Cluster Comput
DOI 10.1007/s10586-011-0161-z

A mobility-based cluster formation algorithm for wireless mobile ad-hoc networks

Javad Akbari Torkestani · Mohammad Reza Meybodi

Received: 26 October 2010 / Accepted: 28 January 2011
© Springer Science+Business Media, LLC 2011

Abstract In the last decade, numerous efforts have been devoted to design efficient algorithms for clustering the wireless mobile ad-hoc networks (MANET) considering the network mobility characteristics. However, in existing algorithms, it is assumed that the mobility parameters of the networks are fixed, while they are stochastic and vary with time indeed. Therefore, the proposed clustering algorithms do not scale well in realistic MANETs, where the mobility parameters of the hosts freely and randomly change at any time. Finding the optimal solution to the cluster formation problem is incredibly difficult, if we assume that the movement direction and mobility speed of the hosts are random variables. This becomes harder when the probability distribution function of these random variables is assumed to be unknown. In this paper, we propose a learning automata-based weighted cluster formation algorithm called MCFA in which the mobility parameters of the hosts are assumed to be random variables with unknown distributions. In the proposed clustering algorithm, the expected relative mobility of each host with respect to all its neighbors is estimated by sampling its mobility parameters in various epochs. MCFA is a fully distributed algorithm in which each mobile independently chooses the neighboring host with the minimum expected relative mobility as its cluster-head. This is done based solely on the local information each host receives from its neighbors and the hosts need not to be synchronized. The experimental results show the superiority of MCFA over the best existing mobility-based clustering algorithms in terms of the number of clusters, cluster lifetime, reaffiliation rate, and control message overhead.

Keywords MANET · Network clustering · Learning automata

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

1 Introduction

Dynamic network topology changes, network mobility, severe constraints on network resources such as communication channel bandwidth, processing power, and battery life, and the lack of a fixed infrastructure or centralized administration are the major challenging issues from which the ad hoc networking protocols suffer. A mobile ad-hoc network is a self-organizing and self-configuring multi-hop wireless communication network supporting a collection of mobile hosts which can be instantly developed in situations where either a fixed infrastructure is unavailable (e.g., disaster recovery), or a fixed infrastructure is difficult to install (e.g., battlefields). Two hosts can directly communicate if they are within the transmission range of each other and otherwise (i.e., when the source can not directly send the packets to the destination due to the limitation of the radio transmission range) the intermediate hosts assume the role of router and relay the packets toward the final destinations. Hence, each node in a MANET acts as both host and router. Besides the multi-hop nature of the MANET and the lack of a fixed infrastructure, these networks inherit the traditional problems of the wireless and mobile communication systems. Host mobility brings about a wide range of new challenges in the design of the MANET protocols. Frequent and hard to predict topology changes due to the host mobility,

and sever resource constraints are the most important issues must be taken into consideration in mobile ad-hoc networking [14, 22, 28].

In MANETs, the network performance is significantly degraded as the network size grows, and network clustering is one of the promising solutions. The main idea behind the network clustering is to group together the network hosts that are in physical proximity. In a clustered network, some of the hosts assume the role of a cluster-head and the others (that can directly communicate with the cluster-heads) act as a cluster member. On behalf of the cluster members, the cluster-heads are responsible for basic functions such as channel access scheduling, power measurements, and coordination of intra and inter-cluster communications, and this imposes an extra overhead on the cluster-head [8, 31–33]. After such a network partitioning, some cluster members may reach more than one cluster-head. These hosts which are called gateway are used for inter-cluster communications. In MANETs, due to the mobility and failure of the hosts, the network topology frequently changes and so the clusters may rapidly lose their validity. In this case, the network must be clustered again, and reclustering procedure consumes too much energy and bandwidth which are scarce resources in MANET. The best, but very difficult, solution is to construct the resistant-to-mobility clusters [13, 15, 20]. To achieve this, recently many studies [10–12, 16, 22, 24, 27] have been conducted on mobility-based network clustering techniques. Generally, these works aim to form the clusters with the maximum stability against the host mobility. The main problem with the above mentioned cluster formation algorithms is that they do not scale well in realistic mobile ad-hoc environments where each mobile host is free to change its mobility parameters randomly at any time. This is due to the fact that the proposed clustering techniques assume the mobility characteristics of the hosts are deterministic, while they are stochastic and vary with time. In real ad-hoc networks, no assumption can be made on the probability distribution function of the mobility parameters. In fact, the mobility parameters in MANET are random variables with unknown probability distributions. Under such circumstances, finding an optimal solution to form the stable (resistant to host mobility) clusters is incredibly hard.

In this paper, we propose a learning automata-based weighted clustering algorithm for wireless MANET in which the weight associated with each host is defined as its expected relative mobility which is assumed to be a random variable with unknown distribution. The proposed clustering solution is a fully distributed algorithm that can be independently run at each host. In this method, each mobile host based solely on the local information received from its one-hop neighbors chooses the adjacent host having the minimum expected relative mobility as its cluster-head. One major advantage of the proposed technique is that the hosts need not to be synchronized for cluster formation, and this allows the cluster maintenance can be locally performed only where it is required. The proposed algorithm taking advantage of learning automata assures that the cluster-head selected at each neighborhood has the minimum relative mobility with respect to all its members. To show the performance of the proposed algorithm, we have conducted several simulation experiments and compared the obtained results with those of HD [21], GDMAC [22], MOBIC [11], and MobHiD [24]. The experimental results show that the proposed algorithm outperforms the others in terms of the number of clusters, cluster lifetime, reaffiliation rate, and control message overhead.

The rest of the paper is organized as follows. In the next section, a brief review of the cluster formation algorithms is first provided, learning automata theory is presented in a nut shell, and the expected relative mobility criterion is introduced finally. In Sect. 3, a mobility-based learning clustering algorithm is proposed for wireless mobile ad-hoc networks based on learning automata. In Sect. 4, the performance of the proposed algorithm is evaluated through simulation experiments, and Sect. 5 concludes the paper.

2 Backgrounds and preliminaries

To provide a sufficient background for understanding the basic structures of the cluster formation algorithm which is proposed in this paper and its superiority over the existing methods, in this section we first review the well-known cluster formation techniques reported in the literature. Then, we provide a brief overview of the learning automata theory and introduce the expected relative mobility criterion.

2.1 Related work

The host mobility is the main origin of the difficulties with the design of efficient ad hoc networking protocols. During the last decade, due to tremendous growth of the wireless mobile ad hoc networks, many efforts have been done to design the mobility-based networking protocols for MANET. Designing the resistant-to-mobility cluster formation protocols is one of the issues that has been extensively considered in the literature. The following presents a brief overview of the best existing mobility-based cluster formation algorithms.

Lin and Gerla [25] proposed a straightforward network clustering algorithm called Lowest ID (LID) in which the node with the lowest identification number at each neighborhood has the highest priority to be selected as the cluster-head. The neighboring nodes with higher IDs assume the role of cluster members and form the cluster. This cluster-head selection procedure is repeated for the remaining nodes

until either each node is selected as a cluster-head or a cluster member. The lowest ID is known to be a two-hop cluster formation algorithm, since the distance between each node and every other node in a cluster is at most two hops. In [9], a synchronous distributed clustering algorithm was proposed by Baker and Ephremides in which each node having the highest ID among all its neighbors is selected as cluster-head. In the proposed algorithm, hereafter is referred to as HID, the number of nodes in the network is assumed to be known a priori. Gerla and Tsai [21] proposed another priority-based clustering technique called HD in which the priority of each host is defined as its degree. The degree of a host is defined as the number of its one-hop neighbors. In this algorithm, the host with the highest degree (priority) among all its adjacent hosts is selected as cluster-head. Experimental results show that the number of clusters into which the network is partitioned in LID and HID is much more than that in HD. This is due to the fact that LID and HID do not take into account the network topological specifications for determination of the hosts' priorities. Since a larger number of hosts can be dominated by the cluster-head having the highest degree, HD [21] significantly reduces the number of clusters. The main problem with HD is the frequent cluster-head changes due to the host mobility. A host of proposed clustering algorithms are based on these three fundamental techniques. In [3], Akbari Torkestani and Meybodi proposed a cluster formation algorithm based on distributed learning automata for wireless ad hoc networks. In this paper, the authors show that finding the weakly connected dominating set (WCDS) of the network topology graph is a promising approach for network clustering. They first propose a centralized approximation algorithm called DLA-CC for solving the minimum WCDS problem. They also propose a distributed implementation of DLA-CC, called DLA-DC, for clustering the ad hoc networks in which the dominator nodes and their closed neighbors assume the role of the cluster-heads and cluster members, respectively. DLA-DC is composed of a number of stages, and at each stage a cluster-head set (or WCDS) is found. At each stage, if the size of the cluster-head set is smaller than that of the minimum cluster-head set found so far, it is rewarded and set to the minimum cluster-head set, otherwise it is penalized.

Basagni [10] generalized the algorithm proposed by Gerla and Tsai [21], by using a generic weight as a criterion for cluster-head selection. In the proposed distributed and mobility adaptive clustering (DMAC) algorithm, a weight is associated with each host in the network that indicates the suitability of the host for selecting as a cluster-head. This weight could be computed as the residual energy or the mobility speed of a host. In DMAC, all hosts are initially in an *undecided* state and in the course of algorithm decide to be a cluster-head or a cluster member. A host decides when all its neighbors with larger weight have decided. When the time comes, a host decides to be cluster member if one of its neighbors is cluster-head. Otherwise, it decides to be cluster-head. This simple scheme can be somewhat optimized by letting hosts to be cluster member as soon as a neighbor gets cluster-head. The cluster-heads form a maximal independent set (i.e., the minimal dominating set) by which the network is clustered. In [22], Ghosh and Basagni investigated the impact of the different mobility degrees and mobility patterns of the mobile hosts on the performance of DMAC protocol. They showed that the cluster reorganization rate of DMAC protocol considerably increases in the presence of the host mobility. To alleviate the negative effects of the host mobility on the performance of DMAC, Ghosh and Basagni [22] proposed a generalization of DMAC protocol called GDMAC in which the clusters are more stable against the host mobility. GDMAC dramatically reduces the rate of unnecessary cluster updates by applying the following two limiting rules. The first rule controls the rate of reclustering and says that a cluster reorganization is required only when the weight of the new cluster-head exceeds the weight of its current one. The second rule controls the spatial density of the cluster-heads. Experimental results reported in [22] show the superiority of GDMAC over DMAC in terms of the cluster reorganization and reaffiliation rate. A reaffiliation happens when a host leaves its current cluster and joins to a new cluster. Basu et al. [11] proposed a weighted clustering algorithm called MOBIC which is similar to DMAC. MOBIC makes use of a new mobility parameter called aggregate local mobility (ALM) to select the cluster-heads. In this method, the ALM of each host is computed based on the strength of the signals that it receives from its neighboring hosts. In this algorithm, a host is elected as a cluster-head, if it has the minimum ALM among its adjacent hosts. Chatterjee et al. [12] proposed a mobility–based clustering algorithm in which the average speed of each mobile host, which is measured by tracking the changes in the position coordinates, is defined as its weight. In the proposed method, the hosts with the lowest weight (average mobility speed) are more likely to assume the role of a cluster-head.

Palit et al. [27] proposed a mobility-aware pro-active low energy (MAPLE) protocol for clustering the wireless ad-hoc networks. The proposed technique exploits the host mobility information to design a proactive energy-efficient clustering protocol. In this method, the mobility pattern of the wireless hosts and the cost associated with the wireless links are estimated based on the radio link level information. MAPLE also considers the problem of cluster formation in a medium access control (MAC) layer, and uses a channel reservation technique to reduce the contention rate among the hosts in the course of cluster formation. In MAPLE, each host estimates its distance from its cluster-head by the signal strength received from the cluster-head. This estimation is used by

the host to make a proactive decision on cluster reaffiliation. That is, a host joins another cluster, if it predicts that it would be out of the current cluster in the near future. However, MAPLE does not take into consideration the host mobility in the course of the cluster-head selection phase. Er and Seah [16] proposed a mobility-based d-hop (MobDHop) clustering algorithm that partitions the mobile adhoc networks into variable-diameter clusters based only on the mobility information each host receives from its one-hop neighbors. To improve the cluster stability, MobDHop groups the hosts with the similar mobility pattern together. For this purpose, the following three mobility metrics are used: variation of estimated distance between nodes over time, local variability, and group variability. In this method, all hosts periodically broadcast Hello messages. Each host then calculates the estimated distance based on the strength of the received signals. The variability of the estimated distance (VED) is computed for each host with respect to all its neighbors. In the proposed method, the host with the lowest VED among its neighbors has the most stability and so is selected as cluster-head. Konstantopoulos et al. [24] proposed a novel mobility-aware technique for cluster formation and maintenance called MobHiD. The main idea behind MobHiD is to predict the future mobility behavior of the hosts so as to form the most stable clusters. In [24], the authors define a new mobility prediction scheme as the probability of having the same neighbors for sufficiently long time. This probabilistic measure exhibits the stability of the host among its neighbors, and indicates the suitability of the host for electing as a cluster-head. In fact, this mobility metric computes the variability of the neighborhood of the hosts over time. The proposed mobility prediction method is then combined with the highest degree clustering technique given in [21] to partition the network into a minimum number of stable clusters.

2.2 Learning automata theory

A learning automaton [26, 29, 30] 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. Figure 1 shows the relationship between the learning automaton and random environment.

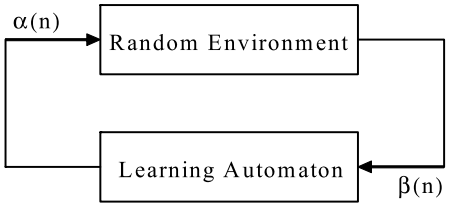

Fig. 1 The relationship between the learning automaton and its random environment

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 that 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 [26]: fixed structure learning automata and variable structure learning automata. Variable structure learning automata are represented by a triple $\langle \underline{\beta}, \underline{\alpha}, T \rangle$, 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 p is updated. Let $\alpha_i(k)$ be the action chosen by the automaton at instant k.

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

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

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

when the taken action is penalized by the environment (i.e., $\beta(n) = 1$), r is the number of actions that can be taken by learning automaton, $a(n)$ and $b(n)$ denote the reward and

penalty parameters and determine the amount of increases and decreases of the action probabilities, respectively. If $a(n) = b(n)$, the recurrence equations (1) and (2) are called linear reward-penalty (L_{R-P}) algorithm, if $a(n) \gg b(n)$ the given equations are called linear reward-ϵ penalty ($L_{R-\epsilon P}$), and finally if $b(n) = 0$ 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.

Learning automata have been found to be useful in systems where incomplete information about the environment, wherein the system operates, exists. Learning automata are also proved to perform well in dynamic environments. It has been shown in [1, 2, 7, 34–36] that the learning automata are capable of solving the NP-hard problems. Recently, several learning automata-based protocols have been designed for improving the performance of the wireless ad hoc networks [4–6, 37, 38] or sensor networks [17–19].

2.3 Expected relative mobility

The mobility speed and movement direction of a host are the parameters upon which the relative mobility of a host can be defined. Let v_i^t and θ_i^t denote the mobility speed and movement direction of host H_i at instant t, respectively. The relative mobility of host H_i with respect to host H_j at instant t is defined as

$$RM_{(i,j)}^t = \sqrt[2]{(v_i^t)^2 + (v_j^t)^2 - [2v_i^t v_j^t \cos(\theta_i^t - \theta_j^t)]} \tag{3}$$

v_i^t and θ_j^t denote the mobility characteristics (speed and direction) of host H_j.

In real life applications, since the mobility characteristics of the mobile host vary with time, the relative mobility of the host (with respect to the adjacent hosts) changes in different epochs, and so the instant relative mobility (the relative mobility determined for a single epoch) can not exhibit the realistic mobility behavior of the mobile host. Therefore, for a true prediction of the mobility parameters, the expected relative mobility is defined as the average of the instant relative mobilities over different epochs. The expected relative mobility between two hosts H_i and H_j is defined as

$$ERM_{(i,j)}^T = \frac{1}{k} \sum_{t=1}^{k} RM_{(i,j)}^t \tag{4}$$

where T denotes the time period over which the relative mobility is averaged, and k is the number of times each host and its neighbors experience new epochs during time interval T. As time interval T becomes larger, the expected relative mobility estimated by (3) gets closer to the mean of relative mobility distribution between hosts H_i and H_j. As stated earlier, an epoch is defined as a time interval during which both the mobility speed and the movement direction of a mobile host are constant. A host enters a new epoch, once one (or both) of these parameters changes. Each mobile host (e.g. H_i) broadcasts its mobility information (average mobility speed and movement direction) to its neighboring hosts immediately, if it experiences a new epoch. Then, each neighboring host (e.g., H_j) calculates its new relative mobility, on the basis of the recently received mobility information as shown in (4).

To extend the clustering lifetime and to reduce the reaffiliation rate of the hosts, our cluster formation algorithm selects a mobile host as a cluster-head, if it has the minimum expected relative mobility with respect to all its neighbors. Hence, we define the relative mobility of a given host H_i with respect to all its neighbors as

$$ERM_i^T = \frac{1}{|N_i|} \sum_{\forall H_j \in N_i} ERM_{(i,j)}^T \tag{5}$$

where N_i is the set of all neighbors of host H_i.

ERM_i^T is assigned to each host H_i as a weight. It denotes the expected mobility degree that host H_i exhibits with respect to all its neighbors. A host with a higher mobility degree is more prone to the unstable mobility behaviors. Therefore, to prolong the lifetime of the clusters, the proposed clustering algorithm attempts to select the hosts with less relative mobility as cluster-heads. In this algorithm, the expected relative mobility of each host (with respect to all its neighbors) is defined as its weight. This weight is updated as the host or one of its neighbors experience a new epoch.

3 Description of clustering algorithm

Due to the strict resource limitations and host mobility in MANETs, the performance of the network rapidly declines as the network size grows. Among the solutions proposed for solving the scalability problem of MANETs, the network clustering is a promising approach in which the adjacent hosts are grouped together in physical proximity and managed locally. In this section, a learning automata-based cluster formation algorithm is proposed for MANETs. In this algorithm, the expected relative mobility of each host with respect to all its neighbors is defined as a mobility criterion to find the most stable clusters against the network dynamics. In MANETs, the mobile hosts freely move anywhere at random, and so the even most stable structures lose their validity soon. To relieve the negative effects of the frequent network topology changes on the clusters (i.e., to keep the clusters up to date), the proposed algorithm offers a self-maintenance procedure to repair the damaged clusters. Therefore, the proposed algorithm is composed of two main parts. The first part is the initial clustering which is

performed as the network starts up, and the second one is the cluster maintenance which is performed whenever and wherever it is required.

3.1 Initial clustering

Let triple $\langle H, L, R\rangle$ describe the stochastic ad-hoc network, where $H = \{h_1, h_2, \ldots, h_n\}$ is the set of network hosts, $L = \{(h_i, h_j)\}$ denotes the set of links connecting every host pairs (h_i, h_j) where h_i and h_j are within the transmission range of each other, and $R = \{ERM_i | i = 1, 2, \ldots, n\}$ denotes the set of expected relative mobilities associated with the network hosts such that ERM_i is the expected relative mobility of host h_i. In this algorithm, a network of learning automata isomorphic to the stochastic ad-hoc network is initially formed by equipping each host h_i of the network with a L_{R-P} learning automaton A_i. The expected relative mobility of each host h_i with respect to all its neighbors is also defined as the weight associated with learning automaton A_i. The resulting network can be described by a triple $\langle \underline{A}, \underline{\alpha}, \underline{W}\rangle$, where $\underline{A} = \{A_1, A_2, \ldots, A_n\}$ denotes the set of the learning automata corresponding to the network hosts, $\underline{\alpha} = \{\underline{\alpha_1}, \underline{\alpha_2}, \ldots, \underline{\alpha_n}\}$ denotes the set of action-sets of learning automata, and $\underline{W} = \{w_1, w_2, \ldots, w_n\}$ denotes the set of weights associated with network hosts such that w_i is the expected relative mobility of hosts h_i which is assumed to be a positive random variable with unknown probability distribution. The set of actions that can be taken by learning automaton A_i (corresponding to host h_i) is denoted as $\underline{\alpha_i}$ and defined as

$$\underline{\alpha_i} = \{\alpha_i^j | h_j \text{ is adjacent to } h_j \text{ or } i = j\}$$

The action-set of learning automaton A_i includes an action, α_i^j, for each of its neighboring host h_j (or its adjacent learning automaton A_j) and an action for h_i itself. To form the action-set, each host sends a *ASFREQ* (i.e., Action-Set Formation REQuest) message to all its one-hop neighboring hosts. The hosts which are within the transmission range of the sender host, upon receiving the *ASFREQ* message reply it. Each host waits a short period of time for the reply messages and then constructs its action-set as follows: Host h_i adds an action α_i^j to its action-set for each host h_j from which it receives the reply message. It also adds an action for itself (i.e., α_i^j, where $i = j$). The action-set of each learning automaton A_i comprises the hosts that can be selected by host h_i as a cluster-head. Choosing action α_i^j by host h_i means that host h_i selects host h_j as its cluster-head. Therefore, the here considered action-set formation method means that each host either selects one of its neighbors as its cluster-head or declares itself as a cluster-head to its neighbors. At first, all the actions are chosen with the same probability. For host h_i, the probability of choosing each action is initialized to $\frac{1}{\Delta_i+1}$, where Δ_i is the degree of host h_i. This probability increases, if an action is rewarded and decreases otherwise.

In the proposed algorithm, each host can be in one of three states, *UN* (unknown), *CH* (cluster-head) and *Cm* (cluster member). All hosts are initially set to an *UN* state. Each host changes its state to *CH*, if it is selected by itself or at least one of its neighboring hosts as a cluster-head, else its state is set to *CM*. The proposed algorithm consists of a number of stages, and at each stage, each host independent of the other hosts elects its cluster-head among its one-hop neighbors or itself assumes the role of a cluster-head. Therefore, the proposed cluster formation algorithm guarantees to cluster the entire network at each stage. At each stage, the proposed algorithm tries to select the host having the minimum expected relative mobility (with respect to all its neighbors) at each neighborhood as cluster-head. It modifies the prior selection as soon as it finds another host with a lower expected relative mobility in the next stages. Therefore, as the proposed algorithm proceeds, the most stable clusters (against the network mobility) with the minimum expected relative mobility are found. In the following, we describe a stage of the proposed algorithm for host h_i.

Learning automaton A_i corresponding to host h_i randomly chooses one of its possible actions according to its action probability vector. The host corresponding to the selected action is defined as its cluster-head. Let us assume that ch_i denotes the cluster-head selected by host h_i at the current stage. According to the action-set formation method described earlier, each host may choose one of its adjacent hosts as its cluster-head or itself assumes the role of a cluster-head. If the selected cluster-head (i.e., ch_i) is one of its neighbors, host h_i sends a request message called *RERM* (short for request to expected relative mobility) to ch_i for receiving its current expected relative mobility. *RERM* message includes the ID number of the sender host (sender ID), h_i, and the ID number of the selected cluster-head (cluster-head ID), ch_i). Host h_i then waits for a reply message. Upon receiving a *RERM* message, each host checks the message to see if its ID is equal to the cluster-head ID. If so, it sends its expected relative mobility calculated as given in (5) to the sender host and drops it otherwise. Let $ERM_{ch_i}^k$ denotes the expected relative mobility associated with cluster-head ch_i until kth stage (i.e., the cluster-head selected by host h_i). Host h_i sets w_i to the expected relative mobility of its selected cluster-head $ERM_{ch_i}^k$. No *RERM* message is required, if host h_i selects itself as cluster-heads. In this case, w_i is set to $ERM_{h_i}^k$. Then, host h_i compares weight w_i (the expected relative mobility of the selected cluster-head) with a dynamic threshold T_i^k. The action selected by learning automaton A_i is rewarded as given in (1), if weight w_i is less than or equal to the dynamic threshold T_i^k. Otherwise, the selected action is penalized by using (2). At stage k, dynamic threshold T_i^k is computed as the average expected

Algorithm *MCFA* (h_i, P) The proposed cluster formation algorithm

01:**Input:** Host h_i, Stop condition P
02:**Begin** algorithm
03:**Assumptions**
04: All hosts are initially set to a *UN* state
05: Let ch_i denotes the cluster-head selected by h_i
06: Let w_i be the weight of the selected cluster-head ch_i
07: Let k be the stage number
08: Let T_i^k be the dynamic threshold associated with h_i at stage k
09: $T_i^k \leftarrow 0$, $k \leftarrow 0$
10:Host h_i forms its action-set as $\underline{\alpha_i} = \{\alpha_i^j\}$, where h_j is adjacent to h_i or $i = j$
11:**Repeat**
12: Host h_i randomly chooses one of its actions as its cluster-head (say ch_i)
13: **If** $ch_i \neq h_i$ **Then**
14: Host h_i sends a *RERM* message to ch_i
15: $w_i \leftarrow ERM_{ch_i}^k$
16: **Else**
17: $w_i \leftarrow ERM_{h_i}^k$
18: **If** $w_i \leq T_i^k$ **Then**
19: Host h_i rewards the action corresponding to ch_i
20: **Else**
21: Host h_i penalizes the action corresponding to ch_i
22: Host h_i computes T_i^k using Equation (6)
23: $k \leftarrow k + 1$
24:**Until** the probability with which host h_i chooses a cluster-head is greater than P
25:**If** $ch_i = h_i$ **Then** {*if host* h_i *assumes the role of cluster-head itself*}
26: Host h_i changes its state to *CH*
27:**Else**
28: Host h_i sends a *CHSEL* message to final cluster-head ch_i
29:**End** algorithm

Fig. 2 The pseudo code of the proposed cluster formation algorithm

relative mobility of host h_i and its one-hop neighbors. T_i^k is defined as

$$T_i^k = \frac{1}{\Delta_i + 1} \sum_{\forall h_j;(h_i,h_j)\in L \text{ or } i=j} ERM_{h_j}^k \tag{6}$$

where Δ_i denotes the degree of host h_i. Figure 2 shows the pseudo code of the proposed clustering algorithm which is run at host h_i.

After updating the action probability vectors, each host computes the dynamic threshold for the next stage as given in (6). Host h_i begins a new iteration, if one of its adjacent hosts or itself experiences new epoch. This process continues until the probability with which host h_i chooses a cluster-head exceeds a predefined threshold (i.e., P). Threshold P is called the stop condition. For each host, the cluster-head which is chosen just before the stop condition is satisfied is declared as its final cluster-head. After the stop condition is met, each host sends a *CHSEL* (i.e., Cluster-Head SELection) message to its final cluster-head, if the selected cluster-head is one of its neighbors. Otherwise, it changes its state to *CH* and plays the role of a cluster-head thereafter. Upon receiving a *CHSEL* message, each host calls procedure *CHSEL* given in Fig. 3. In this procedure, each host first checks to see if its ID number is equal to the selected cluster-head ID number contained in *CHSEL* message. If so, the host assumes the role of a cluster-head and changes its state to *CH* when it receives such a message for the first time. It then also adds the ID number of the sender host to the list of its cluster members. A host changes its state to *CM*, if no host selects it as a cluster-head. For each network host, the information by which it chooses its cluster-head is confined only to its one-hop neighbors, and so the proposed algorithm can be localized at each host well. Figure 3 shows the pseudo code of procedure *CHSEL*, where host h_j sends a *CHSEL* message to host h_i. In this procedure, it is assumed that ch_j is the cluster-head selected by host h_j.

Procedure *CHSEL* (h_i)

01:**Begin** procedure
02:**If** $h_i = ch_j$ **Then**
03: Host h_i changes its state to *CH*
04: Host h_i adds the sender ID to the list of its cluster members
05:**Else**
06: Host h_i changes its state to *CM*
07:**End if**
08:**End** procedure

Fig. 3 Pseudo code of procedure *CHSEL*

Procedure *JREQ* (h_i)

01:**Begin** procedure
02:**If** h_i receives a *JREQ* message from host h_k **Then**
03: $\Delta_i \leftarrow \Delta_i + 1$
04: **For** all $\alpha_i^j \in \underline{\alpha_i}$ **do**
05: $p_i^j \leftarrow \frac{\Delta_i}{\Delta_i+1} \cdot p_i^j$
06: $\underline{\alpha_i} \leftarrow \underline{\alpha_i} + \{\alpha_i^k\}$ $\{h_i$ *creates a new action*$\}$
07: $p_i^k \leftarrow \frac{1}{\Delta_i+1}$
08: **End for**
09:**End if**
10:**If** h_i is a cluster-head **Then**
11: Host h_i sends back host h_k a *JREP* message
12:**End if**
13:**End** procedure

Fig. 4 Pseudo code of procedure *JREQ*

3.2 Cluster maintenance

As stated earlier, in wireless mobile ad hoc networks, due to the host mobility and failures, the network topology frequently changes. In these networks, the intra-cluster links are fragile, and so the clusters are decomposed after a short while. The major superiority of the proposed clustering algorithm over the other methods is that it can be independently and locally run at each host. In existing cluster formation techniques, the normal operation of the network must be stopped until the reclustering process is completed. In this algorithm, the cluster-head selection (reclustering) process can be performed only for a single host or for a group of hosts, while the others continue their normal operation. Since the mobile hosts are free to move anywhere in MANETs, they may leave a cluster or join to another one at any time. Therefore, the cluster membership is highly dynamic due to the frequent topology changes of these networks. Another advantage of the proposed cluster formation and maintenance technique is that during the reclustering phase, each host quickly converges to its new optimal cluster-head. This is because during the initial clustering, the choice probability of the candidate cluster-heads (for a given host) grows proportional to their optimality, and so in the absence of the optimal cluster-head, algorithm rapidly converges to the second (or sub-) optimal cluster-head which have the highest choice probability.

3.2.1 Joining the network

When a new host joins the network or a host can not be connected to its cluster-head any longer, it must be affiliated with a cluster-head in its neighborhood. In our proposed cluster maintenance procedure, when a mobile host receives no response from its cluster-head or when a host joins the network for the first time, it sends a *JREQ* (i.e., Join REQuest) message to its one-hop neighboring hosts. Each adjacent host calls procedure *JREQ* shown in Fig. 4 upon receiving the *JREQ* message. Each cluster-head which receives this message replies it by sending back a *JREP* (i.e., Join REPly) message. A *JREP* message includes the ID numbers of the sender host and the cluster-head (which replies the join request) as well as the expected relative mobility of the replying cluster-head. The sender host waits for a while to

Procedure *LREQ* (h_i)

01:**Begin** procedure
02:**If** h_i receives a *LREQ* message from host h_k **Then**
03: $\Delta_i \leftarrow \Delta_i - 1$
04: $\underline{\alpha_i} \leftarrow \underline{\alpha_i} - \{\alpha_i^k\}$ $\{h_i$ *removes the action corresponding to* $ch_k\}$
05: **For** all $\alpha_i^j \in \underline{\alpha_i}$ **do**
06: $p_i^j \leftarrow p_i^j + p_i^j \cdot \frac{p_i^k}{1-p_i^k}$
07: **End for**
08:**End if**
09:**If** host h_k is a cluster-head **Then**
10: Host h_i calls $MCFA(h_i, P)$
11:**End if**
12:**End** procedure

Fig. 5 Pseudo code of procedure *LREQ*

receive the *JREP* messages, and then selects its cluster-head as follows:

(I) If the sender host receives no *JREP* messages, it chooses the neighboring host with the minimum expected relative mobility as cluster-head. If the sender is a newly joining host, to acquire the mobility information of the neighboring hosts, it must first send a *RERM* (i.e., Request to Expected Relative Mobility) message to all its neighbors and then chooses its cluster-head. After choosing the cluster-head, a *CHSEL* message is sent to it. The selected cluster-head changes its state to *CH*, if it has not yet changed it. Cluster-head then adds the sender ID to its list.

(II) If the sender host (i.e., the newly joining host or the host that can not be connected to its cluster-head) receives a *JREP* message, it chooses the sender of the *JREP* message as its cluster-head, and sends a *CHSEL* message to it.

(III) If the sender host receives more than one *JREP* messages, it chooses the sender host with the minimum expected relative mobility as its cluster-head, and sends it a *CHSEL* message. In this case, if the sender host is a newly joining host, it must acquire the mobility information of its adjacent hosts as described in (I) prior to choosing the cluster-head.

In this algorithm, when a host joins (or leaves) the network or a cluster, the action-set of the hosts which are (or have been) within its radio transmission range must be updated. As described in Sect. 3.1, the action-set of learning automaton A_i (associated with host h_i) is defined as $\underline{\alpha_i} = \{\alpha_i^j | h_j$ is adjacent to h_i or $i = j\}$. Let $\underline{p_i} = \{p_i^j | \forall j; i = j$ or $h_i\}$ denotes the action probability vector of the learning automaton A_i. p_i^j is the probability with which host h_i selects h_j as its cluster-head. If the newly joining host is within the transmission range of host h_i, the forwarding *JREQ* message is heard by h_i. In this case, host h_i updates its action probability vector by calling procedure *JREQ*.

3.2.2 *Leaving the network*

When a host decides to leave the network or a cluster, two different approaches must be taken depending on whether the host is a cluster-head or a cluster-member. If a mobile host decides to leave the network, it sends a *LREQ* (i.e., Leave REQuest) message to all its one-hop neighbors. Each host checks to see if the leaving host is its cluster-head, upon receiving a *LREQ* message. If so, the members by which this host has been selected as cluster-head (i.e., its cluster members) must be clustered again. Whether the leaving host is a cluster-head or a cluster member, each adjacent host that hears the *LREQ* message must keep its action probability vector up to date. To do so, each neighboring host removes the action corresponding to the leaving host and distributes the choice probability of the leaving host between the remaining hosts proportional to their probability values (see Lines 04–07 of procedure *LREQ*). If the leaving host is a cluster member no more action is required. However, if the leaving host is a cluster-head, each (orphan) cluster member must be reaffiliated with another cluster-head. Since the proposed cluster formation algorithm is executed at each host independent of the others and locally, it can be also used for reclustering some portions of network as a partial clustering. Therefore, after updating the action probability vectors, each host calls algorithm MCFA for exploring a new cluster-head, if it receives a *LREQ* from a cluster-head. Figure 5 shows the procedure which is run by each host upon receiving a *LREQ* message.

4 Experimental results

To study the performance of the proposed cluster formation algorithm, several simulation experiments have been conducted in NS2. In these experiments, a mobile ad-hoc network is modeled by randomly and uniformly distributing N (where N ranges from 50 to 100) mobile hosts within a square simulation area of size 1000 m × 1000 m. The mobility speed of each host changes from 1 km/h to 80 km/h. IEEE 802.11 DCF [23] (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 communication channel through which the wireless hosts communicate is a common broadcast channel of capacity 2 mb/s. All mobile hosts are equipped with omnidirectional antennae and have the same radio propagation range changing from 100 m to 200 m. CBR (Continuous Bit Rate) traffic sources are used to generate the traffics with a rate of 20 p/s (packet per second). The packet size is set to 512 bytes. Each host is modeled as a store-and-forward queuing station and assumed to be aware of its mobility information with the aid of a reliable positioning system. Each simulation experiment is run for 20 min, and the obtained results are averaged over 100 different runs. For our proposed algorithm, the reward and penalty parameters are set to 0.1, and the stop condition P is set to 0.95.

To show the outperformance of the proposed cluster formation algorithm, its results are compared with those of HD (a priority-based cluster formation algorithm proposed by Gerla and Tsai [21]), GDMAC (a mobility-based clustering algorithm proposed by Ghosh and Basagni [22]), MOBIC (a weighted clustering algorithm proposed by Basu et al. [11]), and MobHiD (a neighborhood stability-based mobility prediction algorithm for clustering the MANETs proposed by Konstantopoulos et al. [24]). In conducted experiments, the efficiency of the proposed clustering algorithm is compared with the above mentioned algorithms in terms of the following metrics of interest.

- Number of clusters. This metric is defined as the number of partitions into which the entire network is divided. This metric is inversely proportional to the cluster size. As the cluster size increases, the load on the cluster-head becomes considerable, leading to issues with energy consumption, channel access scheduling, latency, contention, complexity and so on.
- Cluster lifetime. The time interval during which a given host assumes the role of a cluster-head in the cluster is defined as the cluster lifetime. This metric is also called cluster duration and shows the stability of the network clusters. Cluster lifetime is measured in second and reduces as the mobility speed of the hosts increase. This metric represents the resistance of the clustering algorithm against the host mobility and network dynamics.

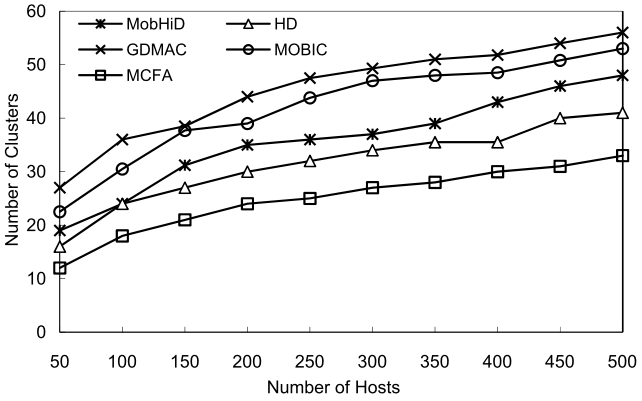

Fig. 6 The average number of clusters as a function of number of hosts for transmission range 100 m

- Reaffiliation rate. This metric shows how often a mobile host changes its cluster-head per unit time. Every ordinary (non cluster-head) host must be affiliated with a neighboring cluster-head. A host reaffiliates to another cluster-head, if the current cluster-head resigns its role or the host can not be connected to its cluster-head any more. It can be seen that reaffiliation rate is inversely proportional to the cluster lifetime.
- Control message overhead. This metric is defined as the number of (extra) control messages required for network clustering. This metric is measured as the number of control messages that must be sent per second.

4.1 Number of clusters

In this group of simulation experiments, we measure the number of clusters into which the network is partitioned as the number of network hosts changes from 50 to 500 with increment step of 50. The mobile hosts travel through the simulation area of size 1000 m × 1000 m with the random waypoint mobility model. We set the maximum mobility speed of each host to 70 km/h, and the radio transmission range of the hosts to 100 m. Figure 6 shows the average number of clusters (or cluster-heads) versus the network size (i.e., the number of mobile hosts within the network). From the results shown in Fig. 6, it is obvious that in all algorithms the number of clusters increases as the number of hosts increases. Comparing the curves shown in Fig. 6, we observe that GDMAC has the worse results, and the number of clusters constructed by MOBIC is slightly smaller than that of GDMAC. The reason for such a large number of clusters is that these algorithms do not aim at minimizing the number of clusters. The results also show that our proposed algorithm (MCFA) outperforms the other clustering algorithms. For number of hosts larger than 100, HD performs better than MobHiD and is ranked after MCFA.

We changed the radio transmission range to 200 m and repeated the same experiments. The results are shown in

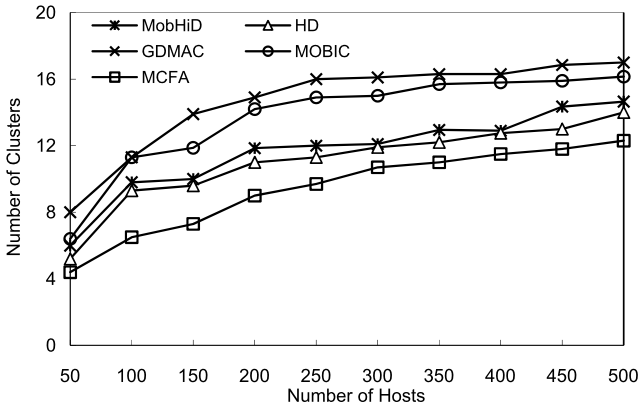

Fig. 7 The average number of clusters as a function of number of hosts for transmission range 200 m

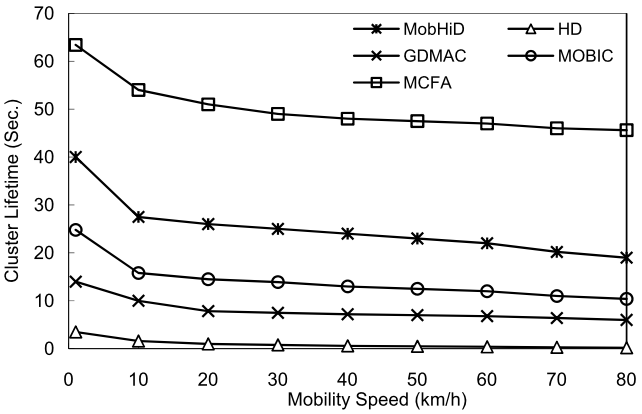

Fig. 8 Cluster lifetime versus the host speed for transmission range 200 m

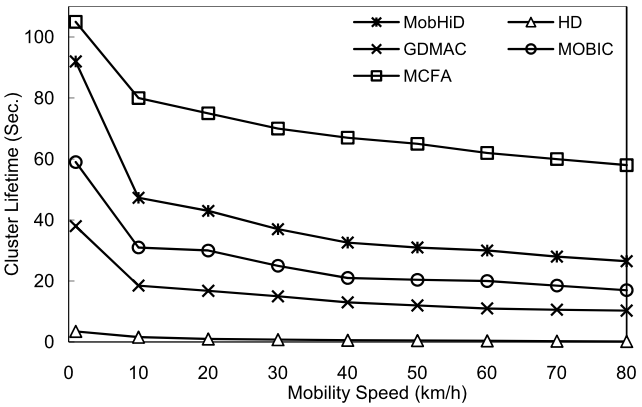

Fig. 9 Cluster lifetime versus the host speed for transmission range 300 m

Fig. 7. The obtained results show that the number of clusters significantly decreases as the radio transmission range increases. This is due to the fact that a larger area, and consequently a larger number of hosts, can be dominated by a cluster-head having a higher transmission range. Therefore, the entire network can be covered by a smaller number of cluster-heads. Comparing the results shown in Fig. 7, we find that the ranking given in Fig. 6 for the clustering algorithms remains unchanged here and the proposed algorithm outperforms the others and GDMAC produces the largest number of clusters.

4.2 Cluster lifetime

In these experiments, we study the impact of the host speed on the cluster duration. It is expected that the cluster lifetime reduces as the mobility speed increases. This set of simulation experiments is conducted on the ad hoc networks including 50 hosts each of radio transmission range 200 m. The average cluster lifetime is measured in seconds as the host speed ranges from 1 km/h to 80 km/h, and the obtained results are shown in Fig. 8. The curves depicted in Fig. 8 show the major superiority of the proposed clustering algorithm over the others in terms of the cluster stability. As mentioned earlier, the mobility characteristics of the host in the existing methods are assumed to be constant while they vary with time, and so they can not pragmatically predict the long-term mobility behavior of the host. In MCFA, the cluster members stay connected to the cluster-head for a longer time. This is because in MCFA the most stable hosts are found by estimating the expected mobility characteristics of the host over different epochs, and selected as cluster-head. Comparing the results shown in Fig. 8, we find that MobHiD lags far behind MCFA, and MOBIC is ranked below MobHiD. GDMAC outperforms HD and so it is ranked lower than MOBIC. Therefore, the clusters constructed by HD have the lowest stability against the host speed. This is because HD does not involve the mobility parameters of the host in cluster formation process. To study the effects of the radio transmission range on the cluster lifetime, we increased the transmission range to 300 m, repeated the same experiments, and reported the obtained results in Fig. 9. The results show that the cluster lifetime is meaningfully improved when we increase the transmission range. When the radio transmission range of the cluster-head increases, its neighboring hosts remain within its neighborhood (or stay connected to the cluster-head) for a longer time, and this extends the cluster lifetime.

4.3 Reaffiliation rate

This set of simulation experiments is concerned with investigating the impact of the speed and the radio transmission range of the hosts on the reaffiliation rate. Reaffiliation rate represents the number of times a mobile host changes its cluster-head and affiliates to a new cluster per unit time. A mobile host reaffiliates to a new cluster, if the current cluster-head resigns its role or the host can not stay connected to its cluster-head any longer. It is expected that the

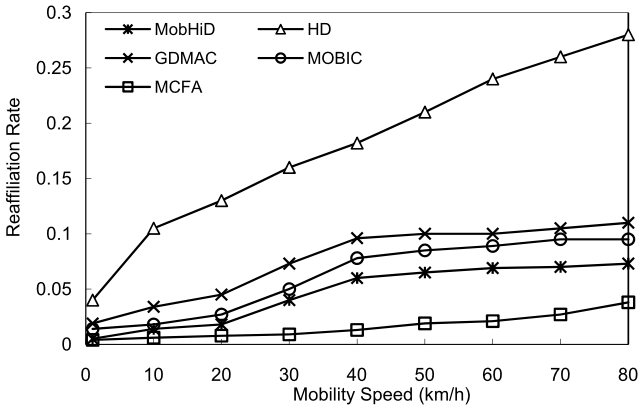

Fig. 10 Reaffiliation rate as a function of host speed

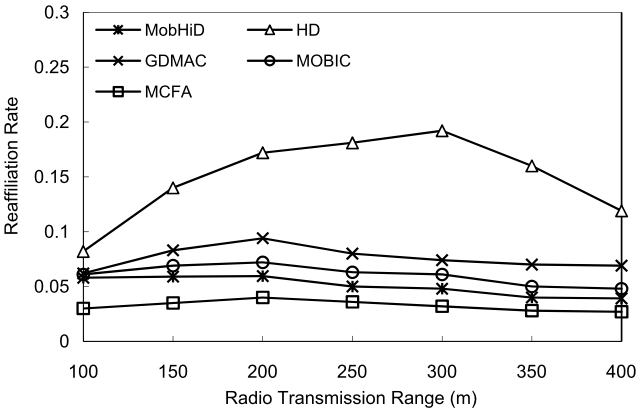

Fig. 11 Reaffiliation rate as a function of radio transmission range

reaffiliation rate increases as the mobility speed of the host increases. This is because the mobiles leave their clusters faster as the host speed increases. In these experiments, it is assumed that 50 mobile hosts, each of transmission range 200 m, are travelling within the simulation area of size 1000 m $\times$ 1000 m with random waypoint mobility model. The average reaffiliation rate is measured as the host speed changes from 1 km/h to 80 km/h. Figure 10 shows the average reaffiliation rate in second versus the changes of the host speed. The curves depicted in this figure shows that the reaffiliation rate of all algorithms increases when the host speed becomes faster. Comparing the results shown in Fig. 10, it is observed that our proposed algorithm has the lowest reaffiliation rate. This could be also predicted from the results obtained for the cluster lifetime. This is due to the fact that in our proposed algorithm the expected relative mobility of the cluster-head with respect to all its members is minimized. Therefore, the cluster-head duration increases and the mobile hosts leave their clusters at a very lower rate. From the obtained results, we find that among the previous algorithms, MobHiD has the lowest reaffiliation rate. This is because in MobHiD each host selects the host which remains its cluster-head for a longer time, and so it dramatically decreases the probability of reaffiliation. The results also show that HD has the highest reaffiliation rate as compared with the other algorithms. The reason for such a result is that HD does not take into account the movement characteristics of the host to improving the reaffiliation rate and to constructing the stable clusters.

We also conducted several experiments to examine the effects of the radio transmission range on the reaffiliation rate. The results are shown in Fig. 11. In these experiments, the maximum host speed is set 40 km/h, and the transmission range changes from 100 m to 400 m with increment step of 50 m. From the results shown in Fig. 11, we observe that for almost all algorithms the reaffiliation rate increases as the transmission range changes from 100 m to 200 m, and thereafter it decreases for transmission ranges larger than 250 m. This is because for small radio transmission ranges, between 100 m and 200 m, the host degree is small (i.e., each host has a few neighbors) and so the hosts rarely find another cluster-head to affiliate. On the other hand, for large radio ranges, between 250 m and 400 m, the time interval during which a host stays connected to (or is within the transmission range of) its cluster-head becomes longer as the transmission range increases. Therefore on two ends of the spectrum, the minimum reaffiliation rates can be achieved, and the maximum reaffiliation rate are obtained in the middle of the spectrum. The curves depicted in Fig. 11 show that MCFA outperforms the other algorithms in terms of the reaffiliation rate for all values of radio range.

4.4 Control message overhead

In these experiments, the control message overhead of the proposed clustering algorithm is measured and the obtained results are compared with those of GDMAC, MOBIC, and MobHiD. The control message overhead is measured as the number of control messages that must be sent per second. In this group of simulation experiments, 50 mobile hosts are randomly and uniformly distributed within the simulation area of size 1000 m $\times$ 1000 m. The mobility model is assumed to be random waypoint. The radio transmission range is initially set to 200 m and the control message overhead is measured as the host speed ranges from 1 km/h to 80 km/h. Then, the radio transmission range increases to 300 m and the same experiments are repeated. The obtained results are depicted in Figs. 12 and 13. The experiments show that the control message overhead of HD is much higher in comparison with the other algorithms. Therefore, to show the obtained results and performance of the other algorithms with more clarity (in a larger scale) HD was excluded from Figs. 12 and 13. From the obtained results, it can be seen that in all algorithms the control message overhead increases as the host mobility increases. Comparing the results shown in Figs. 12 and 13, we observe that the control message overhead significantly decreases as the radio range increases.

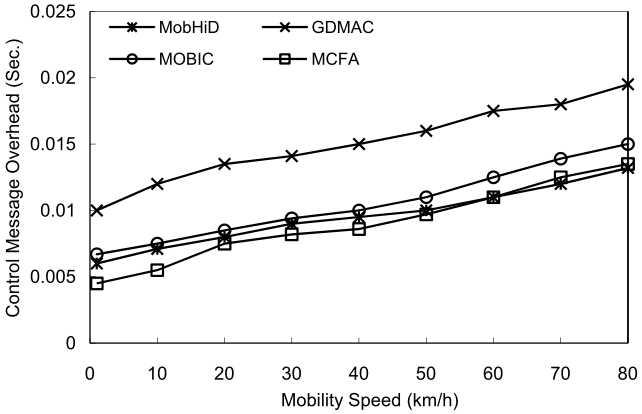

Fig. 12 Control message overhead versus the host speed for transmission range 200 m

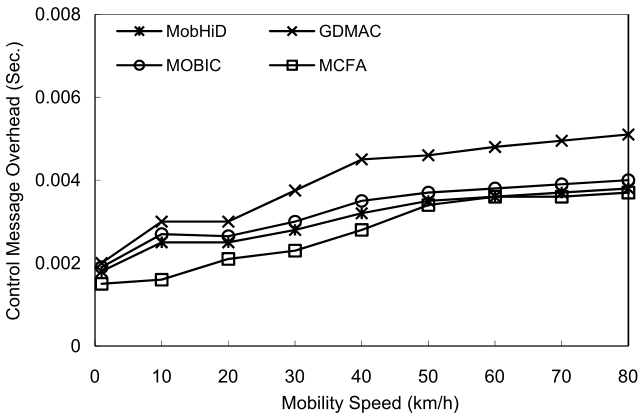

Fig. 13 Control message overhead versus the host speed for transmission range 300 m

This is because the cluster lifetime increases and the reaffiliation rate decreases as the radio transmission range increases. As shown in Figs. 12 and 13, GDMAC and MCFA have the highest and lowest control message overhead, respectively. The reasons for outperformance of MCFA are as follows. In comparison with the other algorithms, MCFA has the longest cluster duration and the lowest reaffiliation rate. Therefore, MCFA forms the most stable clusters and so has the lowest reclustering rate. The reduction in reclustering rate reduces the control message overhead rate. Furthermore, in MCFA, each host independently selects its clusterhead based solely on the local information received from its neighbors, and so no message needs to be flooded within network. For mobility speeds faster than 50 km/h, it can be seen that the obtained results for MobHiD are very close to those of MCFA.

5 Conclusion

In this paper, we proposed a weighted learning automata-based clustering algorithm for wireless mobile ad hoc networks. In our proposed method, the relative mobility of each host with respect to all its neighbors is defined as its weight. In this algorithm, it is assumed that the mobility characteristics of the host and so the weight associated with the host are random variables with unknown distribution parameters. Therefore, the expected weight (relative mobility) of each host is estimated by sampling its mobility parameters in various epochs. In the proposed method, at each neighborhood, the host with the highest expected weight is selected as the cluster-head. This ensures the stability of the clusters against the host mobility. In this algorithm, each host chooses its cluster-head based solely on the local information received from its neighboring hosts. To show the performance of the proposed algorithm, we conducted several simulation experiments and compared the obtained results with the well-known existing clustering methods. The numerical results show that the proposed algorithm outperforms the others in terms of the number of clusters, cluster lifetime, reaffiliation rate, and control message overhead.

Acknowledgements 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. Akbari Torkestani, J., Meybodi, M.R.: 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), Kuala Lumpur, Malaysia, April 3–5 (2009)
2. Akbari Torkestani, J., Meybodi, M.R.: Solving the minimum spanning tree problem in stochastic graphs using learning automata. In: Proceedings of International Conference on Information Management and Engineering (ICIME 2009) Kuala Lumpur, Malaysia, April 3–5 (2009)
3. Akbari Torkestani, J., Meybodi, M.R.: Clustering the wireless ad hoc networks: a distributed learning automata approach. J. Parallel Distrib. Comput. **70**, 394–405 (2010)
4. Akbari Torkestani, J., Meybodi, M.R.: Weighted steiner connected dominating set and its application to multicast routing in wireless MANETs. Wireless Personal Communications (2010). doi:10.1007/s11277-010-9936-4
5. Akbari Torkestani, J., Meybodi, M.R.: A learning automata-based cognitive radio for clustered wireless ad-hoc networks. J. Netw. Syst. Manag. (2010, in press)
6. Akbari Torkestani, J., Meybodi, M.R.: An intelligent backbone formation algorithm in wireless ad hoc networks based on distributed learning automata. J. Comput. Netw. **54**, 826–843 (2010)
7. Akbari Torkestani, J., Meybodi, M.R.: A new vertex coloring algorithm based on variable action-set learning automata. J. Comput. Inf. **29**(3), 1001–1020 (2010)
8. Artail, H., Antoun, R., Fawaz, K.: CRUST: implementation of clustering and routing functions for mobile ad hoc networks using reactive tuple-spaces. Ad Hoc Netw. **7**, 1064–1081 (2009)
9. Baker, D.J., Ephremides, A.: The architectural organization of a mobile radio network via a distributed algorithm. IEEE Trans. Commun. **29**(11), 1694–1701 (1981)
10. Basagni, S.: Distributed clustering for ad-hoc networks. In: Proceedings of International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN'99), pp. 310–315 (1999)

11. Basu, P., Khan, N., Little, T.D.C.: Mobility-based metric for clustering in mobile ad-hoc networks. In: Workshop on Distributed Computing Systems, pp. 413–418 (2001)
12. Chatterjee, M., Das, S., Turgut, D.: WCA: a weighted clustering algorithm for mobile ad-hoc networks. Cluster Comput. **5**, 193–204 (2002)
13. Chen, A., Cai, Z., Hu, D.: Clustering in mobile ad hoc network based on neural network. J. Cent. South Univ. Technol. **13**(6), 699–702 (2006)
14. Elhdhili, M., Azzouz, L., Kamoun, F.: CASAN: clustering algorithm for security in ad hoc networks. Comput. Commun. **31**, 2972–2980 (2008)
15. Er, I.I., Seah, W.K.G.: Clustering overhead and convergence time analysis of the mobility-based multi-hop clustering algorithm for mobile ad hoc networks. J. Comput. Syst. Sci. **72**, 1144–1155 (2006)
16. Er, I., Seah, W.K.G.: Performance analysis of mobility-based D-hop (MobDHop) clustering algorithm for mobile ad-hoc networks. Comput. Netw. **50**, 3375–3399 (2006)
17. Esnaashari, M., Meybodi, M.R.: Data aggregation in sensor networks using learning automata. Wirel. Netw. **16**(3), 687–699 (2010)
18. Esnaashari, M., Meybodi, M.R.: Dynamic point coverage problem in wireless sensor networks: a cellular learning automata approach. J. Ad hoc Sens. Wirel. Netw. **10**(2–3), 193–234 (2010)
19. Esnaashari, M., Meybodi, M.R.: A learning automata based scheduling solution to the dynamic point coverage problem in wireless sensor networks. Comput. Netw. (2010, to appear)
20. Garcia Nocetti, F., Solano Gonzalez, J., Stojmenovic, I.: Connectivity based k-hop clustering in wireless networks. Telecommun. Syst. **22**(1–4), 205–220 (2003)
21. Gerla, M., Tsai, J.: Multicluster, mobile, multimedia radio network. ACM/Baltzer J. Wirel. Netw. **1**(3), 255–265 (1995)
22. Ghosh, R., Basagni, S.: Mitigating the impact of node mobility on ad-hoc clustering. J. Wirel. Commun. Mob. Comput. **8**, 295–308 (2008)
23. 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)
24. Konstantopoulos, C., Gavalas, D., Pantziou, G.: Clustering in mobile ad-hoc networks through neighborhood stability-based mobility prediction. Comput. Netw. **52**, 1797–1824 (2008)
25. Lin, C.R., Gerla, M.: Adaptive clustering for mobile wireless networks. IEEE J. Sel. Areas Commun. **15**(7), 1265–1275 (1997)
26. Narendra, K.S., Thathachar, K.S.: Learning Automata: An Introduction. Prentice-Hall, New York (1989)
27. Palit, R., Hossain, E., Thulasiraman, P.: MAPLE: a framework for mobility-aware pro-active low energy clustering in ad-hoc mobile wireless networks. Wirel. Commun. Mob. Comput. **6**, 773–789 (2006)
28. Shahzad, W., Aslam Khan, F., Basit Siddiqui, A.: Clustering in mobile ad hoc networks using comprehensive learning particle swarm optimization (CLPSO). Commun. Netw. **56**, 342–349 (2009)
29. Thathachar, M.A.L., Harita, B.R.: Learning automata with changing number of actions. IEEE Trans. Syst. Man Cybern. **SMG-17**, 1095–1100 (1987)
30. Thathachat, M.A.L., Sastry, P.S.: A hierarchical system of learning automata that can learn the globally optimal path. Inf. Sci. **42**, 743–766 (1997)
31. Wang, S.-C., Pan, H.-H., Yan, K.-Q., Lo, Y.-L.: A unified framework for cluster manager election and clustering mechanism in mobile ad hoc networks. Comput. Stand. Interfaces **30**, 329–338 (2008)
32. Yang Yu, J., Chong, P.: An efficient clustering scheme for large and dense mobile ad hoc networks (MANETs). Comput. Commun. **30**, 5–16 (2006)
33. Zhang, Y., Mee Ng, J., Ping Low, C.: A distributed group mobility adaptive clustering algorithm for mobile ad hoc networks. Comput. Commun. **32**, 189–202 (2009)
34. Akbari Torkestani, J., Meybodi, M.R.: Learning automata-based algorithms for finding minimum weakly connected dominating set in stochastic graphs. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. **18**(6) (2010)
35. Akbari Torkestani, J., Meybodi, M.R.: A learning automata-based heuristic algorithm for solving the minimum spanning tree problem in stochastic graphs. J. Supercomput. (2010, in press)
36. Akbari Torkestani, J., Meybodi, M.R.: A cellular learning automata-based algorithm for solving the vertex coloring problem. J. Expert Syst. Appl. (2011, in press)
37. Akbari Torkestani, J., Meybodi, M.R.: Mobility-based multicast routing algorithm in wireless mobile ad hoc networks: a learning automata approach. J. Comput. Commun. **33**(6), 721–735 (2010)
38. Akbari Torkestani, J., Meybodi, M.R.: An efficient cluster-based CDMA/TDMA scheme for wireless mobile ad-hoc networks: a learning automata approach. J. Netw. Comput. Appl. **33**, 477–490 (2010)

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.