Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Clustering the wireless Ad Hoc networks: A distributed learning automata approach

Javad Akbari Torkestani [a,*], Mohammad Reza Meybodi [b,c]

[a] Department of Computer Engineering, Islamic Azad University, Arak Branch, Arak, Iran

[b] Department of Computer Engineering and IT, Amirkabir University of Technology, Tehran, Iran

[c] Institute for Studies in Theoretical Physics and Mathematics (IPM), School of Computer Science, Tehran, Iran

ARTICLE INFO

Article history:
Received 14 March 2009
Received in revised form
25 July 2009
Accepted 9 October 2009
Available online xxxx

Keywords:
Wireless Ad Hoc networks
Clustering
Weakly connected dominating set
Distributed learning automata

ABSTRACT

In Ad Hoc networks, the performance is significantly degraded as the size of the network grows. The network clustering by which the nodes are hierarchically organized on the basis of the proximity relieves this performance degradation. Finding the weakly connected dominating set (WCDS) is a promising approach for clustering the wireless Ad Hoc networks. Finding the minimum WCDS in the unit disk graph is an NP-Hard problem, and a host of approximation algorithms has been proposed. In this article, we first proposed a centralized approximation algorithm called DLA-CC based on distributed learning automata (DLA) for finding a near optimal solution to the minimum WCDS problem. Then, we propose a DLA-based clustering algorithm called DLA-DC for clustering the wireless Ad Hoc networks. The proposed cluster formation algorithm is a distributed implementation of DLA-CC, in which the dominator nodes and their closed neighbors assume the role of the cluster-heads and cluster members, respectively. In this article, we compute the worst case running time and message complexity of the clustering algorithm for finding a near optimal cluster-head set. We argue that by a proper choice of the learning rate of the clustering algorithm, a trade-off between the running time and message complexity of algorithm with the cluster-head set size (clustering optimality) can be made. The simulation results show the superiority of the proposed algorithms over the existing methods.

1. Introduction

A wireless Ad Hoc network is a multi-hop wireless communication network supporting a collection of mobile hosts. There is no fixed infrastructure and no central administration and the mobile hosts can form a temporary network infrastructure in an Ad Hoc fashion. Two hosts can directly communicate when they are within transmission range of each other, and indirectly through relaying by the intermediate hosts. In an Ad Hoc network, each host assumes the role of a router and relays the packets toward the final destinations, if a source cannot directly send the packets to a final destination due to the limitation of the radio transmission range. Since the wireless Ad Hoc networks exhibit severe resource constraints such as the bandwidth and power limitations, network topology changes, and the lack of the fixed infrastructures and consequently, centralized administrations, to achieve good performance in Ad Hoc networks, the load on the hosts should be kept as low as possible [22,18].

The network performance is significantly degraded as the network becomes larger, and the theoretical analyses [20] show that even under the optimal circumstances, the throughput of each host rapidly declines towards zero as the network size increases. Among the solutions proposed for solving the scalability problem in Ad Hoc networks, the cluster formation approach has attracted a lot of attention. The main idea behind the clustering approach is to group together the network hosts that are in physical proximity, to achieve scalability and efficiency. The clusters provide a hierarchical structure to abstract the large scale networks which can be simply and locally organized [31,13]. A clustering algorithm is a method of dividing the network into clusters so that every cluster includes a cluster-head and the hosts that can directly communicate with the cluster-head. Unpredictable topology changes due to the mobility of hosts, and resource limitations (e.g., bandwidth and power limitations) are important features of the wireless Ad Hoc networks. Due to the limitations of the Ad Hoc networks, having a small number of the cluster-heads and also minimizing the modifications of the cluster-heads are desired. The most basic clustering methods that have been studied in the context of Ad Hoc networks are based on the dominating sets. Finding the minimum weakly connected dominating set (WCDS) of the network graph is one of the most

* Corresponding author.
E-mail addresses: iau_akbari@yahoo.com, j-akbari@iau-arak.ac.ir (J. Akbari Torkestani), mmeybodi@aut.ac.ir (M.R. Meybodi).

0743-7315/$ – see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2009.10.002

investigated methods for cluster formation in which a dominator node assumes the role of a cluster-head and its one-hop neighbors are assumed to be cluster members. The structure of the network graph can be simplified using WCDS and made more succinct for routing in Ad Hoc networks [16,15].

Clustering the Ad Hoc networks based on the WCDS was first proposed by Chen and Listman [13,10]. The distributed approximation clustering algorithm proposed by Chen and Listman is also inspired by Guha and Khuller's [19] centralized approximation algorithm for finding small connected dominating sets (CDS). Guha and Khuller [19] proposed two centralized greedy heuristic algorithms with bounded performance that guarantee for CDS formation. In the first algorithm, the CDS is grown from one node outward, and in the second algorithm, a WCDS is constructed, and then the intermediate nodes are selected to create a CDS. Chen and Listman [11,12] also proposed a zonal algorithm, in which the graph is divided into regions, a WCDS is constructed for each region, and adjustments are made along the borders of the regions to produce a WCDS for the whole graph. Their algorithm for the partitioning phase is partly based on a Minimum Spanning Tree (MST) algorithm of Gallager et al. [17]. Han and Jia [22] also proposed an area-based distributed algorithm for WCDS construction in Ad Hoc networks with constant approximation ratio, linear time and message complexity. While it has a lower message complexity than the zonal algorithm proposed by Chen and Listman, it outperforms the mentioned algorithm. Basagni and Mastrogiovann [6] presented a performance comparison of the protocols proposed for clustering and backbone formation in large scale Ad Hoc network. Alzoubi et al. [5] presented two distributed algorithms for finding a WCDS in Ad Hoc networks. The first algorithm was implemented by first electing a leader among the nodes, which was going to be the root of a spanning tree. The spanning tree is then traversed and the dominator nodes are selected. But the distributed leader election is extremely expensive in practice, and exhibits a very low degree of parallelism. The second algorithm first constructs a maximum independent set (MIS) by an iterative labeling strategy, and then modifies the MIS by selecting one intermediate node between each pair of dominators separated by exactly three hops.

In this article, we first propose a distributed learning automata (DLA)-based centralized approximation algorithm called DLA-CC for solving the minimum WCDS problem in a unit disk graph. Finding the WCDS is a well-known approach for cluster formation in wireless Ad Hoc networks. In Ad Hoc networks, there is neither a fixed infrastructure nor a central administration, and so the centralized algorithms (like DLA-CC) are not applicable in such environments. Therefore, in the second part of this article, we propose a distributed version of DLA-CC called DLA-DC for clustering the wireless Ad Hoc networks. The proposed clustering algorithm aims at finding a near optimal solution to the minimum WCDS in a distributed fashion. This domination set is called the cluster-head set. In this method, the dominator nodes play the role of the cluster-heads and their one-hop neighbors assume the role of the cluster members. Each of the proposed algorithms consists of a number of stages, and at each stage, a WCDS is constructed by randomly activating a number of automata in a DLA. As the proposed algorithms approach to the end, the learning automata tend to an action selection policy that determines the minimum size cluster-head set (or WCDS) with the highest probability. In this article, we estimate an upper bound on the running time (the number of iterations) and the message complexity of the proposed clustering algorithm for finding a $1/(1-\varepsilon)$ optimal cluster-head set. We also show that by a proper choice of the learning rate of the clustering algorithm, a trade-off between the running time and message complexity of algorithm with the cluster-head set size (clustering optimality) is made. To show the efficiency of DLA-CC, it is compared with Guha I [19], Guha II [19], and WCDS-CTR [13]. The simulation experiments show the superiority of DLA-CC over the others in terms of the cardinality of the WCDS (i.e., the number of dominators). We also compare the proposed clustering algorithm (DLA-DC) with Min ID [22], Max Degree [22], WCDS-DST [13], and AWF [5] in terms of the cluster-head set size and the message overhead of algorithm. The obtained results show that DLA-DC outperforms the best existing WCDS-based clustering algorithms.

The rest of the article is organized as follows. In Section 2, the WCDS problem formulation, learning automata and some preliminaries are presented. In Section 3, a centralized DLA-based approximation algorithm is proposed to solve the minimum WCDS problem in a unit disk graph. In Section 4, a DLA-based distributed algorithm is proposed for clustering the wireless Ad Hoc networks. The worst case running time and message complexity of the proposed clustering algorithm is computed in Section 5. In Section 6, the performance of the proposed algorithms is evaluated through the simulation experiments, and Section 7 concludes the article.

2. Preliminaries

In this section, some preliminaries on dominating sets and WCDS problem, learning automata and some of its variations are presented.

2.1. Dominating sets

A wireless Ad Hoc network can be modeled as a unit disk graph $G = (V, E)$, where the hosts represent the individual hosts and an edge connects two hosts if the corresponding hosts are within transmission range of each other. The closed neighborhood $N[v]$ consists of the hosts adjacent to v and host v itself. The closed neighborhood $N[S]$ of the set S is the union $\bigcup_{v\in S} N[v]$. A dominating set (DS) of a graph $G = (V, E)$ is a host subset $S \subseteq V$, such that every host $v \subseteq V$ is either in S or adjacent to a host of S. A host of S is said to dominate itself and all adjacent hosts. A minimum DS (MDS) is a DS with the minimum cardinality. A dominating set is also an independent dominating set, if no two hosts in the set are adjacent. A CDS S of a given graph G is a dominating set whose induced sub graph, denoted $\langle S\rangle$, is connected, and a minimum CDS (MCDS) is a CDS with the minimum cardinality. A MCDS forms a virtual backbone in the graph by which the routing overhead can be significantly reduced, where the number of hosts responsible for routing can be reduced to the number of hosts in the backbone. The MDS and MCDS problems are known as NP-Hard problems [14,24], and even for a unit disk graph, the problem of finding a MCDS is also NP-Hard [24].

A dominating set S is a WCDS of a graph G, if the graph $\langle S\rangle_W = (N[S], E \cap (N[S] \times S))$ is a connected sub graph of G. In other words, the weakly induced sub graph $\langle S\rangle_W$ contains the hosts of S, their neighbors, and all edges with at least one endpoint in S. A sample UDG and one of its WCDS are shown in Fig. 1A and 1B, respectively. The dominator nodes assume the role of the cluster-heads and they have been colored black.

It is assumed that the Ad Hoc network comprises a group of wireless hosts communicating through a common broadcast channel using omnidirectional antennas and all hosts have the same transmission range. That is, the corresponding topology graph is a unit disk graph. Scheduling of transmissions is the responsibility of the MAC layer, and like many existing approaches, we are not concerned with the issues of using a shared wireless channel to send the messages avoiding the collisions and contentions. Each host has a unique ID number (e.g., IP address) and also needs to know the ID number of all other hosts.

Each node of a WCDS is said to be a dominator node and its corresponding host in an Ad Hoc network a cluster-head. Two

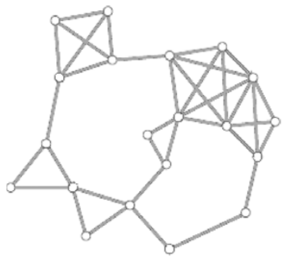

Fig. 1A. A sample unit disk graph.

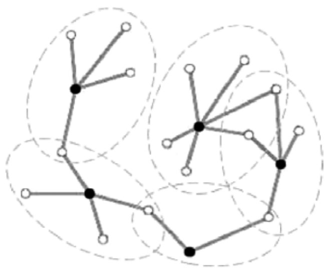

Fig. 1B. Weakly connected dominating set.

hosts u, v are connected by a link (u, v) and are said to be neighbors, if there exists a direct bidirectional communication channel connecting u and v, and so the network graph is assumed to be undirected.

2.2. Learning automata, distributed learning automata and variable action-set learning automata

2.2.1. Learning automata

A learning automaton [25,35,33,34,23,26,7] 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 in 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 to 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 [25].

The environment can be described by a triple $E \equiv \{\alpha, \beta, c\}$, where $\alpha \equiv \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ represents the finite set of the inputs, $\beta \equiv \{\beta_1, \beta_2, \dots, \beta_m\}$ denotes the set of the values can be taken by the reinforcement signal, and $c \equiv \{c_1, c_2, \dots, c_r\}$ 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 environment that allows a finite number of 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 [25,35,33,34,23,26]: 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 $\underline{p}$ is updated. Let $\alpha_i(k)$ be the action chosen by the automaton at instant k.

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

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

$$p_j(n+1) = \begin{cases} (1-b)p_j(n) & j = i \\ \left(\dfrac{b}{r-1}\right) + (1-b)p_j(n) & \forall j\; j \neq i. \end{cases} \tag{2}$$

When the action taken is penalized by the environment (i.e., $\beta(n) = 1$). r is the number of actions that can be chosen by the automaton, $a(k)$ and $b(k)$ denote the reward and penalty parameters and determine the amount of increases and decreases of the action probabilities, respectively. If $a(k) = b(k)$, the recurrence equations (1) and (2) are called linear reward–penalty (L_{R-P}) algorithm, if $a(k) \gg b(k)$ the given equations are called linear reward–ε penalty ($L_{R-\varepsilon P}$), and finally if $b(k) = 0$ they are called linear reward–In action (L_{R-I}). In the latter case, the action probability vectors remain unchanged when the action taken is penalized by the environment.

Learning automata is proved to perform well in the dynamic environments of wireless, Ad Hoc and sensor networks. Haleem and Chandramouli [21] 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. [28] 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 [29] 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. A decentralized approach of the above method is also given [27,30]. Ravana and Morthy [32] 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 is also used in cellular radio networks to dynamically adjusting the number of guard channels [8,9].

2.2.2. Distributed learning automata

A DLA [7] is a network of the learning automata, which collectively cooperate to solve a particular problem. Formally, a DLA can be defined by a quadruple $\langle A, E, T, A_0 \rangle$, where $A = \{A_1, \dots, A_n\}$ 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_{ij} 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 the 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. In [1], Akbari Torkestani and Meybodi proposed distributed learning automata for backbone formation in wireless Ad Hoc networks.

2.2.3. Variable action-set learning automata

A variable action-set learning automaton is an automaton in which the number of actions available at each instant changes with time. It has been shown in [33] 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 automaton has a finite set of n actions, $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$. $A = \{A_1, A_2, \dots, A_m\}$ 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 $\psi(k) = \{\psi_1(k), \psi_2(k), \dots, \psi_m(k)\}$ defined over the possible subsets of the actions, where $\psi_i(k) = prob[A(k) = A_i \mid A_i \in A, 1 \leq i \leq 2^n - 1]$. $\hat{p}_i(k) = prob[\alpha(k) = \alpha_i \mid A(k), \alpha_i \in A(k)]$ is the probability of choosing action α_i, conditioned on the event that the action subset $A(k)$ has already been selected and also $\alpha_i \in A(k)$. The scaled probability $\hat{p}_i(k)$ is defined as

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

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 [33]. 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 [2], connected dominating set problem [3], and stochastic minimum spanning tree problem [4].

3. DLA-based WCDS formation algorithm (DLA-CC)

In this section, a DLA-based centralized approximation algorithm called DLA-CC is proposed for finding a near optimal solution to the minimum WCDS problem described in Section 2.1. In this algorithm, a network of learning automata isomorphic to the input unit disk graph $G(V, E)$ is initially formed by assigning to each vertex (e.g., v_i) of the graph a learning automaton (e.g., A_i). The resulting network of the learning automata can be described by a duple $\langle \underline{A}, \underline{\alpha} \rangle$, where $\underline{A} = \{A_1, A_2, \dots, A_n\}$ denotes the set of learning automata corresponding to the vertex-set, n is the cardinality of the vertex-set V, and $\underline{\alpha} = \{\underline{\alpha_1}, \underline{\alpha_2}, \dots, \underline{\alpha_n}\}$ denotes the set of action-sets. $\underline{\alpha_i} = \{\alpha_{i,j} \mid \overline{(v_i, v_j)} \notin E; \forall \overline{v_j} \in V\}$ denotes the set of actions that can be taken by the learning automata. In other words, the action-set of automaton A_i includes all the vertices of the graph which are not adjacent with vertex v_i. Action $\alpha_{i,j}$ means that vertex v_i chooses vertex v_j as a dominator. Due to such an action-set formation method, some learning automata will have the same actions. The drawback of these common actions is to choose the redundant dominators (i.e., the dominators by which no more dominatee nodes can be spanned) and the same dominators by different automata. To solve these problems, the proposed algorithm deals with the learning automaton with changing number of actions [33]. In this algorithm, each activated learning automaton is allowed to prune its action-set by disabling the actions corresponding to the selected dominators and their neighbors. This reduction in number of actions increases the convergence speed, and consequently, decreases the running time of the proposed algorithm.

In this algorithm, each vertex (or learning automaton) can be in one of two states *active* and *passive*, and is initially set to the passive state. The proposed algorithm consists of a number of stages, and at each stage, a WCDS is constructed by randomly activating a number of automata in a DLA (Step 1). The action probability vector of the activated learning automata is updated on the basis of the optimality of the selected WCDS (Steps 2 and 3). The iterative process of constructing the WCDSs continues until a near optimal solution to the minimum WCDS problem is found (Step 4). The proposed algorithm has been given in more detail in Fig. 2.

As shown in Fig. 2, graph G, and thresholds K and P are the input parameters, and the smallest WCDS of the input graph is the output of DLA-CC. The kth iteration of DLA-CC can be described as follows. The first dominator is selected by automaton A_0. The selected dominator (vertex v_i) and its neighbors are added to the dominatee set (i.e., d_k). The automaton corresponding to the selected dominator (i.e., automaton A_i) is activated and updates its action-set by disabling the actions corresponding to the dominated vertices. Then, this automaton randomly chooses the next dominator. The process of dominator selection (Lines 14–18) continues until the number of dominated vertices is equal to the network size (i.e., $|d_k| = n$). Then, the cardinality of the selected WCDS is compared with dynamic threshold T_k. Threshold T_k is the cardinality of the smallest WCDS found until stage k. The selected WCDS (i.e., δ_k) is penalized if its cardinality is greater than the dynamic threshold T_k and rewarded otherwise. At each stage, the dynamic threshold is set to the smallest WCDS. As shown in Line 26, at the end of each stage the disabled actions must be enabled as described in Section 2.2.3 on variable action-set learning automata. Here, the kth iteration of DLA-CC is over.

As the algorithms proceeds, the automata learn how to choose the dominators so that all the vertices are dominated with the minimum number of dominators. The proposed algorithm terminates, if the probability of choosing the WCDS is greater than P or the number of stages exceeds pre-specified threshold K. It should be noted that the probability of choosing a WCDS is defined as the product of the probabilities with which the dominators of WCDS are selected. The WCDS which is formed before the algorithm stops is the WCDS with the minimum cardinality among all WCDSs of the graph.

```
ALGORITHM DLA-CC
1:Input: Unit Disk Graph $G(V,E)$, Threshold $K$, $P$
2:Output: The minimum size WCDS
3:Assumptions:
4:  Assign learning automaton $A_i$ to vertex $v_i$ and initially set it in a passive state
5:  Let $\underline{\alpha_i}$ denotes the action-set of automaton $A_i$
6:  Begin Algorithm
7:      Let dynamic threshold $T_k$ denotes the cardinality of the smallest WCDS constructed until stage $k$, and initially set to $n$
8:      Let $k$ denotes the stage number and is initially set to 0
9:      Let $A_0$ be the initial learning automaton whose action-set is vertex-set $V$
10:     Repeat
11:         Let $\delta_k$ be the set of dominators selected at stage $k$ and initially set to null
12:         Let $d_k$ denotes the set of dominatees, selected at stage $k$ and is initially set to null
13:         Automaton $A_0$ randomly chooses one of its actions, activates it, denotes it $v_i$ and adds it to $\delta_k$
14:         While $|d_k| < n$ do        {$|x|$ denotes the cardinality of set $x$}
15:             Add the neighbors of dominator $v_i$ to $d_k$
16:             Update the action probability vector of $A_i$ by disabling the actions corresponding to the vertices in $d_k$
17:             Automaton $A_i$ randomly chooses one of its actions, activates it, denotes it $v_i$ and adds it to $\delta_k$
18:         End while
19:         If $|\delta_k| \le T_k$ Then
20:             Reward the actions chosen by the activated learning automata
21:             $T_k \leftarrow |\delta_k|$
22:         Else
23:             Penalize the actions chosen by the activated learning automata
24:         End if
25:         Enable all the actions disabled during the current iteration
26:         Increment stage number $k$
27:     Until (The choice probability of WCDS is greater than $P$ or the stage number $k$ is greater than threshold $K$)
28:End Algorithm
```

Fig. 2. Algorithm DLA-CC.

4. Proposed clustering algorithm (DLA-DC)

Since in wireless Ad Hoc networks there is neither a fixed infrastructure nor a central administration, the centralized algorithms are not feasible in such environments. Moreover, to gather all the required information in a certain host, for executing the algorithm, consumes a large number of messages and considerably more energy which is a very scarce resource in wireless Ad Hoc networks [22]. Therefore, in this section, a distributed approximation algorithm based on DLA, called DLA-DC, is proposed for clustering the wireless Ad Hoc networks by finding a near optimal solution to the WCDS problem. In fact, the proposed clustering algorithm is a generalization of DLA-CC, in which the minimum size cluster-head set (or minimum WCDS of the UDG induced by the network topology) is determined in a fully distributed fashion. In this clustering method, the dominator nodes assume the role of the cluster-heads and their one-hop neighbors (dominatee nodes) the role of the cluster members. At each iteration of the clustering algorithm, the network graph is clustered by randomly choosing the dominator nodes as the cluster-heads. The learning automata, in an iterative greedy strategy, find a policy that determines the minimum size cluster-head set of the network graph.

In this algorithm, like DLA-CC, a network of the learning automata, isomorphic to the UDG induced by the network topology, is first formed by assigning a learning automaton (e.g., A_i) to each host (e.g., H_i) of the network. Each host has a unique ID number and knows its neighbors' ID. In this algorithm, to form the action-set of learning automaton A_i, its corresponding host (i.e., host H_i) sends a message locally to its one-hop neighbors. The hosts which are within the transmission range of the sender host, upon receiving the message, reply it. The sender forms its action-set on the basis of the received replies. Each host by which the message is replied is associated with an action. Let $\underline{\alpha_i} = \{\alpha_{i,j} \mid H_j$ is a neighbor of $H_i\}$ denotes the action-set of learning automaton A_i. Action $\alpha_{i,j}$ corresponds to the selection of host H_j as a cluster-head by host H_i. Each host requires the following data structures to participate in the cluster formation process:

max_iteration, a stopping condition for the algorithm as a maximum number of iterations.
pchs, a threshold required for terminating the cluster formation process as the probability of choosing the cluster-head set.
cluster_head_set, a set of the chosen cluster-heads at each iteration.
cluster_list, a set of hosts in which each member is a one-hop neighbor of at least one host in the *cluster_head_set*.
prob_vector, a vector of the probability of choosing the members of *cluster_head_set*.
min_size, a dynamic threshold contains the cardinality of the smallest *cluster_head_set*, which has been selected yet.
iteration_num, a counter which keeps the number of constructed *cluster_head_set*.

The proposed clustering algorithm consists of two phases. Cluster formation (initial clustering) and cluster maintenance (or re-clustering). The cluster formation phase is performed when the network starts operating, and the cluster maintenance when a cluster-head decides to leave the network or a host cannot communicate with its cluster-head. The cluster maintenance procedure will be described in Section 4.1. At the end of the initial clustering phase, a fully clustered network is created. Since the proposed DLA-based clustering algorithm is fully distributed, cluster formation and cluster maintenance procedures can be initiated by each of the hosts. During the cluster formation or cluster maintenance procedures, each host may receive from or send to the other hosts the following messages: *activation*, *rewarding*, *penalizing* and *clustering* message. When a host (e.g., host H_i) decides to initiate each of the above mentioned procedures, it activates its corresponding automaton (i.e., automaton A_i) and chooses one of its actions. The probability with which learning automaton A_i chooses this action is added to the *prob_vector*. Host H_i then sends an *activation* message to the host (new cluster-head) corresponding to the chosen action.

An *activation* message includes *cluster_list*, *cluster_head_set*, *min_size*, *prob_vector*, and *iteration_num*. The state of a given host changes to the active state when it receives an *activation* message. When a given host H_i receives an *activation* message, it inserts its ID number as a new cluster-head into the *cluster_head_set*, if at least one of its one-hop neighbors is not in the *cluster_list*. This prevents the redundant cluster-heads to be chosen. To update the *cluster_list* it adds its one-hop neighbors' ID to this list. The action-set of learning automaton A_i is updated by disabling the actions associated with the cluster-heads selected so far. As described earlier, this avoids choosing the same cluster-heads by different automata, and so improves the convergence speed of algorithm. In this case, if there exist more actions that can be taken by learning automaton A_i and the *cluster_list* does not include all the hosts, currently active automaton A_i chooses one of its actions as a new cluster-head, updates *prob_vector* by adding the choice probability of this action, and sends an *activation* message to the chosen cluster-head. Otherwise, if the size of the *cluster_list* equals to the network size and the size of the *cluster_head_set* is less than or equal to dynamic threshold *min_size*, the dynamic threshold is set to the cardinality of the selected *cluster_head_set* and all the chosen actions of the activated automata are rewarded by sending back a *rewarding* message, otherwise they (i.e., the chosen actions of the activated automata) are penalized by sending back a *penalizing* message.

At the end of each iteration, after rewarding or penalizing the activated automata, the stopping condition of the cluster formation (or cluster maintenance) process must be verified. The stopping condition is met if the choice probability of the *cluster_head_set* is less than the certain threshold *pchs* or *iteration_num* exceeds the per-specified threshold *max_iteration*. The choice probability of the *cluster_head_set* is calculated as the product of the choice probabilities of the selected cluster-heads based on the information contained in *prob_vector*. If the stopping condition is true, currently active automaton A_i initiates a new iteration, randomly chooses a new cluster-head, and sends an *activation* message to it. Otherwise, it generates a *clustering* message including the last selected *cluster_head_set* and broadcasts it within the network.

A *clustering* message includes the *cluster_head_set* selected during the last iteration. When host H_i receives a *clustering* message, it assumes the role of a cluster-head if it finds its ID number in the *cluster_head_set*. It assumes the role of a cluster-member otherwise.

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

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

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

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

After rewarding the chosen action, the scaled action probability vector must be updated once again (or rescaled) by enabling all the disabled actions according to the rescaling method described in Section 2.2.3 on the variable action-set learning automata.

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 remain unchanged when they receive a *penalizing* message. In this case, the disabled actions of each activated learning automaton are enabled again.

4.1. Cluster maintenance

In wireless Ad Hoc networks, the major resources of the network topology dynamics are the node mobility and node failure. In such networks, a host can move freely and randomly anywhere, and so it may leave its cluster and join the other at any time. For this reason, in Ad Hoc networks, the cluster membership is highly dynamic and hard to predict due to the frequent network topology changes. Therefore, the cluster maintenance (re-clustering) is required to recover the failed clusters.

In our proposed clustering algorithm, when a host joins the network, it initially sends a *JREQ* (i.e., join request) message to its neighboring hosts and then waits for a certain period of time. Each cluster-head replies the received *JREQ* message by sending back a *JREP* (i.e., join reply) message. In this method, the following cases might occur for a newly joining host.

- If the newly joining host receives a *JREP* message, it chooses the sender of the *JREP* message as its cluster-head.
- If it receives more than one *JREP* message, it chooses the sender host with the highest ID number as its cluster-head.
- If the newly joining host receives no *JREP* messages (cannot connect to any cluster-heads), it initiates a re-clustering process.

When a cluster-head decides to leave the network, it initiates a re-clustering process. It is clear that no re-clustering process is required when a cluster member decides to leave the network. The re-clustering process is similar to the initial clustering.

One of the most important advantages of the proposed clustering scheme is that during the re-clustering phase, the proposed algorithm (due to using the learning automata) quickly converges to the new optimal cluster-head set. That is, the overhead of the re-clustering phase is significantly smaller as compared with the initial clustering phase. This is due to the fact that during the initial clustering, the choice probability of each cluster-head set candidate grows proportional to its optimality among the other candidates. Therefore, in the absence of the optimal cluster-head set, the second optimal cluster-head set has the highest probability. Comparing with the initial clustering phase, re-clustering phase requires a significantly smaller number of iterations for finding the new optimal cluster-head set. In Ad Hoc networks where the node mobility and node failure frequently change the network topology, the network re-clustering process is expected to be more frequently used. Therefore, the proposed clustering algorithm significantly reduces the cluster maintenance overhead which is an important result in Ad Hoc networks.

5. Complexity analysis of the clustering algorithm

In this section, to analyze the costs of the proposed clustering method, we compute the worst case running time (Theorem 1) and message complexity (Theorem 2) of algorithm DLA-DC to find a $1/(1-\varepsilon)$ optimal cluster-head set for clustering the network graph.

Theorem 1. *Let* OPT *denotes the size of the smallest cluster-head set for clustering network graph G, and* $\underline{q}(k) = \{q_1(k), \ldots, q_r(k)\}$ *is updated according to the proposed clustering algorithm (DLA-DC). The time required for finding a* $\frac{1}{1-\varepsilon} \cdot |\text{OPT}|$ *(for* $0 < \varepsilon < 1$*) size cluster-head set (e.g.,* ω_i*) in DLA-DC is not longer than*

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

where $q_i(k)$ *denotes the probability of choosing cluster-head set* ω_i *at stage k,* $\varepsilon \in (0, 1)$ *is the error rate of algorithm denoted as* $1 - pchs$ *in DLA-DC, a denotes the learning rate of algorithm, n is the number of hosts in the network graph, and r is the number of cluster-head sets.*

Proof. Let q_i be the initial probability of choosing cluster-head set ω_i, and p_j be the initial probability of choosing host H_j as a cluster-head. The worst case occurs when all the other cluster-head sets to be chosen before the smallest cluster-head set ω_i. In this case, the learning process can be divided into two distinct phases. In the first phase, called *shrinking phase,* it is assumed that all the other cluster-head sets to be chosen, from the largest to the smallest, and so rewarded before ω_i. Such an ordered (cluster-head set) selection procedure decreases the probability of choosing ω_i no more than that given in inequality (6). The second phase called *growing phase* is started when the cluster-head set ω_i is chosen for the first time. According to the proposed algorithm, during the growing phase, the probability of penalizing ω_i, and rewarding the other cluster-head sets is zero. Furthermore, since the reinforcement scheme by which the proposed algorithm updates the probability vectors is L_{R-I}, the conditional expectation of $q_i(k)$ (i.e., the probability of choosing cluster-head set ω_i at stage k), remains unchanged when the other cluster-head sets are penalized, and it increases only when ω_i is rewarded. In other words, during the growing phase, the changes in the conditional expectation of $q_i(k)$ is always non-negative and given in Eq. (7). As described in the proposed algorithm, the growing phase is continued until the probability of choosing cluster-head set ω_i is greater than or equal to $1-\varepsilon$. Let q_i^s denotes the probability of choosing cluster-head set ω_i at the beginning of the shrinking phase (i.e., the initial value of q_i), and a denotes the learning rate of the proposed algorithm. Therefore, during the shrinking phase, the probability of choosing cluster-head set ω_i changes as

$$q_i^s(r) \geq q_i^s(r-1) \cdot (1-a)^m \tag{6}$$

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

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

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

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

where $q_i^s(r-1)$ denotes the probability of choosing cluster-head set ω_i at the end of the shrinking phase. For the sake of simplicity in notation, $q_i^s(r-1)$ is substituted by q_i^g, where $q_i^g = \prod_{H_j\in\omega_i} p_j(1-a)^{(r-1)}$ is the probability of choosing ω_i at the beginning of the growing phase, and $p_j(1-a)^{(r-1)}$ is substituted by ρ_j. As mentioned earlier, in the growing period, q_i^g increases, if ω_i is rewarded, and remains unchanged otherwise. Thus, during this phase, the probability of choosing cluster-head set ω_i increases as

$$\begin{aligned}
q_i^g(1) &= \prod_{H_j\in\omega_i} \rho_j + a \cdot (1-\rho_j) \\
q_i^g(2) &= \prod_{H_j\in\omega_i} \rho_j(1) + a \cdot (1-\rho_j(1)) = \prod_{H_j\in\omega_i} \rho_j(1) \cdot (1-a) + a \\
&\vdots \\
q_i^g(k-1) &= \prod_{H_j\in\omega_i} \rho_j(k-2) + a \cdot (1-\rho_j(k-2)) \\
&= \prod_{H_j\in\omega_i} \rho_j(k-2) \cdot (1-a) + a \\
q_i^g(k) &= \prod_{H_j\in\omega_i} \rho_j(k-1) + a \cdot (1-\rho_j(k-1)) \\
&= \prod_{H_j\in\omega_i} \rho_j(k-1) \cdot (1-a) + a
\end{aligned} \tag{7}$$

where ρ_j denotes the probability of choosing host H_j as a cluster-head at the beginning of the growing phase, and $q_i^g(k)$ denotes the probability with which cluster-head set ω_i is chosen at the end of the growing phase, which according to the theorem, it is assumed to $1-\varepsilon$. After some algebraic simplification, we have

$$\begin{aligned}
q_i^g(k) &= \prod_{H_j\in\omega_i} \rho_j(k-1) \cdot (1-a) + a \\
&= \prod_{H_j\in\omega_i} \left[\rho_j(k-2) \cdot (1-a) + a\right](1-a) + a \\
&= \prod_{H_j\in\omega_i} \rho_j(k-2) \cdot (1-a)^2 + a \cdot (1-a) + a \\
&= \prod_{H_j\in\omega_i} \left[\rho_j(k-3) \cdot (1-a) + a\right](1-a)^2 + a \cdot (1-a) + a \\
&= \prod_{H_j\in\omega_i} \rho_j(k-3) \cdot (1-a)^3 + a \cdot (1-a)^2 + a \cdot (1-a) + a \\
&\vdots \\
&= \prod_{H_j\in\omega_i} \rho_j(1) \cdot (1-a)^{k-1} + a \cdot (1-a)^{k-2} \\
&\quad + \cdots + a \cdot (1-a) + a \\
&= \prod_{H_j\in\omega_i} \rho_j \cdot (1-a)^k + a \cdot (1-a)^{k-1} + \cdots \\
&\quad + a \cdot (1-a) + a.
\end{aligned}$$

Hence, we have

$$\begin{aligned}
q_i^g(k) &= \prod_{H_j\in\omega_i} \rho_j \cdot (1-a)^k + a \cdot (1-a)^{k-1} + \cdots \\
&\quad + a \cdot (1-a) + a.
\end{aligned} \tag{8}$$

Substituting ρ_j by $p_j(1-a)^{(r-1)}$, we have

$$\begin{aligned}
q_i^g(k) &\geqslant \prod_{H_j\in\omega_i} p_j \cdot (1-a)^{k+r-1} + a \cdot (1-a)^{k-1} \\
&\quad + \cdots + a \cdot (1-a) + a
\end{aligned} \tag{9}$$

where k denotes the number of times cluster-head set ω_i must be selected until

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

From inequality (9), it follows that, the running time (the number of iteration) of the proposed algorithm is computed, if we find a value of k (for a given learning rate a) under which the condition given in Eq. (10) to be satisfied. From Eq. (8) we have

$$q_i^g(k) = \prod_{H_j\in\omega_i} \rho_j \cdot (1-a)^k + a\cdot(1+(1-a) + (1-a)^2 + \cdots + (1-a)^{k-1}) \tag{11}$$

and so

$$q_i^g(k) = \prod_{H_j\in\omega_i} \rho_j \cdot (1-a)^k + \sum_{i=0}^{k-1} a\cdot (1-a)^i\,. \tag{12}$$

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

$$q_i^g(k) = \prod_{H_j\in\omega_i} \rho_j \cdot (1-a)^k + a\cdot \frac{1-(1-a)^k}{1-(1-a)}. \tag{13}$$

From Eqs. (10) and (13) we have

$$\prod_{H_j\in\omega_i} \rho_j \cdot (1-a)^k + a\cdot \frac{1-(1-a)^k}{1-(1-a)} = 1-\varepsilon.$$

Since $q_i^g = \prod_{H_j\in\omega_i}\rho_j$ and $\rho_j = p_j(1-a)^{(r-1)}$ (for all $H_j \in \omega_i$), after some algebraic simplification, we have

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

Hence, we have

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

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

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

and thus the number of times ω_i is rewarded in the growing phase, apart from penalizing the other cluster-head sets, is obtained as

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

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

$$\frac{1+\varepsilon-q_i^g}{1-\varepsilon+q_i^g}\cdot k.$$

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

$$\mathrm{K} = \frac{2}{1-\varepsilon+q_i^g}\cdot k.$$

By substituting k from Eq. (15) and we have

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

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

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

which completes the proof of this theorem. ∎

Theorem 2. *The message complexity of the proposed clustering algorithm for finding a* $\frac{1}{1-\varepsilon}\cdot|\mathrm{OPT}|$ *(for* $0<\varepsilon<1$*) size cluster-head set is at most*

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

where OPT *denotes the size of the minimum cluster-head set (optimal solution to the WCDS problem),* $q_i(k)$ *denotes the probability of choosing cluster-head set* ω_i *at stage k,* $\varepsilon \in (0,1)$ *is the error rate of algorithm, a denotes the learning rate of algorithm, n is the number of hosts in the network graph, and r is the number of cluster-head sets.*

Proof. As proved in Theorem 1, the running time (number of iterations) of the proposed clustering algorithm to find a $\frac{1}{1-\varepsilon}\cdot|\mathrm{OPT}|$ size cluster-head set for the network graph is at most

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

Furthermore, as described in Section 5, at each iteration of the clustering algorithm, the number of messages that needs to be sent to form a cluster-head set is equal to the number of hosts in the selected cluster-head set. Hence, the message complexity of the clustering algorithm is smaller than

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

where m is the average size of the cluster-head set, and hence the proof of the theorem is completed. ∎

The immediate conclusion of Theorems 1 and 2 is that a trade-off between the running time and message complexity of the proposed clustering algorithm with the cluster-head set size (clustering optimality) can be made by a proper choice of the learning rate of algorithm. This is a worthwhile superiority of the proposed algorithm over the other clustering methods. That is, choosing a proper learning rate for the proposed clustering algorithm results in finding a near optimal cluster-head set in a reasonable running time and message complexity, regarding the constraints of the Ad Hoc networks.

6. Experimental results

In this section, we have conducted several simulation experiments in two groups to study the performance of the proposed algorithms. In the first group of our experiments (Experiment I), we investigate the effectiveness of the DLA-CC in terms of the dominating set size (i.e., size of the WCDS) as compared with centralized algorithms. In the second group (Experiment II, III), we measure the performance of the proposed clustering algorithm (DLA-DC) in terms of the cluster-head set size and message overhead, and compare the obtained results with those of the best WCDS-based clustering algorithms.

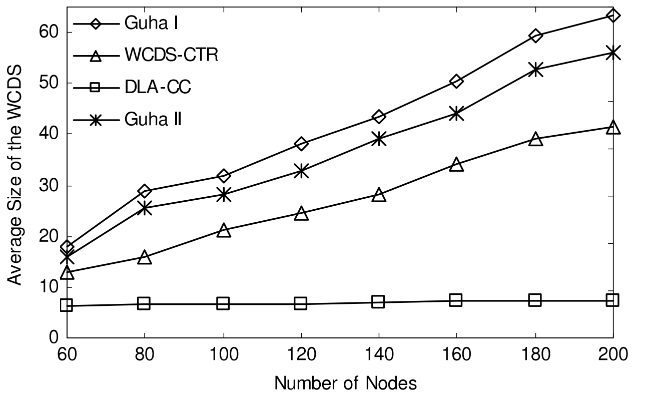

Fig. 3. Average size of the WCDS constructed by the centralized algorithms, when distance R is 15.

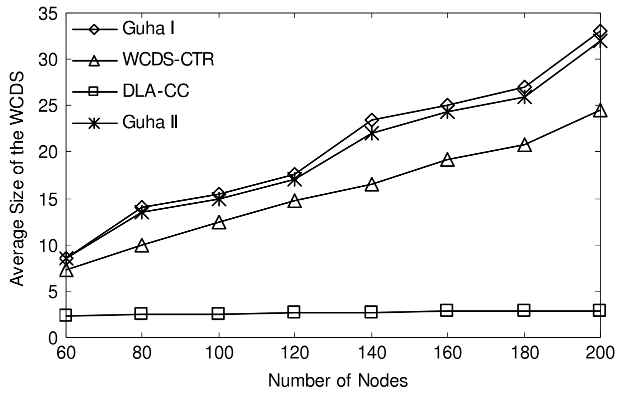

Fig. 4. Average size of the WCDS as a function of number of nodes when distance R is 30.

In our simulation experiments, each host is modeled as an infinite-buffer, store-and forward queuing station. The IEEE 802.11 Distributed Coordination Function (DCF) with Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) 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. We first randomly distribute the hosts in the simulation square area. After distributing the hosts, we will check whether they form a connected graph. We only construct the WCDS (cluster-head set) on the connected graphs. The results presented in this article are averaged over 100 independent runs on the connected graphs. In all simulations, the learning parameter is fixed at 0.2, and each algorithm is terminated when the probability of choosing the WCDS becomes greater than 0.90. That is P in DLA-CC and *pchs* in DLA-DC are set to 0.9.

6.1. Experiment I

This experiment is conducted to study the performance of DLA-CC in terms of the dominating set size. The results of DLA-CC are compared with those of Guha I [19], Guha II [19] and WCDS-CTR [13]. To generate the random graphs, a number of vertices are uniformly distributed in a two-dimensional simulation area of size 100×100 at random. We assume that a link is established between every two vertices, if the distance between them is not longer than R. In this experiment, we first set distance R to 15, and change the number of nodes from 60 to 200 with increment step of 20. The obtained results are shown in Fig. 3.

From the results shown in Fig. 3, it is clear that Guha I always constructs the largest WCDSs, and so is ranked lower the other algorithms. The size of the WCDSs generated by Guha II is only slightly smaller than Guha I, but considerably larger than WCDS-CTR. The results also show that the size of the WCDS constructed by DLA-CC is significantly smaller than that of WCDS-CTR. Therefore, DLA-CC outperforms the other algorithms. The results show that the number of dominators increases as the number of nodes increases.

Then, we change distance R to 30 and repeat the same experiments. The results are shown in Fig. 4. Like Fig. 3, the results shown in Fig. 4 show that the proposed centralized algorithm (i.e., DLA-CC) considerably outperforms the other algorithms, WCDS-CTR is ranked below DLA-CC, and Guah I and Guha II lag far behind. Comparing the results shown in Fig. 4 with Fig. 3, we observe that for all algorithms the average size of the WCDS becomes smaller when R increases. The reason for this reduction is that the number of neighbors of a dominator increases when distance R increases. Therefore, the graph nodes can be thoroughly dominated by a less number of the dominators. From Figs. 3 and 4, it can be seen that the gap between the curves for our proposed algorithm and the curves for the other centralized algorithms becomes significant as the number of nodes increases.

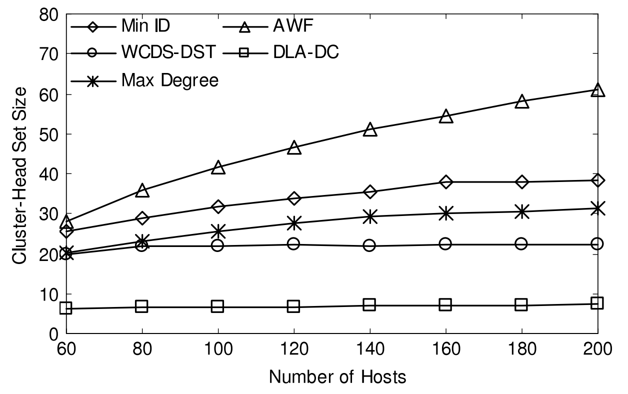

Fig. 5. Average cluster-head set size as a function of the network size when the radio transmission range is 15.

6.2. Experiment II

This experiment is conducted to study the performance of DLA-DC in terms of the cluster-head set size. The network size, simulation square area size, and the radio transmission range are three correlated parameters that affect the number of clusters. In this experiment, we investigate the impact of the above mentioned parameters on the proposed clustering algorithm. The results of DLA-DC are compared with those of Min ID [22], Max Degree [22], WCDS-DST [13], and AWF [5] which are the best WCDS-based clustering algorithms.

In Experiment II, the radio transmission range is initially set to 15 and the number of hosts changes from 60 to 200 with increment step of 20. The simulation square area size is fixed at 100×100. The obtained results are shown in Fig. 5. Then, we change the radio transmission range to 30 and repeat the same experiment. The results are shown in Fig. 6.

From Figs. 5 and 6, it is clear that among the previous clustering algorithms, WCDS-DC and AWF construct the smallest and largest cluster-head sets, respectively. The results show that the size of the cluster-head set constructed by WCDS-DC is nearly four times larger than that constructed by our proposed algorithm, DLA-DC. For instance, the size of the cluster-head set constructed by DLA-DC is 6.76, when the number of hosts is 100 and the radio transmission

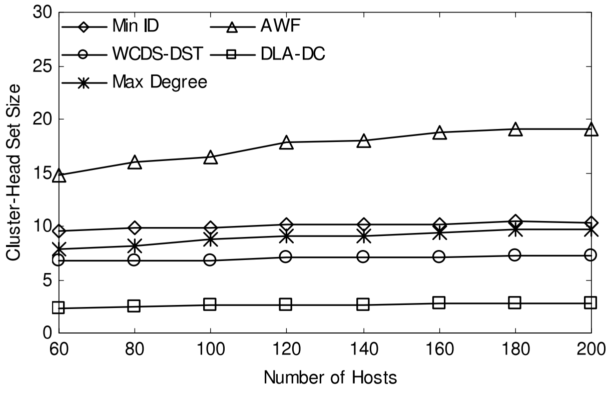

Fig. 6. Average cluster-head set size as a function of the network size when the radio transmission range is 30.

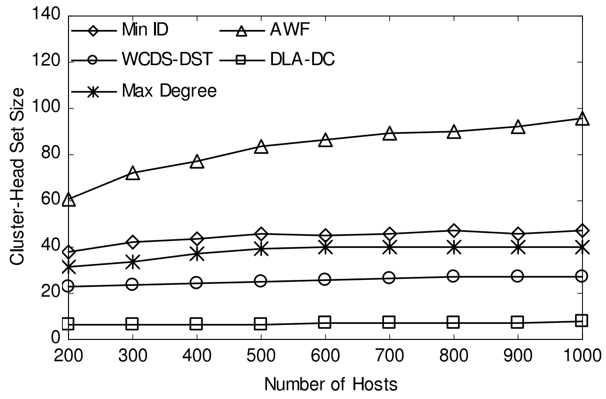

Fig. 7. Average cluster-head set size as a function of the network size when the radio transmission range is 15.

range is 15, while that of WCDS-DC is 21.9, which is 3.2 times larger than our obtained result. Therefore, we conclude that our proposed algorithm significantly outperforms the other algorithms in terms of the cluster-head set size. From the results shown in these figures, we also see that the average size of the cluster-head sets constructed by Max Degree is slightly smaller than that of Min ID, especially as the number of hosts increases. So, Max Degree is ranked lower than WCDS-DC.

Comparing the results shown in Fig. 5 with Fig. 6, we find that the size of the cluster-head set decreases as the number of hosts increases. As mentioned in Experiment I, this is because a cluster-head with a larger radio transmission range is capable of covering more hosts and so the whole network can be covered by a smaller cluster-head set (or by a less number of cluster-heads). When the size of the simulation square area grows, the hosts are distributed in a larger area. This causes a smaller number of hosts that can be covered by each cluster-head. Therefore, it is expected that the size of the cluster-head set increases as the simulation square area increases.

To study the scalability of the clustering algorithms, they are tested on the dense network graphs. In this part of Experiment II, the number of hosts ranges from 200 to 1000 with increment step of 100, and the radio transmission range is set to 15 and 30, respectively. The simulation results are depicted in Figs. 7 and 10. Comparing the average size of the cluster-head set constructed by DLA-DC with that of the other clustering algorithms, we observe that the proposed clustering algorithm significantly outperforms the other algorithms. It can be seen that the ranking given for

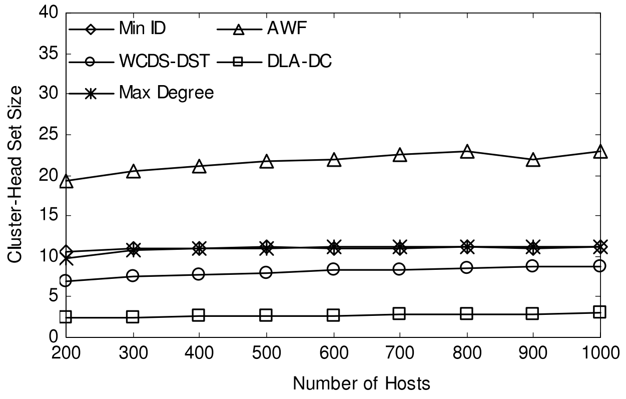

Fig. 8. Average cluster-head set size as a function of the network size when the radio transmission range is 30.

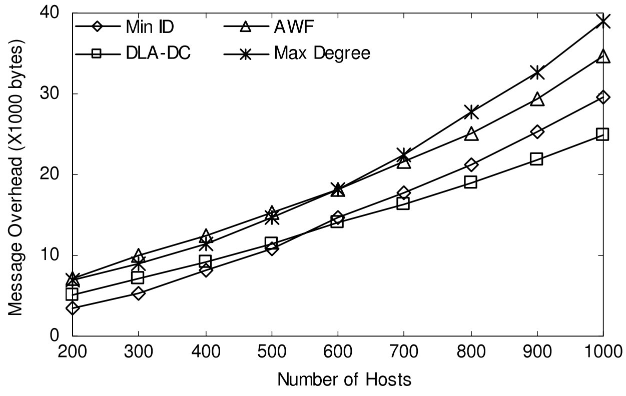

Fig. 9. Message overhead as a function of the network size when the radio transmission range is 15.

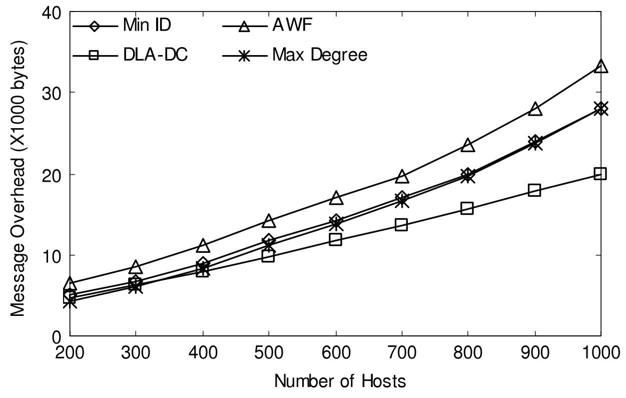

Fig. 10. Message overhead as a function of the network size when the radio transmission range is 30.

the clustering algorithms with the sparse network graphs remains unchanged, and DLA-DC is ranked above the other algorithms and AWF below them. As expected, comparing the results shown in Figs. 7 and 8, we observe that the size of the cluster-head set reduces as the radio transmission range increases.

6.3. Experiment III

In Ad Hoc networks, the hosts suffer from the strict resource limitations (e.g., power and bandwidth resources). Therefore, message overhead is one of the key issues in designing the

Ad Hoc protocols that must be kept as low as possible. This experiment is conducted to study the message overhead of DLA-DC in comparison with Min ID, Max Degree, and AWF. Figs. 9 and 10 show the message overhead (in bytes) as a function of the number of hosts when the radio transmission range is set to 15 and 30, respectively. As shown in Figs. 9 and 10, for all clustering algorithms the message overhead increases as the number of hosts (network size) increases. From the results shown in Fig. 9, it can be seen that Min ID outperforms AWF and Max degree. The results show that Min ID also performs better than DLA-DC, when the number of hosts is less than 600. However, the message overhead of DLA-DC becomes smaller than that of Min ID as the number of hosts exceeds 600. Comparing the results shown in Figs. 9 and 10 for different radio transmission ranges, we observe that the message overhead of all algorithms decreases as the radio transmission range increases. As shown in Fig. 10, in most cases DLA-DC outperforms the other algorithms, Min ID and Max Degree are ranked below it, and AWF has the highest message overhead. It can also be seen that the gap between the curves for DLA-DC and Max Degree (or Min ID) becomes significant as the number of hosts increases.

7. Conclusion

The network clustering is a method by which the hosts are hierarchically organized on the basis of the proximity, and the hierarchical structure thus formed abstracts the large scale networks so that the hosts can be simply and locally organized. In this article, we first proposed a DLA-based centralized algorithm (DLA-CC) for solving the minimum WCDS problem. Since finding the WCDS is a well-known approach for clustering the wireless Ad Hoc networks, we also proposed an approximation algorithm (DLA-DC) for clustering the wireless Ad Hoc networks. The proposed clustering algorithm is a distributed implementation of DLA-CC in which the dominators assume the role of the cluster-heads and their one-hop neighbors play the role of the cluster members. In this article, we also found an upper bound on the running time and message complexity of the proposed clustering algorithm for finding a near optimal size cluster-head set. It was shown that by a proper choice of the learning rate of the clustering algorithm, a trade-off between the running time and message complexity of algorithm with the cluster-head set size (clustering optimality) can be made. The simulation results showed the superiority of the proposed algorithms over the previous methods.

References

[1] J. Akbari Torkestani, M.R. Meybodi, An inelligent backbone formation algorithm for wireless ad hoc networks based on distributed learning automata, Computer Networks (2009) in press, Elsevier Publishing Company.

[2] J. Akbari Torkestani, M.R. Meybodi, A new vertex coloring algorithm based on variable action-set learning automata, Journal of Computing and Informatics (2009) in press.

[3] J. Akbari Torkestani, M.R. Meybodi, 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, pp. 672–676.

[4] J. Akbari Torkestani, M.R. Meybodi, 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, pp. 643–647.

[5] K.M. Alzoubi, P.J. Wan, O. Frieder, Maximal independent set, weakly connected dominating set, and induced spanners for mobile Ad-Hoc networks, International Journal of Foundations of Computer Science 14 (2) (2003) 287–303.

[6] S. Basagni, M. Mastrogiovanni, C. Petrioli, A performance comparison of protocols for clustering and backbone formation in large scale Ad-Hoc network, in: Proceedings of the First IEEE International Conference on Mobile Ad-Hoc and Sensor Systems, MASS'2004, 2004, pp. 70–79.

[7] H. Beigy, M.R. Meybodi, Utilizing distributed learning automata to solve stochastic shortest path problems, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 14 (2006) 591–615.

[8] H. Beigy, M.R. Meybodi, A learning automata-based dynamic guard channel scheme, in: Lecture Notes on Information and Communication Technology, vol. 2510, Springer Verlag, 2002, pp. 643–650.

[9] H. Beigy, M.R. Meybodi, An adaptive uniform guard channel algorithm: A learning automata approach, in: LANCES, in: Lecture Notes in Intelligent Data Engineering and Automated Learning, vol. 2690, Springer Verlag, Germany, 2003, pp. 405–409.

[10] Y.P. Chen, A.L. Liestman, Maintaining weakly connected dominating sets for clustering Ad-Hoc networks, Ad-Hoc Networks 3 (2005) 629–642.

[11] Y.P. Chen, A.L. Liestman, Maintaining weakly-connected dominating sets for clustering Ad-Hoc networks, Ad-Hoc Networks 3 (2005) 629–642.

[12] Y.P. Chen, A.L. Liestman, A Zonal algorithm for clustering Ad-Hoc networks, International Journal of Foundations of Computer Science 14 (2) (2003) 305–322.

[13] Y.Z. Chen, A.L. Listman, 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, 2002, pp. 157–164.

[14] B.N. Clark, C.J. Colbourn, D.S. Johnson, Unit disk graphs, Discrete Mathematics 86 (1990) 165–177.

[15] B. Das, V. Bharghavan, Routing in Ad-Hoc networks using minimum connected dominating sets, in: IEEE International Conference on Communications, ICC'97, 1997.

[16] O. Dousse, F. Baccelli, P. Thiran, Impact of interferences on connectivity in Ad-Hoc networks, IEEE/ACM Transactions on Networking 13 (2) (2005) 425–436.

[17] R.G. Gallager, P.A. Humblet, P.M. Spira, A distributed algorithm for minimum weight spanning trees, ACM Transaction on Programming Languages and Systems 5 (1983) 66–77.

[18] R. Ghosh, S. Basagni, Mitigating the impact of node mobility on Ad-Hoc clustering, Journal of Wireless Communications and Mobile Computing 8 (2008) 295–308.

[19] S. Guha, S. Khuller, Approximation algorithms for connected dominating sets, Algorithmica 20 (4) (1998) 374–387.

[20] P. Gupta, P.R. Kumar, The capacity of wireless networks, IEEE Transaction on Information Theory 46 (2) (2000) 388–404.

[21] M. Haleem, R. Chandramouli, Adaptive downlink scheduling and rate selection: A cross layer design, special issue on mobile computing and networking, IEEE Journal on Selected Areas in Communications 23 (6) (2005).

[22] B. Han, W. Jia, Clustering wireless Ad-Hoc networks with weakly connected dominating set, Journal of Parallel and Distributed Computing 67 (2007) 727–737.

[23] S. Lakshmivarahan, M.A.L. Thathachar, Bounds on the convergence probabilities of learning automata, IEEE Transactions on Systems, Man, and Cybernetics SMC-6 (1976) 756–763.

[24] M.V. Marathe, H. Breu, H.B. Hunt III, S.S. Ravi, D.J. Rosenkrantz, Simple heuristics for unit disk graphs, Networks 25 (1995) 59–68.

[25] K.S. Narendra, K.S. Thathachar, Learning Automata: An Introduction, Printice-Hall, New York, 1989.

[26] K.S. Narendra, M.A.L. Thathachar, On the behavior of a learning automaton in a changing environment with application to telephone traffic routing, IEEE Transactions on Systems, Man, and Cybernetics SMC-l0 (5) (1980) 262–269.

[27] P. Nicopolitidis, G.I. Papadimitriou, M.S. Obaidat, A.S. Pomportsis, Carrier-sense-assisted adaptive learning MAC protocol for distributed wireless LANs, International Journal of Communication Systems 18 (7) (2005) 657–669.

[28] P. Nicopolitidis, G.I. Papadimitriou, A.S. Pomportsis, Exploiting locality of demand to improve the performance of wireless data broadcasting, IEEE Transactions on Vehicular Technology 55 (4) (2006) 1347–1361.

[29] P. Nicopolitidis, G.I. Papadimitriou, A.S. Pomportsis, Learning-automata-based polling protocols for wireless LANs, IEEE Transactions on Communications 51 (3) (2003) 453–463.

[30] P. Nicopolitidis, G.I. Papadimitriou, A.S. Pomportsis, Distributed protocols for Ad-Hoc wireless LANs: A learning-automata-based approach, Ad-Hoc Networks 2 (4) (2004) 419–431.

[31] R. Rajaraman, Topology control and routing in Ad-Hoc networks: A survey, SIGACT News 33 (2) (2002) 60–73.

[32] B.V. Ramana, C.S.R. Murthy, 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, 2005, pp. 454–464.

[33] M.A.L. Thathachar, B.R. Harita, Learning automata with changing number of actions, IEEE Transactions on Systems, Man, and Cybernetics SMG17 (1987) 1095–1100.

[34] M.A.L. Thathachar, V.V. Phansalkar, Convergence of teams and hierarchies of learning automata in connectionist systems, IEEE Transactions on Systems, Man and Cybernetics 24 (1995) 1459–1469.

[35] M.A.L. Thathachar, P.S. Sastry, A hierarchical system of learning automata that can learn the globally optimal path, Information Science 42 (1997) 743–766.

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

Mohammad Reza Meybodi received his B.S. and M.S. degrees in Economics from Shahid Beheshti University in Iran during 1973 and 1977, respectively. He also received his M.S. and Ph.D. degrees from Oklahoma University, USA during 1980 and 1983, respectively in Computer Science. Presently, he is a Full Professor in Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran. Prior to the present 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.