Clustering the Wireless Ad Hoc Networks: A Distributed Learning Automata Approach

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

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

Abstract

In ad hoc networks, the performance is significantly degraded as the size of the network grows. The network clustering is a method by which the nodes are hierarchically organized on the basis of the proximity and thus the scalability problem is alleviated. Finding the weakly connected dominating set (WCDS) is a well-known approach, proposed 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 have been proposed. In this paper, an approximation algorithm based on distributed learning automata is first proposed for finding a near optimal solution to the minimum WCDS problem in a unit disk graph. Then, a distributed learning automata-based algorithm is proposed for clustering the wireless ad hoc networks. This clustering method is a generalization of the algorithm proposed for solving the WCDS problem, in which the dominator nodes and their closed neighbors assume the role of the cluster-heads and cluster members, respectively. The proposed clustering algorithm, in an iterative process tries to find a policy that determines a cluster-head set with the minimum cardinality for the network. In this paper, the worst case running time and message complexity of the clustering algorithm to find a $1/(1-\varepsilon)$ optimal cluster-head set are computed. It is 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. For both algorithms, the simulation results show that they outperform the best existing algorithms in terms of the number of hosts (nodes) in the cluster-head set (dominating set).

Keywords: Wireless ad hoc networks, clustering, weakly connected dominating set, distributed learning automata

1 Introduction

A wireless ad hoc network is a multi hop wireless communication network supporting a collection of mobile hosts. There is no fixed infrastructure and no central administration and the mobile hosts can form a temporary network infrastructure in an ad hoc fashion. Two hosts can directly communicate when they are within transmission range of each other, and indirectly through relaying by the intermediate hosts. In an ad hoc network, each host assumes the role of a router and relays the packets toward the final destinations, if a source can not directly send the packets to a final destination due to the limitation of the radio transmission range. Since the wireless ad hoc networks exhibit severe resource constraints such as the bandwidth and power limitations, network topology changes, and the lack of the fixed infrastructures and consequently, centralized administrations, to achieve good performance in ad hoc networks, the load on the hosts should be kept as low as possible[3, 22].

The network performance is significantly degraded as the network becomes larger and the theoretical analysis [2] implies that even under the optimal circumstances, the throughput for each host declines

rapidly towards zero as the number of hosts is increased. Among the solutions proposed for solving the scalability problem in ah doc networks, the network clustering approach has attracted a lot of attention in study such large scale networks. 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 [4, 7]. 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 WCDS of the network graph is one of the most 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 [12, 14].

Clustering the ad hoc networks based on the weakly connected dominating sets was first proposed by Chen and Listman [7, 32]. The distributed approximation clustering algorithm proposed by Chen and Listman is also inspired by Guha and Khuller's centralized approximation algorithm [11] for finding small connected dominating sets (CDS). Guha and Khuller [11] proposed two centralized greedy heuristic algorithms with bounded performance guarantees for connected dominating set formation. In the first algorithm, the connected dominating set is grown from one node outward, and in the second algorithm, a WCDS is constructed, and then the intermediate nodes are selected to create a CDS. Chen and Liestman also proposed a zonal algorithm [1, 8], 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. [10]. Han [3] 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. Alzoubi et al. [9] 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 labelling strategy, and then modifies the MIS by selecting one intermediate node between each pair of dominators separated by exactly three hops.

In this paper, two distributed learning automata-based algorithms are proposed to solve the minimum WCDS problem and to cluster the wireless ad hoc networks, respectively. The proposed clustering algorithm is aimed at finding a near optimal solution to the minimum WCDS problem in which the dominator nodes play the role of the cluster-heads and their one-hop neighbors assume the role of the cluster members. In both algorithms, a network of the learning automata, isomorphic to the network topology graph (or unit disk graph) is formed by assigning a learning automaton to each node. At each iteration, the learning automata randomly choose their actions to form a WCDS, and the selected actions are evaluated by the random environment. After a number of iterations, the learning automata tend to a common policy that determines the minimum size cluster-head set (or WCDS) with the highest probability. Furthermore, in this paper, the worst case running time and message complexity of the proposed clustering algorithm for finding a $1/(1-\varepsilon)$ optimal cluster-head set are computed. 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. For both algorithms, the simulation results show that they outperform the best existing algorithms in terms of the number of hosts (nodes) in the cluster-head set (dominating set).

The rest of the paper is organized as follows. In the next section, the WCDS problem formulation, learning automata and some preliminaries are presented. In section 3, a distributed learning automata-based approximation algorithm is proposed to solve the minimum WCDS problem in an arbitrary unit disk graph. In section 4, a clustering algorithm based on distributed learning automata for wireless ad hoc networks is proposed. The worst case running time and message complexity of the 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 paper.

2 Preliminaries

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 $N_G[v]$ of a host v in graph G consists of the hosts adjacent to v and host v itself. The closed neighborhood $N_G[S]$ of the set S is the union $\bigcup_{v \in S} N_G[v]$. The subscript G can be omitted if the meaning is clear from the context. 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 connected dominating set (CDS) S of a given graph G is a dominating set whose induced sub graph, denoted $< S >$, is connected, and a minimum CDS (MCDS) is a CDS with the minimum cardinality. A MCDS forms a virtual backbone 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 [5, 6], and even for a unit disk graph, the problem of finding a MCDS is also NP-Hard [6].

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

Figure 1.A. a sample unit disk graph

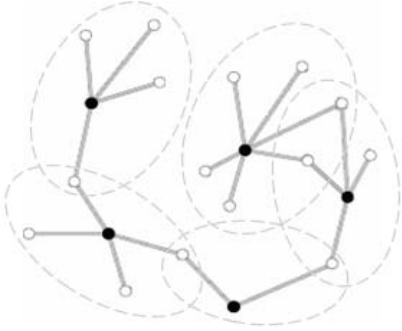

Figure 1.B. the weakly connected dominating set

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 (e.g., IP address) and also needs to know the ID 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 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 [15-21] is an adaptive decision-making unit that improves its performance by learning how to choose the optimal action from a finite set of allowed actions through repeated interactions with a random environment. The action is chosen at random based on a probability distribution kept over the action set and at each instant the given action is served as the input to the random environment. The environment responds the taken action in turn with a reinforcement signal. The action probability vector is updated based on the reinforcement feedback from the environment. The objective of a learning automaton is to find the optimal action from the action set so that the average penalty received from the environment is minimized [15].

The environment can be described by a triple $E \equiv \{\alpha, \beta, c\}$, where $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ represents the finite set of the inputs, $\beta \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ denotes the set of the values can be taken by the reinforcement signal, and $c \equiv \{c_1, c_2, ..., c_r\}$ 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 $[a, b]$. The relationship between the learning automaton and its random environment has been shown in figure 2.

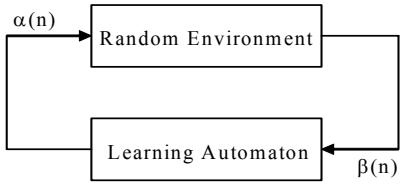

Figure 2. The relationship between the learning automaton and its random environment

Learning automata can be classified into two main families [15-20]: fixed structure learning automata and variable structure learning automata. Variable structure learning automata are represented by a triple $< \underline{\beta}, \underline{\alpha}, T >$, where $\underline{\beta}$ is the set of inputs, $\underline{\alpha}$ is the set of actions, and T is learning algorithm. The learning algorithm is a recurrence relation which is used to modify the action probability vector. Let $\alpha(k)$ and $\underline{p}(k)$ denote the action chosen at instant k and the action probability vector on which the chosen action is based, respectively. The recurrence equation shown by (1) and (2) is a linear learning algorithm by which the action probability vector $\underline{p}$ is updated. Let $\alpha_i(k)$ be the action chosen by the automaton at instant k.

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

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

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

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

Learning automata is proved to perform well in the dynamic environments of wireless, ad hoc and sensor networks. Haleem and Chandramouli [23] 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. [24] 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 [25] 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 [26, 27]. Ravana and Morthy [28] 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 [29, 30, 31].

2.2.2. Distributed Learning Automata

A Distributed learning automata (DLA) [21] is a network of the learning automata which collectively cooperate to solve a particular problem. Formally, a DLA can be defined by a quadruple $< A, E, T, A_0 >$, where $A=\{A_1,...,A_n\}$ is the set of learning automata, $E \subset A \times A$ is the set of the edges in which edge $e_{(i,j)}$ corresponds to the action α_{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 . An example of a DLA has been shown in figure 3.

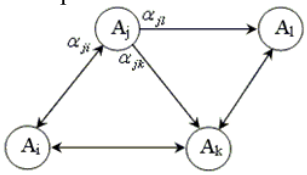

Figure 3. Distributed learning automata

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

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 [17] that a learning automaton with a changing number of actions is absolutely expedient and also ε-optimal, when the reinforcement scheme is L_{R-I}. Such an automaton has a finite set of n actions, $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$. $A = \{A_1, A_2, ..., A_m\}$ denotes the set of action subsets and $A(k) \subseteq \alpha$ is the subset of all the actions can be chosen by the learning automaton, at each instant k. The selection of the particular action subsets is randomly made by an external agency according to the probability distribution $\psi(k) = \{\psi_1(k), \psi_2(k), ..., \psi_m(k)\}$ defined over the possible subsets of the actions, where $\psi_i(k) = prob[A(k) = A_i \mid A_i \in A, 1 \le i \le 2^n - 1]$. $\hat{p}_i(k) = prob[\alpha(k) = \alpha_i \mid A(k), \alpha_i \in A(k)]$ is the probability of choosing action α_i, conditioned on the event that the action subset $A(k)$ has already been selected and also $\alpha_i \in A(k)$. The scaled probability $\hat{p}_i(k)$ is defined as

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

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 equation (3). The automaton then randomly selects one of its possible actions according to the scaled action probability vector $\hat{p}(k)$. Depending on the response received from the environment, the learning automaton updates its scaled action probability vector. Note that the probability of the available actions is only updated. Finally, the probability vector of the actions of the chosen subset is rescaled as $p_i(k+1) = \hat{p}_i(k+1) \cdot K(k)$, for all $\alpha_i \in A(k)$. The absolute expediency and $\varepsilon -$optimality of the method described above have been proved in [17].

3 DLA-Based WCDS Formation Algorithm

In this section, a distributed learning automata-based approximation algorithm is proposed for finding a near optimal solution to the minimum WCDS problem described in section 2. In this algorithm, a network of the learning automata isomorphic to the input UDG is first formed by assigning to each vertex v_i of the graph a learning automaton, A_i. The resulting network of the learning automata can be described by a duple $< \underline{A}, \underline{\alpha} >$, where $\underline{A} = \{A_1, A_2, ..., A_m\}$ denotes the set of learning automata corresponding to the vertex set, and $\underline{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_m\}$ denotes the set of actions, in which $\alpha_i = \{\alpha_{i,1}, \alpha_{i,2}, ..., \alpha_{i,ri}\}$ defines the set of actions can be taken by the learning automata A_i and initially includes the set of all learning automata except automata A_i and its neighbors, for each $\alpha_i \in \underline{\alpha}$. For optimizing the behavior of the learning automata, the proposed algorithm deals with the automata with changing number of actions [17]. Each vertex (automaton) can be in one of two states *active* and *passive*, and it is initially set to the passive state. The proposed algorithm, which we call it DLA-CC, consists of a number of stages, and at each stage, the selected vertices are activated and added to the WCDS being formed in that stage. The proposed algorithm iteratively constructs a number of WCDSs and updates the action probability vectors until it finds a near optimal solution to the minimum WCDS problem with a probability higher than a pre-specified threshold. Stage k of the proposed algorithm is briefly described in the following steps:

Step 1. ***WCDS formation***

The first dominator vertex is randomly selected (say activated), denoted as v_i and added to the WCDS which is being constructed at stage k.

While (The cardinality of the constructed dominatee set is less than the number of vertices of The input graph) **do**

- The neighbors of activated vertex v_i are added to the set of dominatee vertices.
- Learning automaton A_i (assigned to activated vertex v_i) changes its number of actions (or scales its action probability vector) by disabling the actions corresponding to all the members of the dominatee set.
- Learning automaton A_i randomly chooses one of its actions according to its scaled action probability vector, activates its corresponding vertex, denotes it v_i, and adds it to the WCDS.

Step 2. ***Comparing the cardinality of the constructed WCDS with a dynamic threshold***

- Let dynamic threshold T_k denotes the cardinality of the minimum size WCDS constructed until stage k.
- Dynamic threshold T_k is updated to the cardinality of the constructed WCDS, if it is lager than the cardinality of the WCDS.

Step 3. ***Updating the action probability vectors***

- Depending on the result of the comparison in step 2, all the activated learning automata, using a L_{R-I} reinforcement scheme, reward their chosen actions if the cardinality of the constructed WCDS is less than or equal to the dynamic threshold T_k, and penalize them otherwise.

Step 4. ***Stopping Condition***

- The process of constructing the WCDSs and updating the action probabilities are repeated until the product of the probability of choosing the vertices of the constructed WCDS is greater than a certain threshold or the number of constructed WCDS exceeds a pre-specified threshold. The WCDS which is formed last before stopping the algorithm is the WCDS with the minimum cardinality among all WCDSs of the graph. The proposed algorithm has been described in more detail in figure 4.

Algorithm DLA-CC

Input: Unit Disk Graph $G(V, E)$, Threshold K, P

Output: The minimum size WCDS

Assumptions:

Assign a learning automaton to each vertex v_i and initially set it in a passive state

Let α_i denotes the set of actions for each automaton A_i, and contains the set of all vertices of graph G except vertex v_i

Begin Algorithm

Let T_k be the dynamic threshold, which is the cardinality of the smallest WCDS constructed until stage k, and initially set to n

Let k denotes the stage number and is initially set to 0

Let A_0 be the initial learning automaton whose action set is the set of all the vertices in graph G

Repeat

Let δ_k be the set of dominators selected at stage k and initially set to null

Let d_k denotes the set of dominatees, selected at stage k and is initially set to null

Automaton A_0 randomly chooses one of its actions according to its action probability vector, activates it, denotes v_i and adds to δ_k

While (The cardinality of set d_k is less than n) **do**

Add the neighbors of dominator v_i to d_k

Update the action probability vector of learning automaton A_i by disabling the actions corresponded to all the vertices in d_k

Dominator v_i randomly chooses one of its actions according to its action probability vector, activates it, denotes v_i and adds to δ_k

End while

Find the WCDS selected at stage k

Compute the cardinality of the selected WCDS

If (The cardinality of the selected WCDS is less than the dynamic threshold T_k) **Then**

Reward the actions chosen by the activated learning automata

Set the dynamic threshold T_k to the cardinality of the selected WCDS

Else

Penalize the actions chosen by the activated learning automata

End if

Increment stage number k

Enable all the actions disabled during the current iteration and update the action probability vector

Until (The probability of choosing the WCDS is greater than a pre-specified threshold P or the stage number k is greater than threshold K)

End Algorithm

Figure 4. Algorithm DLA-CC for solving the minimum WCDS problem

4 The Proposed Clustering Algorithm

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 [3]. Therefore, in this section, a distributed approximation algorithm based on distributed learning automata, 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 and knows its neighbors' ID. In this algorithm, to form the action-set of each learning automaton, its corresponding host 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. Therefore, the action-set size of each learning automaton is strongly dependent to the network density. Assuming that the hosts are uniformly distributed in the network (a uniform network density), each learning automaton has $n\pi R^2 / A$ actions, each of initial probability $A/n\pi R^2$. Action $\alpha_{i,j}$ corresponds to the selection of host h_j as a cluster-head by learning automaton A_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 termination 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*.

A *READY* message, which contains *MAX_ITERATION* and *PCHS*, is initially flooded within the network to inform the hosts of the cluster formation (or re-clustering) start. Each host, upon receiving the *READY* message, calls the *WCDS* procedure. During the *WCDS* procedure, each host may receive from or send to the other hosts the following messages: *INITIALIZATION, ACTIVATION*, *REWARDING*, *PENALIZING* and *CLUSTERING* message. After the *READY* message is sent, one of the hosts is randomly selected and denoted as the initial host. An *INITIALIZATION* message is then sent to the initial host and the process of the cluster formation is continued by receiving the *INITIALIZATION* message as described below.

I. INITIALIZATION MESSAGE

When a given host h_i receives an *INITIALIZATION* message, it initially updates the *CLUSTER_HEAD_SET* by adding its ID. The initial host then adds its one-hop neighbors' ID to the *CLUSTER_LIST*. Learning automaton L_i (corresponding to host h_i) disables the actions associated with the selected cluster-heads in its action set. In this clustering algorithm, the aim of using the variable action set learning automata is to avoid selecting the cluster-heads by which no more hosts can be added to the *CLUSTER_LIST*. For this purpose, learning automaton L_i scales its action probability vector by disabling the actions associated with all the selected cluster-heads as described in section 2.2.3 on the variable action set learning automata, if it has such actions. Then, it chooses one of its actions according to the scaled action probability vector. The probability with which learning automaton L_i chooses its action is added to the *PROB_VECTOR*. Finally, an *ACTIVATION* message is sent to the host (new cluster-head) corresponding to the chosen action.

II. ACTIVATION MESSAGE

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 as a new cluster-head into the *CLUSTER_HEAD_SET*, if it has at least a one-hop neighbor not to be in the *CLUSTER_LIST*. To update the *CLUSTER_LIST* it adds its one-hop neighbors' ID to this list. The action set of learning automaton L_i is updated by disabling the actions associated with the selected hosts as described earlier. In this case, if there is no more actions can be taken by learning automaton L_i, the cardinality of the *CLUSTER_HEAD_SET* is calculated. If the cardinality is less than the dynamic threshold *MIN_SIZE*, the dynamic threshold is set to the cardinality of the selected set (*CLUSTER_HEAD_SET*) and all the chosen actions of the activated automata are rewarded by sending back a *REWARDING* message, otherwise (cardinality is larger than the dynamic threshold) they are penalized by sending back a *PENALIZING* message.

To verify the stopping condition of the cluster formation process the probability of choosing the *CLUSTER_HEAD_SET* is calculated as the product of the probability of choosing the selected cluster-heads based on the information contained in *PROB_VECTOR*. If this probability is less than the certain threshold *PCHS* and *ITERATION_NUM* does not exceed than a per-specified threshold *MAX_ITERATION*, host h_i increments the *ITERATION_NUM* and randomly chooses one of its actions (or hosts), according to its action probability vector. Then, it sends an *INITIALIZATION* message to the host corresponding to the taken action to start a new iteration. Otherwise it generates a *CLUSTERING* message including the last selected *CLUSTER_HEAD_SET* and broadcasts it in the network.

If there exist actions can be taken by learning automaton L_i, it chooses one of its actions as a new cluster-head, updates *PROB_VECTOR* by adding the probability of choosing the action, and sends an *ACTIVATION* message to the chosen cluster-head.

III. CLUSTERING MESSAGE

A *CLUSTERING* message includes the *CLUSTER_HEAD_SET* selected during the last iteration. When host h_i receives a *CLUSTERING* message, it is noticed that the clustering process has been completed, and so accepts the list of the cluster-heads, contained in the *CLUSTERING* message, as a new cluster-head set. It will then terminate the *WCDS* procedure.

IV. REWARDING MESSAGE

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

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

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

V. PENALIZING MESSAGE

Since the reinforcement scheme by which the learning automata update their action probability vectors is L_{R-I}, the action probabilities of the activated learning automata remain unchanged when they receive a *PENALIZING* message. In this case, the disabled actions of each activated learning automaton are enabled again.

5 Complexity Analysis of the Clustering Method

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 $\frac{1}{1-\varepsilon} \cdot |\mathrm{OPT}|$ (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 parameter 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 $p_j = A/n\pi R^2$ be the initial probability of choosing host h_j as a cluster-head, where A denotes the square area, R denotes the radio transmission range, and n is the number of hosts. 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 equation (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 $q_i^g(k)$ (i.e., *PCHS* in DLA-DC) 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

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

Hence, we have

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

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

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

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 equation (10) to be satisfied. From equation (8) we have

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

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 equation (12) is a geometric series that sums up to $a\cdot\left(1-(1-a)^k / 1-(1-a)\right)$, where $|1-a|<1$. Therefore, we have

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

From equation (10) and (13) we have

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

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 equation (14), we derive

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

and thus the number of times ω_i is rewarded 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 ω_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 equation (10)) is obtained as

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

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

$$\mathrm{K} = \left(\frac{2}{1-\varepsilon+q_i^g}\right)\cdot k$$

By substituting k from equation (15) and we have

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

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

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

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

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

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 parameter of algorithm, a denotes the learning rate of algorithm, n is the number of hosts in the network graph, and r is the number of 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

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

Furthermore, as described in section 5, at each iteration of the clustering algorithm, the number of messages 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

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

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

It can be seen that by a proper choice of the learning rate of the clustering algorithm, a trade-off between the running time and message complexity of the algorithm with the cluster-head set size (clustering optimality) can be made. 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

To study the performance of the proposed algorithms, we have conducted two groups of simulation and evaluated the obtained results in terms of the size of the cluster-head set (or the cardinality of the WCDS) in comparison with that of the best existing algorithms. Since the first proposed algorithm (DLA-CC) is a centralized algorithm to solve the minimum (size) WCDS problem in a unit disk graph, the first group of simulation is concerned with investigating the efficiency of the first proposed algorithm in comparison with the centralized WCDS-based clustering algorithms. Therefore, algorithm DLA-CC is compared with the centralized algorithms Guha I [11], Guha II [11] and WCDS-CTR [7]. The second group of simulation compares the results of the proposed distributed clustering algorithm (DLA-DC) with two distributed zonal algorithms Min ID[3] and Max Degree[3], distributed algorithm WCDS-DST[7] and the second proposed algorithm in [9](AWF). The first algorithm presented in [9] is extremely message expensive in practice [13], and we do not consider it in the following performance evaluation. Since algorithm DLA-DC is proposed to cluster the ad hoc networks with a large number of hosts, we have also conducted some simulation experiments on large and dense graphs to evaluate the scalability of algorithm.

In our simulation studies, we first randomly distribute the hosts in a $m\times m$ square area, where m is the length of the square edge. All hosts are assumed to have the same radio transmission range R, and each host can directly communicate with other hosts within the distance R. For simplicity, we assume that all links are bidirectional and symmetric. After distributing the hosts in the square area, we will check whether they form a connected graph. We only construct WCDSs on the connected graphs and measure the sizes of the generated WCDSs. The radio transmission range of a host and the number of hosts in the network are two correlated parameters by which the size of the constructed WCDS can be affected. Thus, we study how these parameters impact the size of WCDS constructed by different algorithms.

The radio transmission range for a host is assumed to be 15 and 30, respectively, and a given number of hosts are randomly and uniformly distributed in a square simulation area of size 100×100 units, which remains unchanged in all experiments. We vary the number of hosts from 60 to 200 and measure the size of WCDSs constructed by different algorithms. To study the scalability of algorithm DLA-DC, we have conducted some simulations on the relatively dense network graphs, when the number of hosts ranges from

200 to 1000. The density of a graph can be calculated as $\mu = n\pi R^2 / A$, where n is the number of hosts in the network graph, A is the square area and R is the radio transmission range. All the simulation results depicted in figures 6 to 11 are obtained by running the studied algorithms on 100 connected graphs and averaged over the given runs.

It is first assumed that each host has a fixed radio transmission range of 15, and the number of hosts ranges from 60 to 200 with increment step of 20. Figures 6 and 8 show the simulation results obtained by DLA-CC and DLA-DC in terms of the average size of cluster-head set, respectively. Figures 7 and 9 show the obtained results when the radio transmission range is then set to 30 and the number of hosts and the square area size remain unchanged in experiments. Then, the simulation experiments are repeated for the proposed clustering algorithm (DLA-DC) on the dense network graphs, when the number of hosts ranges from 200 to 1000 with increment step of 100. Figures 10 and 11 show the average size of the cluster-head set, when the radio transmission range is set to 15 and 30, respectively. In all simulations, the learning parameter is fixed at 0.2, and each algorithm is terminated when the probability of choosing the weakly connected dominating set (i.e., P in DLA-CC and *PCHS* in DLA-DC) is 0.90. A connected sparse graph with 100 hosts and its cluster-head set (or WCDS) constructed by the centralized proposed algorithm DLA-CC have been depicted in figures 5.A and 5.B, respectively. The radio transmission range is assumed to be 10 and the square area size 100×100. The cluster-heads have been colored red.

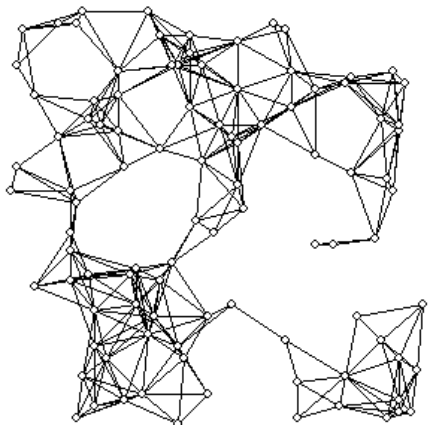

Figure 5.A. a connected sparse network graph with 100 hosts

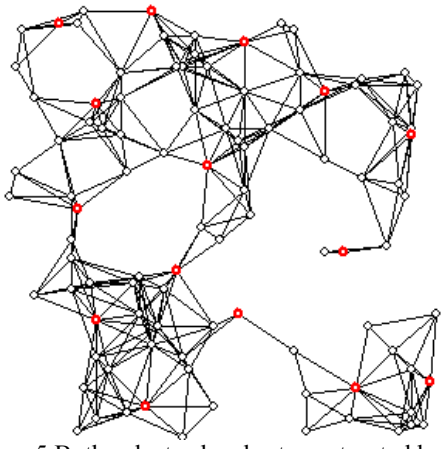

Figure 5.B. the cluster-head set constructed by the centralized proposed algorithm

As shown in all figures 6 to 11, the average size of the cluster-head set (WCDS) increases as the number of hosts (nodes) increases. Comparing the results depicted in figures 7, 9 and 11 with the figures 6, 8 and 10, we observe that the sizes of the cluster-head sets (WCDSs) constructed by all the studied algorithms become smaller when the radio transmission range for a host increases. The reason for this reduction is that a cluster-head with a larger radio transmission range can cover more hosts and so the whole network can be covered by a less number of the cluster-heads.

Comparing the results shown in figure 6 with figure 7 for the studied centralized algorithms, we find that Guha I always constructs the largest WCDSs, and so gives the worst results. The size of the WCDSs generated by Guha II are only slightly smaller than Guha I, but considerably larger than WCDS-CTR. The proposed centralized algorithm DLA-CC considerably outperforms (in terms of the size of the WCDS) the other well-known centralized algorithms regardless of the size and density of the network graph. As shown in figures 6 and 7, in general, as the number of nodes increases, the gap between the curves for our proposed algorithm and the curves for other centralized algorithms becomes significant.

As shown in figures 8 and 9, the proposed clustering algorithm DLA-DC always considerably outperforms the other distributed clustering algorithms in terms of the size of the cluster-head set. The size of the cluster-head sets constructed by WCDS-DC is nearly four times larger than that constructed by our distributed algorithm, and it is ranked lower DLA-DC. Max Degree performs better than Min ID, especially as the number of hosts increases, and AWF constructs the largest cluster-head sets. For example, the size of the cluster-head set constructed by DLA-DC is 6.76, when the number of hosts in network is 100 and the

radio transmission range is 15, while that of WCDS-DC is 21.9, which is 3.2 times larger than our algorithm. Comparing figure 8 with figure 9, we find that increasing the host's radio transmission range can increase the coverage area of each host, and so increase the density of the network, which leads to a smaller size of the cluster-head set. When the size of the square area grows, the hosts are distributed in a larger simulation area, and so it is expected that the number of cluster-heads to be increased.

The same simulation experiments are repeated for the proposed distributed clustering algorithm on the dense network graphs, where the number of hosts ranges from 200 to 1000 with increment step of 100, and the results are depicted in figures 10 and 11 as the radio transmission range is set to 15 and 30, respectively. Comparing the average size of the cluster-head set constructed by DLA-DC with that of the other distributed algorithms mentioned earlier, we observe that the proposed distributed algorithm outperforms the other studied algorithms, and the ranking given for the distributed algorithms, with the sparse network graphs, remains unchanged and DLA-DC is ranked above the other algorithms and AWF below.

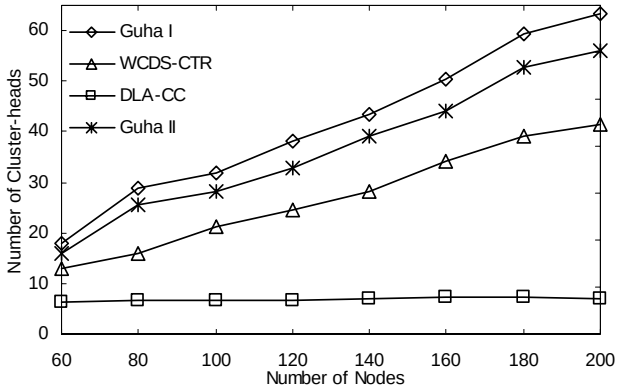

Figure 6. The average size of the WCDS constructed by the centralized algorithms, when the radio transmission range is 15 and the square area size is 100×100

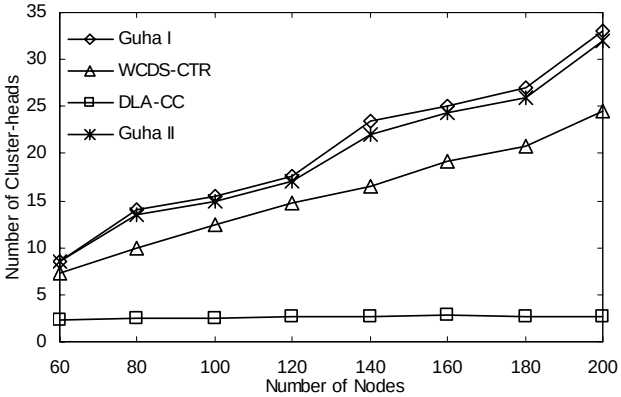

Figure 7. The average size of the WCDS constructed by the centralized algorithms, when the radio transmission range is 30 and the square area size is 100×100

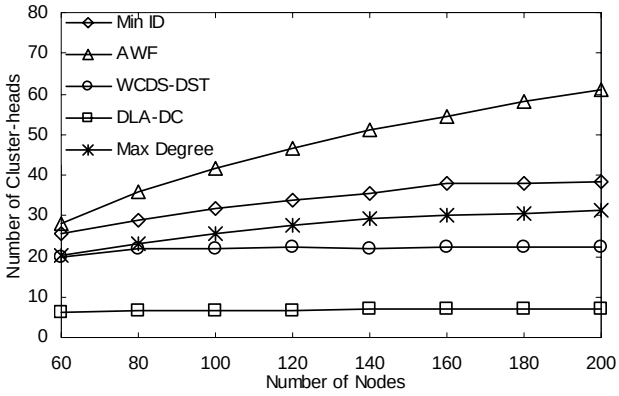

Figure 8. The average size of the cluster-head set constructed by the distributed clustering algorithms, when the radio transmission range is 15 and the square area size is 100×100

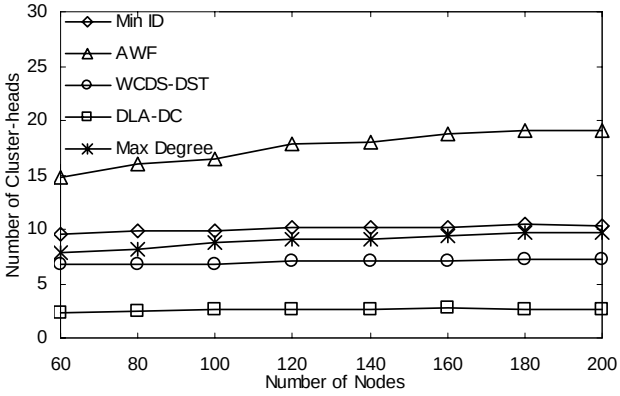

Figure 9. The average size of the cluster-head set constructed by the distributed clustering algorithms, when the radio transmission range is 30 and the square area size is 100×100

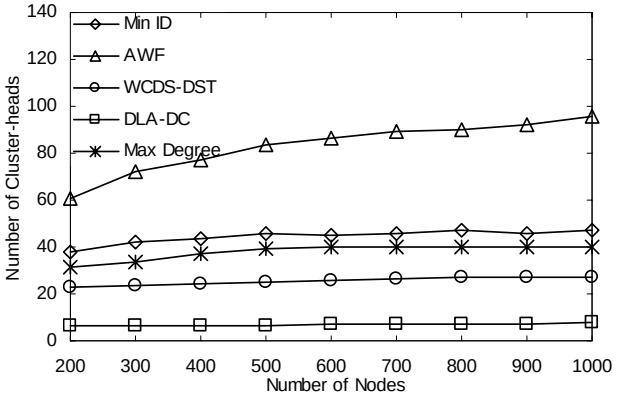

Figure 10. The average size of the cluster-head set constructed by the distributed clustering algorithms on the dense network graphs, when the radio transmission range is 15 and the square area size is 100×100

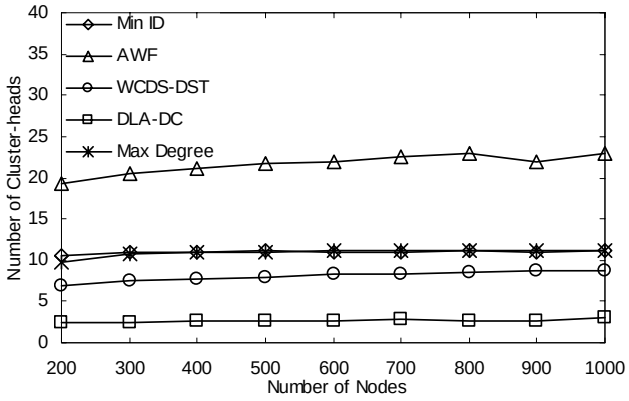

Figure 11. The average size of the cluster-head set constructed by the distributed clustering algorithms on the dense network graphs, when the radio transmission range is 30 and the square area size is 100×100

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. Since finding the weakly connected dominating set is a well-known approach for clustering the wireless ad hoc networks, in this paper, we first proposed a distributed learning automata-based algorithm for solving the minimum WCDS problem, and then proposed a clustering algorithm for wireless ad hoc networks in which a near optimal solution to the minimum WCDS problem is found. 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. We also computed the worst case running time and message complexity of the proposed clustering algorithm for finding a $1/(1-\varepsilon)$ optimal 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 that the proposed algorithms outperformed the best existing algorithms in terms of the size of the cluster-head set (or WCDS).

References

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

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

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

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

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

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

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

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

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

[10] R.G. Gallager, P.A. Humblet and P.M. Spira, "A Distributed Algorithm for Minimum Weight Spanning Trees," *ACM Transaction on Programming Languages and Systems,* Vol. 5, pp. 66-77, 1983.

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

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

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

[14] B. Das, and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," *IEEE International Conference on Communications (ICC'97)*, 1997.

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

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

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

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

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

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

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

[22] R. Ghosh, and S. Basagni, "Mitigating the Impact of Node Mobility on Ad Hoc Clustering," *Journal of Wireless Communications and Mobile Computing*, Vol. 8, pp. 295-308, 2008.

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

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

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

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

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

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

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

[30] H. Beigy and M. R. Meybodi, "An Adaptive Uniform Guard Channel Algorithm: A learning Automata Approach," *Lecture Notes in Intelligent Data Engineering and Automated Learning*, Springer Verlag, LANCES 2690, Germany, pp. 405-409, 2003.

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

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