a reliable positioning system. The IEEE 802.11 DCF [44] (Distributed Coordination Function) with CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) is used as the medium access control protocol, and two ray ground as the propagation model. The wireless hosts communicate through a common broadcast channel of capacity 2 Mb/s using omnidirectional antennas. All mobile hosts have the same radio propagation range of 250 m. CBR (Continuous Bit Rate) traffic sources are used to generate the traffics with a rate of 20 packets per second. The packet size is 512 bytes. In our experiments, *MRP* is set to 0.90, and *MAX_ITR* is set to 100. Mobility characteristics change at the beginning of each epoch and remain constant during the epoch. Each multicast group is associated with a multicast source, and the multicast members and source are randomly chosen with a uniform distribution. The multicast members join the group at the start of the multicast session and remain as members throughout the session. Each experiment is run on 100 connected graphs and the results, presented in this paper, are averaged over these runs.

In these experiments, the packet delivery ratio is measured as a function of the host mobility speed. It is clear from Fig. 6 that, the packet delivery ratio of GY-I, GY-II and AP rapidly

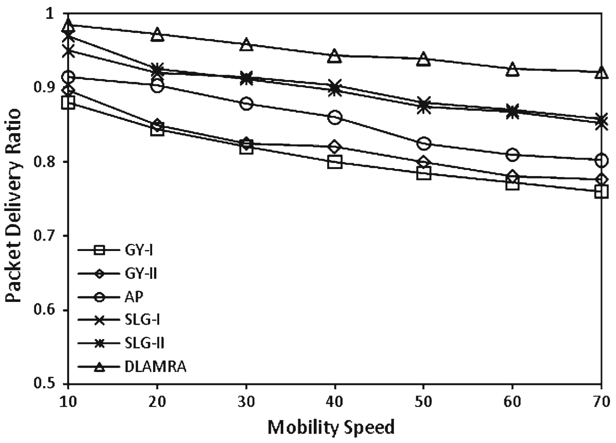

Fig. 6 Packet delivery ratio of the multicast routing algorithms as a function of the mobility speed

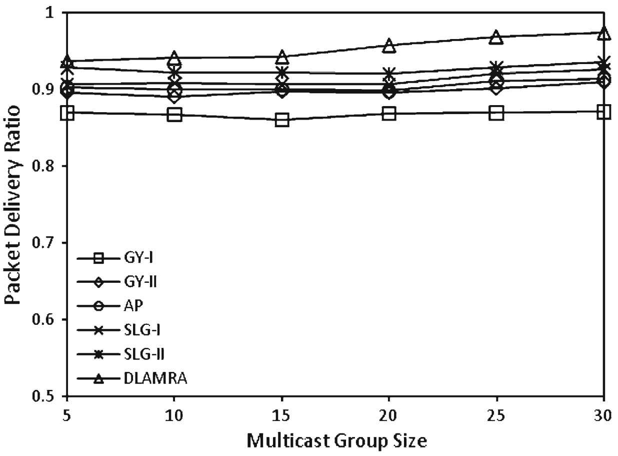

Fig. 7 Packet delivery ratio of the multicast routing algorithms as a function of the multicast group size

degrades as the mobility speed increases. SLG-I and SLG-II are more stable, and the packet delivery ratio of DLAMRA more slowly decreases compared with the other algorithms. The results given in Fig. 6 show that GY-I has the lowest packet delivery ratio, and DLAMRA provides the highest packet delivery ratio. This is due to the fact that, it uses the relative speed of the host (with respect to all its neighbors) as a criterion for selection of the routes. Therefore, it chooses the more stable routes that are not affected by the host mobility. In algorithms SLG-I and SLG-II, since they reconstruct the routes in advance of the topology changes, most data are delivered to the multicast receivers without being dropped. Therefore, they show a good performance in highly dynamic environments, and are ranked below DLAMRA.

The packet delivery ratio of the multicast routing algorithms as a function of the multicast group size is drawn in Fig. 7. As shown in this figure, the packet delivery ratio of all algorithms is slightly improved as the multicast group size increases. Since in DLAMRA the dominator hosts broadcast the multicast packets to all its neighboring hosts, DLAMRA is

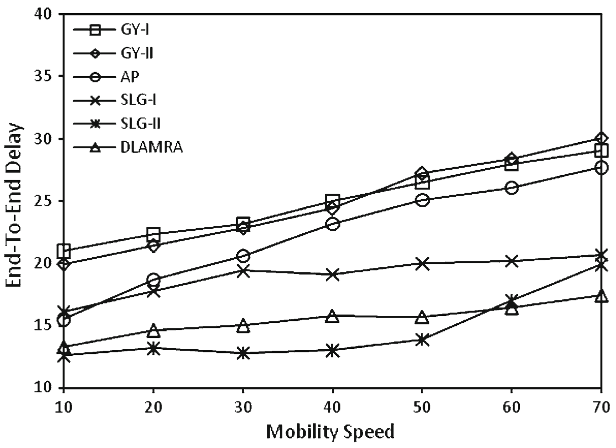

Fig. 8 End-to-end delay as a function of the mobility speed

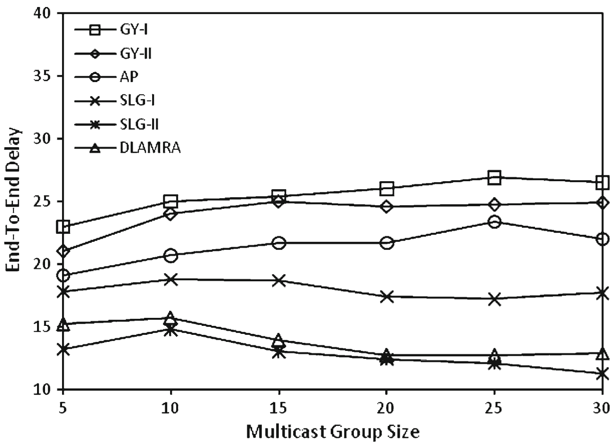

Fig. 9 End-to-end delay as a function of the multicast group size

robust to multicast group size. Like those in Fig. 6, here DLAMRA has the highest packet delivery ratio, and SLG-II lags behind. The results show that the packet delivery ratio of SLG-I is very close to that of AP and GY-II. It can be also seen that GY-I has the lowest delivery ratio and is ranked below GY-II.

Figures 8 and 9 show the end-to-end delay of each multicast routing algorithm. Figure 8 shows the end-to end delay as a function of the mobility speed, and Fig. 9 shows it as a function of the multicast group size. In these experiments, we varied the mobility speed from 10 to 70 km/h and measured the end-to-end delay of each algorithm. As shown in these figures, SLG-II and DLAMRA have the shortest end-to-end delay, and GY-I performs worst compared with the others. From the results shown in Fig. 8, it is clear that GY-II only slightly outperforms GY-I. In SLG-II, multicast sources flood the *Join Data* before they form the forwarding groups. Therefore, in SLG-II, the route acquisition latency is eliminated and the packets are delivered to the multicast receivers in a shorter time. DLAMRA finds the more stable routes, and this causes a longer delay. On the other side, it minimizes the size (number

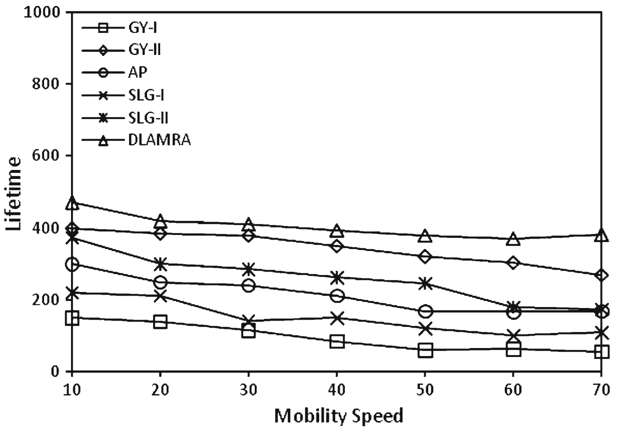

Fig. 10 The lifetime of the multicast route as a function of the mobility speed

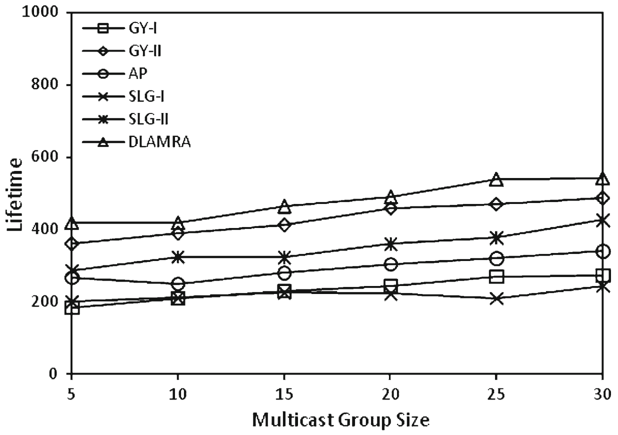

Fig. 11 The lifetime of the multicast route as a function of the multicast group size

of relay hosts) of the multicast routes also. Generally, it sends the multicast packets in a reasonable delay.

Among the studied algorithms, AP, SLG-II, and DLAMRA consider the mobility-prediction issues for constructing the stable multicast routes. As described earlier, SLG-II uses the route expiration time as the route selection metric, AP utilizes the concept of the relative mobility to characterize the mobility degree of a host. SLG-II and AP predict the motion behaviors of a host based on the samples taken from the mobility parameters during a single epoch. DLAMRA samples the mobility characteristics in different epochs to estimate their expected values. Therefore, it finds the more stable routes that stay connected for a longer time. The lifetime of the multicast routes constructed by the various algorithms is presented in Figs. 10 and 11. The results depicted in these figures show that DLAMRA significantly outperforms the others and GY-I and SLG-I have the worst results in terms of the route lifetime. The routes constructed by SLG-II are more stable than those of AP, but their lifetime is much shorter compared with GY-II. From Fig. 10, it is obvious that the route lifetime is degraded as the mobility speed increases. This is because the wireless connections between the hosts become looser as the host speed increases. The number of intermediate hosts required for relaying the multicast packets increases as the number of multicast receivers increases. On the other side, the duration of the multicast route is directly proportional to the number of relay nodes. Therefore, as shown in Fig. 11, the multicast route duration increases as the multicast group size increases.

7 Conclusion

In this paper, we first proposed three learning automata-based approximation algorithms for finding a near optimal solution to the minimum weighted Steiner connected dominating set problem. Then, we proposed a multicast routing algorithm for wireless mobile ad-hoc networks based on distributed learning automata. Our multicast routing algorithm is a distributed implementation of the last algorithm proposed in this paper for solving the weighted Steiner connected dominating set problem. We compared the proposed centralized WSCDS algorithms with the best existing algorithms and showed that the last proposed algorithm (Algorithm III) outperforms the others both in terms of the dominating set size and running time. The results of the simulation experiments showed that our proposed multicast routing algorithm is superior to the well-known multicast routing protocols in terms of the packet delivery ratio, multicast route lifetime, and end-to-end delay.

Acknowledgement The authors would like to thank the editor and anonymous reviewers whose helpful comments and constructive criticism improved the quality of this paper.

References

1. Nadeem, T., & Parthasarathy, S. (2006). Mobility control for throughput maximization in ad-hoc networks. *Wireless Communication and Mobile Computing, 6*, 951–967.
2. Guha, S., & Khuller, S. (1998). Approximation algorithms for connected dominating Sets. *Algorithmica, 20*(4), 374–387.
3. Wu, Y., Xu, Y., Chen, G., & Wang, K. (2004). On the construction of virtual multicast backbone for wireless ad-hoc networks. In *IEEE international conference on mobile ad-hoc and sensor systems*, pp. 294 –303.
4. Lim, H., & Kim, C. (2001). Flooding in wireless ad-hoc networks. *Journal of Computer Communications, 24*, 353–363.
5. Robins, G., & Zelikovsky, A. (2000). Improved steiner tree approximation in graphs. In *Proceedings of the 11th annual ACM-SIAM symposium on discrete algorithms*, pp. 770–779.
6. Singh, G., & Vellanki, K. (1998). A distributed protocol for constructing multicast trees. In *Proceedings of the international conference on principles of distributed systems*, France.
7. Muhammad, R. B. (2006). Distributed steiner tree algorithm and its application in ad-hoc wireless networks. In *Proceedings of the 2006 international conference on wireless networks (ICWN'06)*, USA, pp. 173–178.
8. Muhammad, R. B. (2007). A distributed graph algorithm for geometric routing in ad-hoc wireless networks. *Journal of Networks, 2*(6), 50–57.
9. Aggarwal, D., Dubey, C. K., & Mehta, S. K. (2006). Algorithms on Graphs with Small Dominating Targets. *ISAAC, 4288*, 141–152. LNCS.
10. Lee, S. J., Gerla, M., & Chiang, C. C. (1999). On-demand multicast routing protocol. In *Proceedings of IEEE WCNC'99*, pp. 1298–1302. New Orleans, LA.
11. Chiang, C. C., Gerla, M., & Zhang, L. (1998). Forwarding group multicast protocol (FGMP) for multihop, mobile wireless networks. *Journal of Cluster Computing, 1*(2), 187–196.
12. Su, W., Lee, S. J., & Gerla, M. (2001). Mobility prediction and routing in ad-hoc wireless networks. *International Journal of Network Management, 11*, 3–30.

13. An, B., & Papavassiliou, S. (2003). MHMR: Mobility-based hybrid multicast routing protocol in mobile ad-hoc wireless networks. *Wireless Communication and Mobile Computing, 3*, 255–270.
14. Guo, S., & Yang, O. (2008). Maximizing multicast communication lifetime in wireless mobile ad-hoc networks. *IEEE Transactions on Vehicular Technology, 57*, 2414–2425.
15. Haleem, M., & Chandramouli, R. (2005). Adaptive downlink scheduling and rate selection: A cross layer design, special issue on mobile computing and networking. *IEEE Journal on Selected Areas in Communications, 23*(6).
16. Nicopolitidis, P., Papadimitriou, G. I., & Pomportsis, A. S. (2006). Exploiting locality of demand to improve the performance of wireless data broadcasting. *IEEE Transactions on Vehicular Technology, 55*(4), 1347–1361.
17. Nicopolitidis, P., Papadimitriou, G. I., & Pomportsis, A. S. (2003). Learning-automata-based polling protocols for wireless LANs. *IEEE Transactions on Communications, 51*(3), 453–463.
18. Nicopolitidis, P., Papadimitriou, G. I., & Pomportsis, A. S. (2004). Distributed protocols for ad-hoc wireless LANs: A learning-automata-based approach. *Ad-Hoc Networks, 2*(4), 419–431.
19. Nicopolitidis, P., Papadimitriou, G.I., Obaidat, M. S., & Pomportsis, A. S. (2005). Carrier-sense-assisted Adaptive Learning MAC Protocol for Distributed Wireless LANs. *International Journal of Communication Systems*, Wiley *18*(7), 657–669.
20. Ramana, B. V., & Murthy, C. S. R. (2005). Learning-TCP: A novel learning automata based congestion window updating mechanism for ad-hoc wireless networks. In *12th IEEE International Conference on High 13 Performance Computing*, pp. 454–464.
21. Beigy, H., & Meybodi, M. R. (2008). Learning automata-based Dynamic guard channel algorithms. In *Journal of High Speed Networks*, (to appear).
22. Beigy, H., & Meybodi, M. R. (2005). A general call admission policy for next generation wireless networks. *Computer Communications, 28*, 1798–1813.
23. Beigy, H., & Meybodi, M. R. (2005). An adaptive call admission algorithm for cellular networks. *Computers and Electrical Engineering, 31*, 132–151.
24. Beigy, H., & Meybodi, M. R. (2002). A learning automata-based dynamic guard channel scheme. In *Lecture notes on information and communication technology* vol. 2510, pp. 643–650. Springer.
25. Akbari Torkestani, J., & Meybodi, M. R. (2010). An efficient cluster-based CDMA/TDMA scheme for wireless mobile ad-hoc networks: A learning automata approach. *Journal of Network and Computer applications*, (in press).
26. Akbari Torkestani, J., & Meybodi, M. R. (2010). An intelligent backbone formation algorithm for wireless ad-hoc networks based on distributed learning automata. Computer Networks, Elsevier Publishing Company, (in press).
27. Akbari Torkestani, J., & Meybodi, M. R. (2010). Mobility-based multicast routing algorithm in wireless mobile ad-hoc networks: A learning automata approach. *Journal of Computer Communications, 33*, 721–735.
28. Wu, J., Dai, F., Gao, M., & Stojmenovic, I. (2002). On calculating power-aware connected dominating sets for efficient routing in ad-hoc wireless networks. *Journal of Communications and Networks, 4*(1), 1–12.
29. Wu, J., & Li, H. (1999). On calculating connected dominating set for efficient routing in ad-hoc wireless networks. In *Proceedings of the 3rd international workshop on discrete algorithms and methods for mobile computing and communication*, pp. 7–14.
30. Wang, Y., Wang, W., & Li, X.-Y. (2005). Distributed low-cost backbone formation for wireless ad-hoc networks. In *Proceedings of the sixth ACM international symposium on mobile ad-hoc networking and computing (MobiHoc 2005)*, pp. 2–13.
31. Han, B. (2009). Zone-based virtual backbone formation in wireless ad-hoc networks. *Ad-Hoc Networks, 7*, 183–200.
32. Alzoubi, K. M., Wan, P. -J., & Frieder, O. (2003). Maximal independent set, weakly connected dominating set, and induced spanners for mobile ad-hoc networks. *International Journal of Foundations of Computer Science, 14*(2), 287–303.
33. Clark, B. N., Colbourn, C. J., & Johnson, D. S. (1990). Unit Disk Graphs. *Discrete Mathematics, 86*, 165–177.
34. Marathe, M. V., Breu, H., Hunt, H. B., Ravi, S. S., & Rosenkrantz, D. J. (1995). Simple Heuristics for Unit Disk Graphs. *Networks, 25*, 59–68.
35. Chen, Y. Z., & Listman, A. L. (2002). Approximating minimum size weakly connected dominating sets for clustering mobile ad-hoc networks. In *Proceedings of the third ACM international symposium on mobile ad-hoc networking and computing (MobiHoc'2002)*, pp. 157–164.
36. Torkestani, J. A., & Meybodi, M. R. (2009). Approximating the minimum connected dominating set in stochastic graphs based on learning automata. In *Proceedings of international conference on information management and engineering (ICIME 2009)*, pp. 672–676. Malaysia.

37. Narendra, K. S., & Thathachar, K. S. (1989). *Learning automata: An introduction*. New York: Printice-Hall.
38. Thathachar, M. A. L., & Sastry, P. S. (1997). A hierarchical system of learning automata that can learn the globally optimal path. *Information Science, 42*, 743–766.
39. Thathachar, M. A. L., & Harita, B. R. (1987). Learning automata with changing number of actions. *IEEE Transactions on Systems, Man, and Cybernetics, SMG17*, 1095–1100.
40. Thathachar, M. A. L., & Phansalkar, V. V. (1995). Convergence of teams and hierarchies of learning automata in connectionist systems. *IEEE Transactions on Systems, Man, and Cybernetics, 24*, 1459–1469.
41. Lakshmivarahan, S., & Thathachar, M. A. L. (1995). Bounds on the convergence probabilities of learning automata. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-6*, 756–763.
42. Narendra, K. S., & Thathachar, M. A. L. (1980). On the behavior of a learning automaton in a changing environment with application to telephone traffic routing. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-l0*(5), 262–269.
43. Torkestani, J. A., & Meybodi, M. R. (2009). Solving the minimum spanning tree problem in stochastic graphs using learning automata. In *Proceedings of international conference on information management and engineering (ICIME 2009)*. pp. 643–647. Malaysia.
44. IEEE Computer Society LAN MAN Standards Committee, *Wireless LAN Medium Access Protocol (MAC) and Physical Layer (PHY) specification*, IEEE Standard 802.11-1997, The Institute of Electrical and Electronics Engineers, New York (1997).
45. Akbari Torkestani, J., & Meybodi, M. R. (2010). Clustering the wireless ad-hoc networks: A distributed learning automata approach. *Journal of Parallel and Distributed Computing*, Elsevier Publishing Company (in press).
46. Akbari Torkestani, J., & Meybodi, M. R. (2010). A new vertex coloring algorithm based on variable action-set learning automata. *Journal of Computing and Informatics* (in press).

Author Biographies

Javad Akbari Torkestani received the B.S. and M.S. degrees in Computer Engineering in Iran, in 2001 and 2004, respectively. He also received the Ph.D. degree in Computer Engineering from Science and Research University, Iran, in 2009. Currently, he is an assistant professor in Computer Engineering Department at Arak Azad University, Arak, Iran. Prior to the current position, he joined the faculty of the Computer Engineering Department at Arak Azad University as a lecturer. 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.