A Backbone Formation Algorithm in Wireless Sensor Network Based on Pursuit Algorithm

YISONG JIANG, WEIREN SHI
College of Automation
Chongqing University
No. 174 Shazhengjie, Shapingba, Chongqing
China
jys12398@126.com, wrs@cqu.edu.cn

Abstract: -In wireless sensor network, virtual backbone formulation is a cost effective method to complete the broadcasting. Minimum connected dominating set is an outstanding candidate of virtual backbone. However, it is NP-Hard to find a minimum connected dominating set in an arbitrary graph. In this paper, we propose a novel backbone formation algorithm to construct a connected dominating set. In the proposed method, a dominating set and action sets are got at first. Then, sink executes the pursuit algorithm, in which nodes are treated as learning automata and action probability vector is changed with time, and chooses actions from their action sets to construct a connected dominated set Finally, the automaton converges to a common policy and an approximate solution of the minimum connected dominating set is got. It is also shown that our method is ε-optimality with the changing speed of learning parameter. The simulation results show that our algorithm has a good performance in terms on the size of backbone, the message overhead and average node degree.

Key-Words: -Backbone; Wireless sensor network; connected dominating set; pursuit algorithm

1 Introduction

A wireless sensor network (WSN) is composed by many sensor nodes and one or multiple sinks. In WSN, sensors gather data from the sensing environment and transfer those data to the sink nodes or base stations. Generally, WSNs are deployed in some emergent or temporary situation, for example accidents or environment gatherings. Broadcasting is a fundamental means in WSNs. However, the way of packet transmission in WSN is different from the way in the wired network. Generally, there are many redundant retransmissions in broadcasting because the omnidirectional radio propagation and a physical location may be covered by the transmission ranges of several nodes. Virtual backbones are usually used to support broadcasting and multicast in WSNs and a minimum connected dominating set (MCDS) is an outstanding candidate to work as a virtual backbone.

In this paper, a graph G(V,E) is used to represent a WSN. In G, the vertexes set V represent the nodes in the WSN, and E represents all the links in the network. We also assume that all nodes have the same transmission range. Therefore, the graph G is also a unit disk graph. A dominating set (DS) of graph G is a subset of V such that each vertex which is not in set will be joined to at least one member of the set by an edge. A connected dominating set (CDS) is a DS that induces a connected subgraph of G. A minimum connected dominating set (MCDS) is a connected dominating set with the smallest cardinality among all possible connected dominating sets of G. However, it is NP-hard to find a MCDS of a graph [1].

We propose a quasi-distributed algorithm to construct a CDS of a WSN. The proposed method consists of two steps. At the first step, a DS and the action sets of the nodes in the DS are got. At the second step, according to the cross entropy method, the sink get a CDS based on those action sets. In detail, the DS is got by a color process. Then, each node in the DS (called dominator) gets its action set based on the connection path between itself and its 2 or 3 hops dominator neighbors. When the sink receives the action sets from dominators, it treads those dominators as automata and executes the revised pursuit algorithm to form a CDS. In the pursuit algorithm, each automaton chooses an action from its action set to construct a CDS and updates the estimate vector and actions probability vector based on the size of the feedback. Finally, the automaton converges to a common policy that constructs an approximate solution of the MCDS of the network.

The organization of the rest of this paper is shown as follows. Section 2 shows the related works and Section 3 briefly presents the learning automata and the pursuit algorithm. The purposed CDS formation algorithm is presented in Section 4 and the ε-optimality of the purposed algorithm is proven in Section 5. In addition, the performance of the proposed algorithm is shown in Section 6. The conclusion is presented in Section 7.

2 Related works

Most of MCDS approximation algorithms construct a CDS based on a maximal independent set (MIS). A MIS is a set that every edge in G has at least one endpoint not in the set and every vertex not in the set has at least one neighbor in it. In [2], Alzoubi et al. proposed two MIS-based algorithms. The first algorithm requires a spanning tree to complete the process in which a CDS is generated. The second algorithm does not need the spanning tree and enables the maintenance of the weakly-connected dominating set to be simpler. Bo [3] presents a zone-based distributed algorithm, in which every node is assigned “Zone” and “Level” marks to indicate its subtree and the distance to root of the subtree. Li et al. [4] put forward the S-MIS algorithm, which is a greedy algorithm and constructs a CDS with the help of Steiner tree. At the same time, a distributed version of this algorithm is also introduced. Gao et al. [5] purpose another MIS-based distributed algorithm that has a better message complexity compared with Alzoubi’s algorithm [2] and is stable and scalable in large and dense network.

In [6], Wu and Li purpose a prune-based algorithm, in which prunes some redundant nodes from the original CDS that got by “Rule 1” and “Rule 2”. Another enhanced prune method, called Rule k, is introduced by Dai and Wu in [7]. In Rule k, a node will be removed from CDS if its neighborhoods are covered by a set of k neighbors with higher IDs and the node set is strongly connected.

Akbari Torkestani and Meybodi [8] described an intelligent CDS-based backbone formation algorithm in which the learning automata are used to construct the CDS of the network. The automata will be rewarded, if it is the smallest one that has been constructed so far. Otherwise, it will be penalized. However, the learning automata apply a direct algorithm that only uses the environmental feedback of the current stage to update the probability vector. Akbari also purposes another backbone formation algorithm to the energy efficient in WSNs. In this algorithm [9], a learning automata-based heuristic is purposed for finding an optimal solution of the proxy equivalent constrained CDS problem. The degree and the backbone delay also considered in this algorithm to prolong the backbone duration and to shorten the delay of transmissions. In [10], a load-balanced virtual backbone construction algorithm is proposed by He et al. they consider the size and the load-balance factors when constructing the backbone of WSN. After a backbone is got, they propose an approximation algorithm by using the linear relaxing and random rounding technique to allocate non-backbone nodes to proper backbone nodes with an objective to minimize the maximum valid degree of all the backbone nodes [10].

3 Learning automata and pursuit algorithm

A learning automaton consists of an adaptive learning agent operating in unknown environment [11]. It takes an action from its action set to maximize the probability of being rewarded from the environment. And the learning automaton tries to find optimal action through iterations. In each iteration, it gets a feedback from the environment and updates its action probability vector. An automaton can be represented as a 4-tuple (**A, P, F, T**) and environment can be represented as a 3-tuple (**A, F, D**).

Where, **A**=$\{\alpha_1,\alpha_2,\cdots,\alpha_i,\cdots,\alpha_r\}$,$1\le i\le r$,is the set of actions and*r*is the number of actions.

T is a scheme that used to update the action probability vector as follows,

$$\mathbf{P}(t+1)=\mathbf{T}(\mathbf{P}(t),\alpha(t),\beta(t)) . \quad (1)$$

P is theaction probability vector of each action. $\mathbf{P}(t)=[p_1(t),p_2(t),\cdots,p_r(t)]$ is the probability vector at iteration t. In each iteration, the automaton selects an action form A with respective probabilities in P(t).

Fis the set of the feedback from environment. The feedback at the iteration t is denoted asβ(t)(β(t)∈**F**). In this paper, β(t)=1 if the action get a reward form the environment. Otherwise, β(t)=0.

$\mathbf{D}=\{d_1(t),\cdots,d_i(t),\cdots d_r(t)\},1\le i\le r$, is the set of average reward value, where $d_i(k)=E[\beta(t)\,|\,\alpha(t)=\alpha_i]$ stands for the expectation of reward value when an automaton chooses the i-*th* action. If $d_i(t)$ is independent of t for all $1\le i\le r$, the environment will be stationary. Otherwise,it will be non-stationary. In this paper, the environment is stationary, the expression of **D** can be simplified to $\mathbf{D}=\{d_1,\cdots,d_i,\cdots d_r\}$,m is used to denote the index of the optimal action, $d_m=\max_i\{\mathrm{d}_i\}$.

It is important to choose a scheme of **T** to get better performances, i.e. a better result or fast speed of convergence. Thathachar and Sastry [12] introduced the estimator algorithms where the automaton runs an estimator to provide guidance for updating the action probability vector. One of the estimator algorithms presented by Thathachar and Sastry in [13] is called the pursuit algorithm (PA) or continuous pursuit reward-penalty (CP_{RP}) algorithm [14].

Table 1

Algorithm PA

Initialize t=1, $p_i(1)=1/r$ for all$1\le i\le r$.

Initialize the estimate vector $\hat{\mathbf{d}}(t)$ by taking each action a small number of times.

Repeat

(1) At the iteration t, the automaton chooses an action α_i in line with its action probability vector $\mathbf{P}(t)$. Let $\alpha(t)=\alpha_i$.

(2) After getting the feedback $\beta(t)$, update the estimate vector $\hat{\mathbf{d}}(t)$ according to the following equations,

$$\begin{aligned}&W_i(t+1)=W_i(t)+\beta(t)\\&N_i(t+1)=N_i(t)+1\\&\hat{d}_i(t+1)=\frac{W_i(t+1)}{N_i(t+1)}\end{aligned} \quad . \quad (2)$$

(3) Update the action probability vector $\mathbf{P}(t)$ on the basis of the following equation

$$\mathbf{P}(t+1)=(1-\lambda)\mathbf{P}(t)+\lambda\mathbf{e}_{M(t)}. \quad (3)$$

End repeat

Where $\hat{\mathbf{d}}(t)=[\hat{d}_1(t),\cdots,\hat{d}_r(t)]$ is the estimate vector and $\hat{d}_i(t)$ refers to the average reward value of i-th action at the iterationt.

$\mathbf{W}(t)=[W_1(t),\cdots,W_r(t)]$ is a vector and $W_i(t)$ is the number of times that action α_i has been rewarded up to the time t.

$\mathbf{N}(t)=[N_1(t),\cdots,N_r(t)]$ is a vector and $N_i(t)$ is the number of times that the action α_i has been chosen up to the time t.

λis the speed of a learning parameter and satisfies the condition 0<λ<1.

$e_{M(t)}$ is a r-vector with 1 in the M(t)-th coordinate, and the others are 0. $M(t)$ is theindex of the maximal component of $\hat{\mathbf{d}}(t)$, and $\hat{d}_{M(t)}(t)=\max_i\{\hat{d}_i(t)\}$.

Note that PA stops when the condition is satisfied. For instance, once one of the action probabilities is larger than 0.9 or other values, the repeat will be stopped. It is easy to observe that the automaton always pursues the 'current' action, i.e., optimal action, and hence the name pursuit algorithm [15].

4 Backbone formation algorithm based on the pursuit learning algorithm

Our algorithm is clustering-based. In this algorithm, a MIS is generated at first. Next,every node in the MIS finds its action set. Then, those action sets are sent to the sink and the PA algorithm will be executed to get a CDS with small size. Finally, the topology of the CDS will send to each node in the MIS.

4.1 Dominator selection process

Before we get the dominator selection process, each node is marked with a unique rank, i.e., (degree, ID), where the degree refers to the number of neighbors and ID is the label of a node. We assume that rank(n_1)is higher than rank(n_2)if the degree of n_1 is larger than the degree of n_2 or the ID of n_1 is larger than the ID of n_2 when the degree of n_1 is equal to the degree of n_2.

The way to select the dominator is given in the following content.

(1) Initial all nodes with white color.

(2) If a white node has a higher rank than all its white neighbors, it will be marked black and all of its neighbors will be marked gray.

When a node is marked black, it broadcasts a BLACK message to its neighbors. As soon as a node receives a BLACK message, it changes its color to gray and records the ID of the black node. Similarly, once a node is marked gray, it will broadcast GRAY message to its neighbors to mentionthe change of its color. The white node compares its rank with all of its white neighbors whenit receives a GRAY message. If the condition (2) is satisfied, it will be marked black.After the color process, there are no white nodes in the network;all nodes will bemarked black or gray, and, the black nodes form a MIS [16]. Therefore, each MIS is also a DS of the graph. The black node is also called dominator, while the gray node is called dominatee.

4.2 The formation of action sets

Every node in the MIS is treated as a learning automaton. To getting a CDS, each dominator selects some gray nodes as connectors to build connection paths to other dominators, which are three hops or two hops away. In this paper, if the connection paths between those dominators are replaced with edges, to getting a CDS with a small size, there is no cycle in the CDS. Thus, the topology of the CDS is a tree. Each dominator except the root of the tree will choose another dominator as its father node. An action of a dominator is its choice (the choice of the root is null). The action set consists of two parts. The first parts is those dominators that are three hops or two hops away (we call those nodes the dominators need to be connected (DNC)). The second part is the connection paths, which includes the connector. For example, as shown in Table2, d1 d2 and d3 are the DNC of dominator d. The connection paths form d to d1, d2 and d3 are (d, c1, d1), (d, c2, c3, d2) and (d, c4, c5, d3), respectively. Therefore, the connectors of those paths are c1, (c2, c3), (c4, c5), respectively.

Table2. Anillustration of the action set of dominator d.

DNC	Connectors
d1	c1
d2	(c2,c3)
d3	(c4,c5)
…	…

Generally, the formation of action setscan be divided into two stages.At the first, each dominatee broadcast a One-Hop-Dominator message in which the IDs of its neighbor dominator are added. Thus, each node will get the information of its two hops dominator neighbors. Then, every dominator adds the actions of those dominators, which are twohops away,to its action set. We use a triple (a1, a2, a3) to represent a path fromdominator a1 to another dominator a3 that is two hops away. If there are severalpaths between a1 and a3, the one with the maximum rank will be chose and added to the action set. For example, there are two routes between node 1 and node 8 in Fig.1, so the connector of the pathcould be node 4 or node 5. According to the principle we have mentioned before, node 4 will be chose because it has a larger rank than node 5. This principle can decrease the size of the final CDS because the chance that the connectorwith a larger rank is chose by other dominators is larger, for example, node 4 also is chose as a connector by node 10.

At the second stage, every dominatee broadcast a Two-Hop-Dominator that includes its two hop dominator and the connector that connect the dominatee and the dominator. After this process, each dominator will get the information of the three hops away dominators and find connections path to its three hops dominator neighbors. A four-tuple (b1, b2, b3, b4) is used to signify a path from dominator b1 to another dominator b4, which is3 hops away from b1. If there are multiple routes between them, the route that has a greater amount of degree (b2) and degree (b3)will be chose. For example, there are three routes from node 1 to node 11 in Fig.1, i.e., (1, 2, 6, 11), (1, 3, 6, 11) and (1, 3, 7, 11). Based on the principle, the route (1, 3, 7, 11) will be chose.

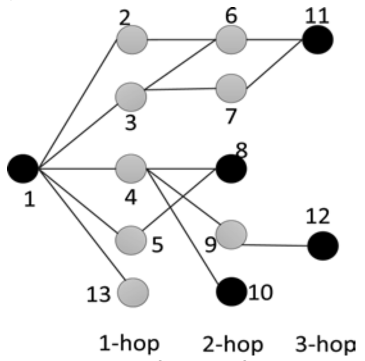

Fig.1. An illustration for the formation of action sets.

4.3 Getting the CDS based on the pursuit algorithm

After the formation of action sets, every dominator sends its action set to the sink (so, our algorithm is quasi-distributed or quasi-global). Based on the action sets of dominators, sink finds nodes as little as possible to connect the dominators and get a small CDS. This is a Steiner tree problem. However, this version of the Steiner tree problem is NP-complete. The algorithm PA, in which the dominators are treated as automata, is used to get an approximate solution. More details about the algorithm and its parameters are described as follows.

To getting the feedback of the environment, the following principle is utilized. If the size of the CDS of current iteration t (we denote the size as V_t) is not greater than the best value (BV) of the sizes of CDSs that have been got so far, the value of the feedback β(t) will be set to 1 (as shown in formula (4)) and the values in the array also be updated.β(t)=1 means the actions adopted by the automata are rewarded by the environment.

$$\beta(t) = I\{V\_t \le \mathrm{BV}\}\,. \quad (4)$$

Where $I\{V\_t \le BV\} = \begin{cases} 1 & V\_t \le \mathrm{BV}) \\ 0 & otherwise \end{cases}$.

The speed of the learning parameter λcontrols the steps that can be taken fromP(t) toP(t+1). In general, for a fixed λ, a large valuewill lead to a higher rate of convergence; and vice versa. We follow [17] to take the situation thatλ changes with time (as shown in formula (5)) because they argue that a changing λ is consistent with the notion of convergence.

$$\lambda(t) = 1 - \theta^{1/t}\,, \quad (5)$$

whereθ∈(0,1).

According to value of β(t) and λ(t), each dominator updates the values of parameters as shown in formula (2). Next, it updates the value of M(t) and the action probability vector P(t) according to formula (6) given below. However, in the process of initialization,the action probability vector P(t) keeps consistent.

$$\mathbf{P}(t+1) = \left(1-\lambda(t)\right)\mathbf{P}(t) + \lambda(t)\mathbf{e}_{M(t)} \quad (6)$$

We assume that the stop condition meets if one of the following conditions is satisfied. The first condition is that the number of iterations is equal to a threshold. The second one is that the values of the sizes of CDSs, which are obtained in the last **K** iterations, are the same (**K** is a positive integer).When the pursuit algorithm finishes, the CDS with the smallest size will be returned as the output of this algorithm.

As mentioned in section 4.2, the topology of the CDS is a tree if we replace the connection paths between dominators with edges. On the other hand, each dominator stores the connection paths in the action. So, the sink just needs to send each dominator a message, which contains the dominator neighbors of the dominator in the tree, to make dominators establish necessary connections that included in the CDS.

5 The convergence of our algorithm

The convergence of a learning algorithm implies that the automaton will always implement the optimal action eventually. Such a kind of convergence is typically called ε-optimality. Rajaraman and Sastry [15] put forward a framework to analyse the finite time behaviour of the pursuit learning algorithm. However, the speed of learning parameters is a fixed value in their analysis. In this paper, following the framework, we analyse the convergence of the pursuit learning algorithm when λ changes with time, as shown in formula (5).

Before we start to analyse ε-optimality of the algorithm, there are two lemmas that can contribute to the analysis of the convergence.

Lemma 1: For any given positive constant δ>0 and a positive integer 0<n<∞, there is $0 < T_1 < \infty$ such that

$$\Pr\{\min_{i=1,\dots,r} Z_i(t) \ge n\} > 1-\delta \quad \forall t > T_1\,. \quad (7)$$

Proof: Proving (7) is equal to: for all i∈{1, 2,…, r}

$$\Pr\{Z_i(t) < n\} < \delta\,. \quad (8)$$

That is,

$$\sum_{j=0}^{n-1} \Pr\{Z_i(t) = \mathrm{j}\} < \delta\,.$$

If for all j, 0≤j≤n-1, the inequality will follow

$$\Pr\{Z_i(t) = \mathrm{j}\} < \frac{\delta}{n}\,. \quad (9)$$

At any iteration k, $\Pr\{\alpha(\mathrm{k}) = \alpha_i\} \ge \prod_{t=1}^{k}\left(1-\lambda(\mathrm{t})\right) p_i(0)$ is always tenable because the action probability decreases by (1−λ(t)) at most at each iteration.

Thus,

$$\begin{aligned}\Pr\{\alpha(\mathrm{k}) \ne \alpha_i\} &\le 1 - \left(\prod_{t=1}^{k}\left(1-\lambda(\mathrm{t})\right) p_i(0)\right) \\ &= 1 - \left(\prod_{t=1}^{k}\theta^{1/t}\ p_i(0)\right)\end{aligned},$$

where, θ∈(0,1) and $p_i(0) = 1/\mathrm{r}$. We can get

$$\begin{aligned}\Pr\{\alpha(\mathrm{k}) \ne \alpha_i\} &< 1 - \left(\prod_{t=1}^{k}\theta^{1/\mathrm{k}}\ \frac{1}{r}\right). \\ &= 1-\theta/r\end{aligned}$$

Based on the foregoing inequality, we can get the following inequality.

$$\begin{aligned}\Pr\{Z_i(t) = \mathrm{j}\} &< C_k^j \times 1^j \times \left(1-\theta/\mathrm{r}\right)^{k-j} \\ &< k^j \times \left(1-\theta/\mathrm{r}\right)^{k-j} = k^j\sigma^{k-j}\end{aligned},$$

whereσ=θ/r.

Define a function $\Psi(x) = x^n\sigma^{x-n}$ (x>0). If we prove that there is a value T_1 that makes $\Psi(x) < \delta/n$ be always satisfied when $x > T_1$, the inequality (9) will be proved and Lemma 1 will be proved as well. Where,

$$\Psi'(x) = d\Psi(x)/dx = x^{n-1}\sigma^{x-n}(x \cdot \ln\sigma + n).$$

It is easy to get $\Psi'(x) < 0$ when $x > n/\ln(1/\sigma) = T_2$.

Consequently, $\Psi(x)$ is a decreasing function when $x>T_2$. If $\Psi(T_2) \le \delta/n$, we can take $T_1 = \lceil T_2 \rceil$ to make Formula (9) and Lemma 1 be followed all the time. $\lceil x \rceil$ stands for the smallest integer that is equal to or larger than x. When $\Psi(T_2) > \delta/n$, if we want to prove Lemma 1, it will be essential for us to prove that there is another value T_3 that is larger than T_2 and $\Psi(T_3) = \delta/n$. We can get

$$\lim_{x\to+\infty} \Psi(x) = \lim_{x\to+\infty} x^n\sigma^{x-n} = \lim_{x\to+\infty} \frac{x^n}{(1/\sigma)^{x-n}}.$$

By virtue of the L' Hôpital's rule and $1/\sigma>1$, we can get

$$\lim_{x\to+\infty} \Psi(x) = \lim_{x\to+\infty} \frac{x^n}{(1/\sigma)^{x-n}} = \lim_{x\to+\infty} \frac{n!}{(1/\sigma)^{x-n}[\ln(1/\sigma)]^n} = 0.$$

Because $\Psi(x)$ is a decreasing and continuous function when $x>T_2$, there must be a value T_3 that is larger than T_2 and $\Psi(T_3) = \delta/n$ is satisfied. Therefore, Formula (9) and Lemma 1 always follow when $T_1 = \lceil T_3 \rceil$. ■

Lemma2: For any given positive constant $\delta>0$, $\kappa>0$ and $i \in \{1, 2,\dots,r\}$, there is $0< T^* <\infty$ so that

$$\Pr\left\{\left|\hat{d}_i(t) - d_i\right| \le \kappa\right\} > 1-\delta \quad \forall t > T^*.$$

Proof: According to the definition

$$\hat{d}_i(t) = \frac{W_i(t)}{N_i(t)} = \frac{I\{\alpha(\mathrm{t}) = \alpha_i\}\beta(\mathrm{t})}{N_i(t)}.$$

We can find that $\hat{d}_i(t) \in [0,1]$ acts as the estimate of d_i. For any iteration t, it is possible to get the following inequality by applying the Theorem 2 of Hoeffding [18].

$$\Pr\left\{\left|\hat{d}_i(t) - d_i\right| > \kappa\right\} < 2\exp(-2N_i(t)\kappa^2). \quad (10)$$

In accordance with the laws of total probability, we can obtain

$$\begin{aligned}&\Pr\left\{\left|\hat{d}_i(t) - d_i\right| > \kappa\right\} \\ &= \Pr\left\{\left|\hat{d}_i(t) - d_i\right| > \kappa \mid N_i(t) \ge n\right\}\Pr\{N_i(t) \ge n\} \\ &+ \Pr\left\{\left|\hat{d}_i(t) - d_i\right| > \kappa \mid N_i(t) < n\right\}\Pr\{N_i(t) < n\} \\ &< \Pr\left\{\left|\hat{d}_i(t) - d_i\right| > \kappa \mid N_i(t) \ge n\right\} + \Pr\{N_i(t) < n\}\end{aligned}.$$

(11).

According to Inequality (10),

$$\Pr\left\{\left|\hat{d}_i(t) - d_i\right| > \kappa \mid N_i(t) \ge n\right\} < 2\exp(-2\mathrm{n}\kappa^2).$$

Set $n = \left\lceil \frac{1}{2\kappa^2}\ln\frac{4}{\delta} \right\rceil$, then

$$\Pr\left\{\left|\hat{d}_i(t) - d_i\right| > \kappa \mid N_i(t) \ge n\right\} < \frac{\delta}{2}. \quad (12)$$

According to Lemma 1,

$$\Pr\{N_i(t) < n\} < \frac{\delta}{2}. \quad (13)$$

The above inequality always follows when $t>T^*$. Here, we will obtain $T^* = \lceil T_2 \rceil$ if $\Psi(T_2) \le \delta/(2n)$. Otherwise, $T^* = \lceil T_4 \rceil$, where the value of T_4 is larger than that of T_2 and $\Psi(T_4) = \delta/(2n)$.

Combine (11)-(13), t the following inequality can be obtained:

$$\Pr\left\{\left|\hat{d}_i(t) - d_i\right| > \kappa\right\} < \delta \quad \forall t > T^*. \quad (14)$$

That is,

$$\Pr\left\{\left|\hat{d}_i(t) - d_i\right| \le \kappa\right\} > 1-\delta \quad \forall t > T^*. \quad (15)$$

Hence, Lemma 2 is proved. ■

Theorem 1: For any given positive constant $\delta \in (0, 1)$ and $\varepsilon \in (0, 1)$, there is $0<T'<\infty$ such that

$$\Pr\{p_m(t) > 1-\varepsilon\} > 1-\delta \quad \forall t > T'.$$

Proof: According to Lemma 2 and by taking $\kappa = \frac{1}{2}(d_m - d_j)$, $\forall j \ne m$, we can get the probability that $\left|\hat{d}_m(t) - d_m\right| < \kappa$ and $\left|\hat{d}_j(t) - d_j\right| < \kappa$ is larger than $1-\delta$ when $t>T^*$.

Thus,

$$\Pr\left\{\left|\hat{d}_m(t) - d_m\right| + \left|\hat{d}_j(t) - d_j\right| < 2\kappa\right\} > 1-\delta.$$

It is easy to get

$$\begin{aligned}\left|\hat{d}_m(t) - d_m\right| + \left|\hat{d}_j(t) - d_j\right| &> d_m - \hat{d}_m(t) + \hat{d}_j(t) - d_j \\ &= 2\kappa - \hat{d}_m(t) + \hat{d}_j(t)\end{aligned}.$$

Based on the inequalities, for all $\forall j \ne m$ and $t>T^*$, we can get the following inequality.

$$\Pr\left\{\hat{d}_m(t) > \hat{d}_j(t)\right\} > 1-\delta. \quad (16)$$

According to the law of total probability,

$$\Pr\{p_m(t) > 1-\varepsilon\} > \Pr\left\{p_m(t) > 1-\varepsilon \mid \hat{d}_m(t) > \hat{d}_j(t)\right\} \Pr\left\{\hat{d}_m(t) > \hat{d}_j(t)\right\} .(17)$$

Theorem 1 will be proved if

$$\Pr\left\{p_m(t) > 1-\varepsilon \mid \hat{d}_m(t) > \hat{d}_j(t)\right\} = 1,$$

for all t>$T' \geq T^*$.

Assume $T' = t_1 + T^*$, where t_1>0.

$$p_m(t) = 1 - \sum_{j \neq m} p_j(t) .$$

Therefore, we can prove $\sum_{j \neq m} p_j(t) < \varepsilon$ when $\hat{d}_m(t) > \hat{d}_j(t)$ and t>T'.

$$\sum_{j \neq m} p_j(\mathrm{T}') = \sum_{j \neq m} \left\{ p_j(\mathrm{T}^*) \prod_{q=1}^{t_1} [1-\lambda(\mathrm{T}^*+q)] \right\}$$
$$< \prod_{q=1}^{t_1} [1-\lambda(\mathrm{T}^*+q)] = \prod_{q=1}^{t_1} \theta^{1/(\mathrm{T}^*+q)} .$$
$$< \theta^{t_1/(\mathrm{T}^*+t_1)}$$

If $\theta^{t_1/(\mathrm{T}^*+t_1)} < \varepsilon$, the following inequality should be satisfied.

$$(\ln\theta - \ln\varepsilon)\mathrm{t}_1 < \mathrm{T}^* \ln\varepsilon .$$

Take θ<ε, $\mathrm{t}_1 > \mathrm{T}^* \dfrac{\ln\varepsilon}{\ln\theta - \ln\varepsilon}$, $\mathrm{t}_1 = \left\lceil \mathrm{T}^* \dfrac{\ln\varepsilon}{\ln\theta - \ln\varepsilon} \right\rceil$ and $T' = t_1 + T^*$.

According to (17), we could get

$$\Pr\{p_m(t) > 1-\varepsilon\} > 1-\delta ,$$

for all t>T'.■

Based on Theorem 1, it can be concluded that the choice probability of the expected action converges at an approximate solution to the MCDS for a larger iteration.

6 Experiment results

To investigate the performance of our algorithm, several experiments are shown in this section (the simulator is written in M language of Matlab 2010b). We assume that an ideal MAC layer is used in those simulations. The packet can be transmitted without contention, packet losses or collision. In those simulations, nodes are uniformly distributed in a square area of 100 units by 100 units. The number of nodes in the network ranges from 100 to 500 (the number of nodes is presented by SN), and more than 50 connected graphs are random generated for each given number of node (the simulation result are the average values of those graphs). In the simulation of BFA-PA, the threshold of iterations is set to 3000 and the value of K is set to 20. The value of SN, the radio transmission range of nodes (presented by Tr), and the speed of learning parameters (presented by λ) are the parameters that influence the backbone of the network.

We compare the performance of BFA-PA with Zone-based virtual backbone formation (ZVBF-MD) [3] and Torkestani's DAL-BF algorithm [8] in terms of the size of CDS, message overhead and the average node degree in the CDS. On the other hand, Butenko's algorithm [19] is used to give a reference to the best solution of CDS. In our simulation, the dominator in ZVBF-MD uses the quasi-global version to get a smaller CDS [3]. In the simulation, the learning rate [8] of DAL-BF is 0.2, the PCDS [8] of DAL-BF is 0.9 and the maximum iterations of DAL-BF is 200.

Accordance to the definition of λ, we change the value of θ to get the influences of λ. In the following experiment, the value of θ is set as 0.1, 0.2, 0.3, 0.4, 0.5 and 0.6, respectively. At the same time, SN is set as 300 and the transmission range is varied from 15 to 30 units. The size of CDS and the number of iterations are the metrics. The simulation results are shown in Fig. 2 and Fig. 3. Based on Fig. 2, we get that the size of CDS decreases slightly when θ increases. Fig. 3 shows that, in general, the number of iterations increase with the value of θ. This can be attached to the fact that the smaller value of θ make the algorithm has a larger value of λ, and led it converges quickly. On the other hand, rapid convergence leads to greater results about the size of CDS and the smaller result about the number of iterations. In the following simulations, we set the value of θ to 0.4 to make a trade-off between the convergence rate and the size of the CDS.

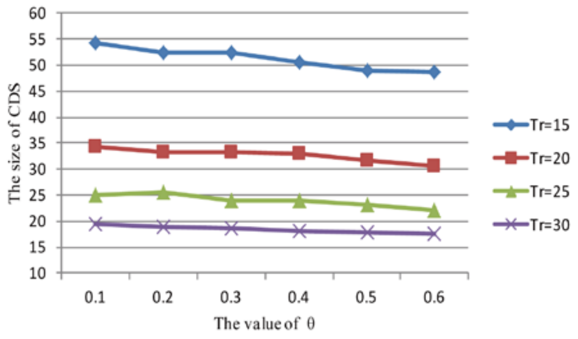

Fig.2.The size of CDS when the value of θ is changed.

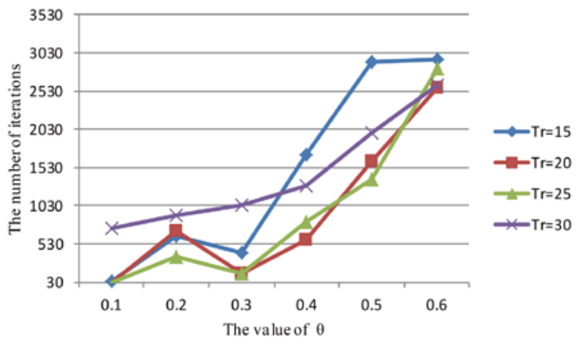

Fig.3.The number of iterationswhen the value of θ is changed.

In the next simulation, we get the results when SN and Tr are changed. The transmission range Trchanges from 15 to 50 units andthe size of the network SN varies from 100 to 500. In Fig.4, the x-axis denotes the value of SN and different curves have different values of Tr. According to the results shown in Fig. 4, we can get that the size of CDS increases with SN when Tr is small and the ascendant trend is not significantwhen Tr is large, i.e., Tr = 35. When SN is fixed, the size of CDS decreases with the increase of Tr becausea dominator node can cover more nodes when Tr increase. However, the decline trend of the size of CDS is slight when Tris greater than 45.

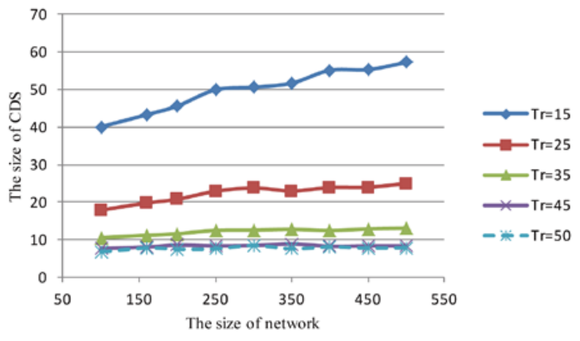

Fig.4. The number of CDSs when SN and Tris changed.

In the following experiments, we compare our algorithm BFA-PA, DAL-BF algorithm [8], Butenko’s algorithm and ZVBF-MD [3] in terms of the size of CDS, message overhead and average node degree in the CDS.

Fig.5 and Fig.6show the resultswhen transmission range is 15 and 30 units, respectively, and SN is changed from 100 to 500. Those results show that the size of CDS increases when SN increases. However,when SN is a large number, the increase rate slows down because it makes the network denser and a dominator covers more nodes. Furthermore, it can be found that BFA-PA constructs the smaller CDS compared with ZVBF-MD and DAL-BF when the network is dense (i.e., when SN is more than 300). We also make a comparison among those algorithms when the size of network is fixed and Tr is changed. Related results are shown in Fig.7 and Fig. 8, in which SN is set as 200 and Tr is changed from 15 to 50 with a step of 5 units. The size of CDS decreases whenTr increases because the increase of Tr results in higher density of the network. Consequently, a dominator can cover more nodes so that fewer dominators are needed. When Tr is a large number, the performance of ZVBF-MD, BFA-PA and DAL-BF are close.

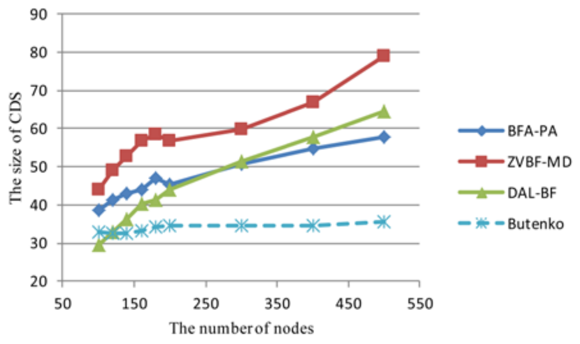

Fig.5. The size of CDS when SN is changed and Tr equals 15 units.

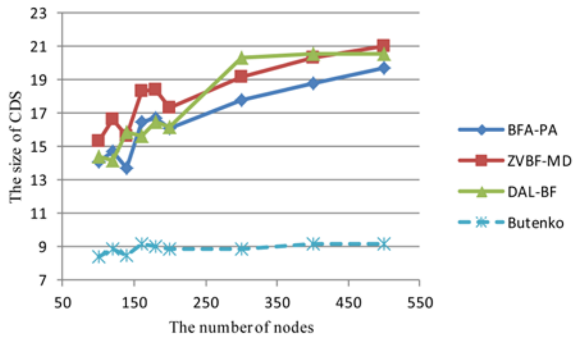

Fig.6. The size of CDS when SN is changed and Tr equals 30 units.

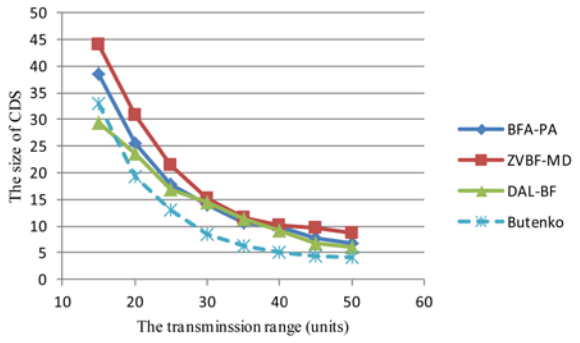

Fig.7.The size of CDS when Tr is changed and SN equals 100.

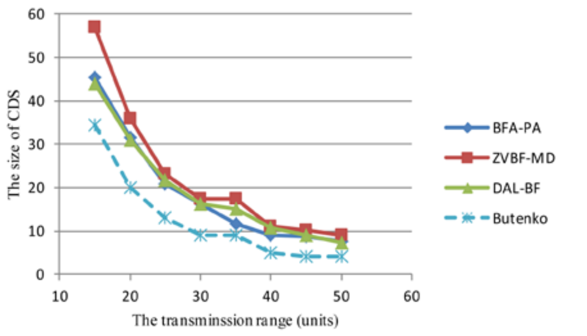

Fig.8. The size of CDS when Tr is changed and SN equals 200.

Message overhead is another metric of the CDS formation algorithm. In the wireless sensor network, the hosts suffer from strict resource limitations so that the communication overhead should be kept as low as possible. Fig.9 and Fig. 10 show the message overhead of different algorithms when the size of the network is changedand Tr is set to15 and 30, respectively.The message overhead of DAL-BF is not shown in the figures because its message overhead is large. For example, when SN is 100 and Tr is 15 units, the message overhead DAL-BF is more than 400 thousands bytes. Based on those results, we can find that BFA-PA has alarge message overhead than ZVBF-MD. This is cause by the fact that, in BFA-PA, each dominator will get its action set and send the set to the sink. On the other hand, in ZVBF-MD, only the dominator at the zone border executes a similar process. We also could find that the number of message overhead increases when the number of nodes or Tr increases.

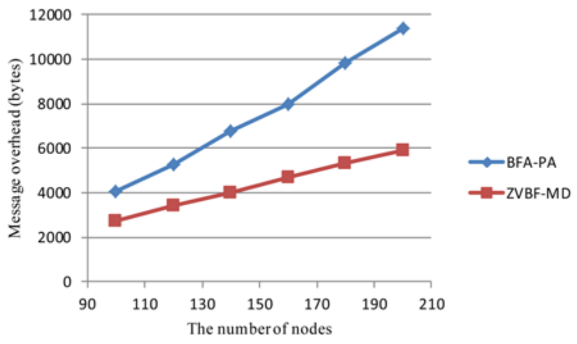

Fig.9. The Average messageoverhead when SN is changed and Tr equals15 units.

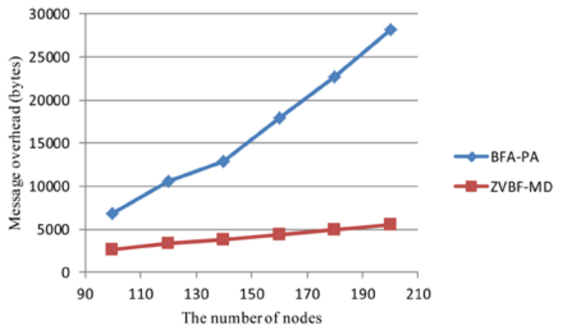

Fig.10. The Average messageoverhead when SN is changed and Tr equals 30 units.

The average node degree in the CDS is another parameter proposed by Bo [3], which is also the average node degree in the induced sub graph of the constructed CDS. Bo believes that a low degree may cause less interference for communication. Both Fig. 11 and Fig. 12 show the experiment results when the value of SN changes from 100 to 500, and Tris set as 15 and 30 units, respectively. Based on those results, we can get that the node degree in the CDS increases when the size of network increases and Trincreases. When the network is dense (T*r* is set as 30 units and SN is more than 300), the rising trends of those algorithms are not conspicuous any more, and BVF-PA has similar performance to ZVBF-MD.

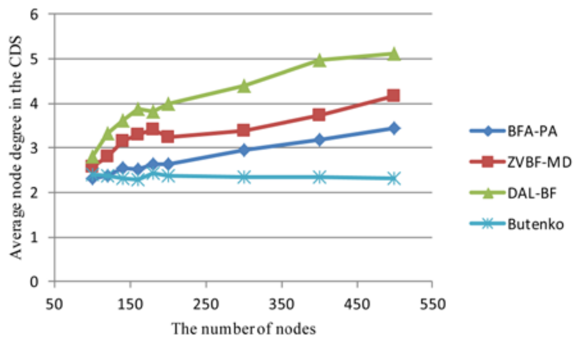

Fig.11.Average node degree in the CDS when SN is changed and *Tr* equals 15 units.

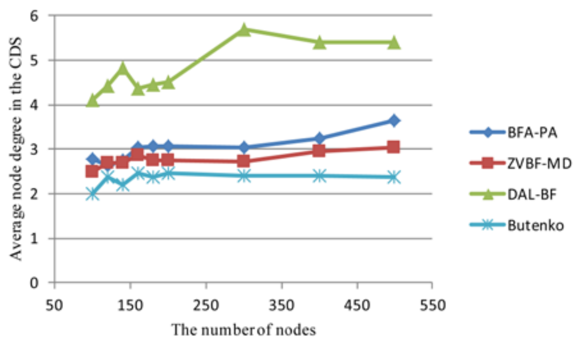

Fig.12. Average node degree in the CDS when SN is changed and *Tr* equals 30 units.

7 Conclusion

In this paper, based on the pursuit algorithm, we propose a novel backbone formation algorithm, called BFA-PA. In this algorithm, a DS is got at first and each node in the DS gets its action set based on the connection path between the dominator and their 2 or 3 hops dominator neighbors. Sink executes the pursuit algorithm and treads dominators as automata with to get an approximate solution of MCDS of network. Moreover, a changing speed of learning parameter is used to avoid choosing a special learning rate, which should be carefully selected to make a trade-off between the convergence speed and the size of backbone. It is also shown that our method is ε-optimality to connect the DS with the changing speed of learning parameter. According to the results of the simulation, our algorithm generates a smaller CDS than ZVBF-MD and has a similar result with DLA-BF about the size of CDS. However, it is efficient than DLA-BF because it has a small overhead.

However, in this algorithm, we assume that the network is stationary. When some new sensor nodes are added to the network or removed, the algorithm should be run again to get a backbone of network. Because the topology of network is changed, BFA-PA is difficult to work on a dynamic network. Is there a simple method to fix the backbone when there are some additions or failures of nodes? The answer to this question is significant to the further application of this method, and it will be a guide for our future work.

8 Acknowledgements

This work is supported partially bythe science and technology project of CQ CSTC (No. cstc2012jjA40037).

References:

[1] Yu J, Wang N, Wang G, et al. Connected dominating sets in wireless ad hoc and sensor networks–A comprehensive survey [J]. Computer Communications, 2013, 36(2): 121-134.

[2] Alzoubi K M, Wan P J, Frieder O. Maximal independent set, weakly-connected dominating set, and induced spanners in wireless ad hoc networks [J]. International Journal of Foundations of Computer Science, 2003, 14(02): 287-303.

[3] Han, Bo. Zone-based virtual backbone formation in wireless ad hoc networks [J]. Ad Hoc Networks 7.1 (2009): 183-200.

[4] Li Y, Thai MT, Wang F, Yi C-W, Wang P-J, Du D-Z. On greedy construction of connected dominating sets in wireless networks [J]. Special issue of Wireless Communications and Mobile Computing (WCMC), 2005.

[5] Gao B, Yang Y, Ma H. A new distributed approximation algorithm for constructing minimum connected dominating set in wireless ad hoc networks [J]. International Journal of Communication Systems 2005, 18(8), 734–762.

[6] Jie. Wu, Hailan. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks [C]. The Third ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (ACM DIALM 1999), August 1999, 7–14.

[7] Fei Dai and Jie Wu. An extended localized algorithm for connected dominating set formation in ad hoc wireless networks [J]. IEEE Transactions on Parallel and Distributed Systems 15 (10) (2004), 908–920.

[8] J. Akbari Torkestani, M.R. Meybodi. An intelligent backbone formation algorithm in wireless ad hoc networks based on distributed learning automata [J]. Computer Networks 54 (2010) 826–843.

[9] Akbari Torkestani J. Energy-efficient backbone formation in wireless sensor networks [J]. Computers & Electrical Engineering, 2013, 39(6): 1800-1811.

[10] He Jing, JiShoujing, Pan Yi, et al. Approximation algorithms for load-balanced virtual backbone construction in wireless sensor networks [J]. Theoretical Computer Science, 2013, 507: 2-16.

[11] Narendra, K. S. and Thathachar, M. A. L. Learning automata: An introduction. Englewood Cliffs, NJ: Prentice Hall (1989).

[12] M. A. L. Thathachar and P.S. Sastry. A Class of Rapidly Converging Algorithms for Learning Automata [C]. IEEE Int. Conf. on Cybernetics and Society, Bombay, India, Jan. 1984.

[13] M. A. L. Thathachar and P.S. Sastry. Estimator Algorithms for Learning Automata [J]. Proc. Platinum Jubilee Conf. on Syst. Signal Processing, Dept. Elec. Eng., Indian Institute of Science, Bangalore, India, Dec. 1986.

[14] Oommen B J, Agache M. Continuous and discretized pursuit learning schemes: Various

algorithms and their comparison [J]. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2001, 31(3): 277-287.
[15] Rajaraman, K. and Sastry, P. S.. Finite time analysis of the pursuit algorithm for learning automata [J]. IEEE Trans. Systems Man CybernetB(1996), 26, 590–598.
[16] Kaled M. Alzoubi, Peng-Jun Wan, Ophir Frieder. Maximal independent set, weakly connected dominating set, and induced spanners for mobile ad hoc networks [C]. International Journal of Foundations of Computer Science 2003 14(2), 287–303.
[17] Tilak, O., Martin, R., and Mukhopadhyay, S.,A decentralized, indirect method for learning automata games [J]. IEEE Trans. Systems Man Cybernet B (2011), 41, 1213–1223.
[18] W. Hoeffding. Probability inequalities for sums of bounded random variables [J]. Journal of the American Statistical Association, 1963, vol. 58, pp. 13-30.
[19] Butenko S, Cheng X, Oliveira C A, et al. A new heuristic for the minimum connected dominating set problem on ad hoc wireless networks [M]. Recent Developments in Cooperative Control and Optimization. Springer US, 2004: 61-73.