Yu *et al. EURASIP Journal on Wireless Communications and Networking* (2017) 2017:187
DOI 10.1186/s13638-017-0961-9

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Multi-leader election in dynamic sensor networks

Kan Yu[1], Meng Gao[2], Honglu Jiang[2] and Guangshun Li[2*]

Abstract

The leader election problem is one of the fundamental problems in distributed computing. Different from most of the existing results studying the multi-leader election in static networks or one leader election in dynamic networks, in this paper, we focus on the multi-leader election in dynamic sensor networks where nodes are deployed randomly. A centralized simple leader election algorithm (VLE), a distributed leader election algorithm (NMDLE), and a multi-leader election algorithm (PSMLE) are proposed so as to elect multi-leaders for the purpose of saving energy and prolonging the network lifetime, respectively. Specifically, the proposed algorithms aim at using less leaders to control the whole network, which is controlled by at least k_{opt} leaders, here k_{opt} denotes the optimal number of network partitions. Then we analyze the impacts of the sleep scheme of nodes and node moving on energy consumption and establish a theoretical model for energy cost. Finally, we provide extensive simulation results valuating the correctness of theoretical analysis.

Keywords: Multi-leader election, Dynamic networks, Energy consumption, Network lifetime

1 Introduction

A sensor network (SN) is a wireless network that consists of small, low-cost sensors, which can sense the environment information, then collect and disseminate message. A wireless sensor network (WSN) is spatially distributed to monitor and control the given area with a random or deterministic manner. WSNs have applications in a lot of fields such as military purpose environmental monitoring and disaster prevention [1–5]. Thus, a sensor network can be described as a collection of sensor nodes that can perform specific actions. Unlike traditional wired networks, the sensors communicate with each other and carry out the common goal cooperatively.

A dynamic network (DN) is a random network that changes over time. Different from the static networks, nodes in dynamic networks are distributed randomly and have mobility to any direction with a given rate. For the problems in DNs, most of the existing works adopt the ideas from static networks such as leader election, connected dominating sets construction, and topology control. A dynamic sensor network (DSN) consists of mobile hand-held smart devices. These devices are carried by humans or attached to mobile objects such as cars. They can collect the sensing data within a particular range and report the data to the Internet by WiFi or 2G/3G/4G networks. DSNs also have applications in other domains such as environmental monitoring [6], transportation [7], and social networking [8]. Thus, a DSN can be described as a dynamic set of nodes (i.e., sensors), which partially is connected by dynamic undirected communication links. Unlike static networks, nodes in DSNs are dynamically added to the network and are removed from the network.

Leader election is a fundamental problem in distributed computing for breaking symmetry and studying other problems, such as broadcasting and data aggregation. Leader election is for each processor eventually to decide that whether it is a leader or not subject to only one processor decides that it is the leader [9]. In this paper, we study the multi-leader election problem in dynamic sensor networks, the main reason is that if there exist multiple leaders in the network, it is convenient to achieve power management, resource allocation, message collection, and energy saving. There have been many results for multi-leader election problem, such as in [10]; Chung et al. applied "regional consecutive leader election" to mark the process of leader election.

*Correspondence: guangshunli@sohu.com
[2]School of Information Science and Engineering Qufu Normal University, 276826 Rizhao, Shandong, China
Full list of author information is available at the end of the article

In this paper, we first use Voronoi diagram to divide the network and obtain the number of the regions. Secondly, we give the solution of redundant nodes for saving energy. Finally, we design algorithms to solve the multi-leader election problem. Specifically, we need to consider two cases where the nodes may leave or join in the network.

The main contributions of this paper can be summarized as follows. For the target of saving energy and extending network lifetime, we propose a centralized leader election algorithm (VLE), a distributed leader election algorithm (NMDLE), and a multi-leader election algorithm (PSMLE). Different from the previous works, VLE, NMDLE, and PSMLE have the following advantages.

1. Voronoi division, mobility model, and communication model are proposed respectively; based on the optimal number of partitions k_{opt}, we can get the number of leaders.
2. NMDLE and PSMLE can deal with redundant nodes, resulting in reducing the node redundancy and energy consumption and improving network performance.
3. NMDLE and PSMLE can balance the energy consumption among sensor nodes and prolong the lifetime of the whole network by the schemes of sleeping and period sleeping.
4. NMDLE and PSMLE consider the mobility of nodes, and we present two management schemes of energy for different mobility scenarios.
5. NMDLE and PSMLE are distributed algorithms, so that they can be easily implemented in DSNs.

The rest of the paper is organized as follows. Section 2 provides an overview of the related work. In Section 3, we present the definition of leader election, network model, mobility model, and the overview of our design. We provide the detailed execution process of the designed algorithms in Section 4. We analyze the performance of the three algorithms in Section 5 and present the simulation results in Section 6. Finally, we summarize our work and conclude the paper in Section 7.

2 Related work

Distributed computing and wireless communication are typical applications for randomization, since the randomized algorithms are simpler and more time-efficient than the deterministic algorithms. Leader election is the fundamental problem in the area of distributed algorithms and is studied over decades in various communication and network models. Recently, the leader election problem is receiving more and more attention in dynamic networks. In this section, we present a brief overview of the related work on the leader election in various wireless networks.

Existing solutions to dynamic leader election problem consider only deterministic algorithms [10, 11], but the powerful adversary knows the execution of algorithm in advance.

In [10], Chung et al. defined the regional consecutive leader election (RCLE) problem. They extended the traditional leader election problem to mobile ad hoc network and gave an algorithm for solving RCLE in two-dimensional or three-dimensional spaces. The proposed algorithm is independent of the total number of nodes in the network and the startup time. In addition, they provided an algorithm that solves the RCLE problem with a message bit complexity of $O(n(\log n + \log r))$ per node in round r, n and is the number of nodes.

An improved result of [10] was shown in [11], in which the RCLE requires that mobile nodes must elect a leader in bounded time, and an algorithm with the time complexity of $\Omega(Dn)$ was proposed, where D is the diameter of the network and n is the number of nodes. In [11], Chung et al. presented a matching algorithm that guarantees algorithm can terminate in $O(Dn)$ rounds and proved that the algorithm presented in [10] only sends once broadcasting message per round.

Based on the context of mobile ad hoc networks, in [12], Malpani et al. presented a leader election algorithm in every connected component. The proposed algorithm can make sure that there is only one leader in the final network topology, based on routing algorithm given by TORA, and validate the algorithmic correctness. In [13], Ingram et al. gave a protocol of leader election in asynchronous and changeable network topology, it can successfully elect a leader before the topology changes as soon as possible. Then they further extended the scheme proposed in [12] to deal with the changing topology. There are several other leader election algorithms for mobile environment, such as [14–17].

In [18], Kuhn et al. gave a new definition which is called as abstract MAC layer, and they proved the correctness of the new layer by solving the multi-message broadcast and regional leader election problem.

In [19], Kuhn et al. introduced some models and algorithms in dynamic networks and studied them by a simple manner. However, these models have same characteristic assumptions: the nodes keep moving, and the number of nodes and the mobility frequency can not be limited.

In [20], Augustine et al. considered the model of peer-to-peer dynamic network, where nodes can leave or enter the network randomly and the topology can tolerate some changes of the network. They gave a random distributed leader election algorithm with multiple logarithmic time, ensuring almost everywhere to be stable under the high probability of interference.

In addition, for leader election problem, many existing papers considered the lower bound of time complexity and message complexity in the static synchronous networks.

In [21], Korach et al. presented the lower bound of message complexity, namely $O(n \log n)$. In [22], Afek and Gafni studied electing a leader in both synchronous and asynchronous complete networks. The message complexity of the synchronous algorithm is $O(n \log n)$, and they also proved that any message-optimal synchronous algorithm requires $\Omega(\log n)$ time. In [23], Kutten et al. focused on studying the message and time complexities of randomized implicit leader election in synchronous distributed networks. The lower bounds of message complexity and time complexity are $\Omega(m)$ and $\Omega(D)$, respectively, where m denotes the number of edges and D is the diameter of the network.

In [21–23], although they gave the deterministic lower bounds of time and message complexities, these lower bounds are not suitable for our model. When nodes leave or enter the network dynamically, the network topology changes over time.

In [24], Pandurangan et al. presented a random leader election algorithm in synchronous distributed networks. It elected one leader in a complete graph, running $O(1)$ round, and the message complexity is $O\left(\sqrt{n}\log^{\frac{3}{2}} n\right)$.

In [25], Dinitz et al. generalized smoothed analysis to distributed algorithms in dynamic networks. They applied this technique to solve three problems of standard dynamic network with the worst-case lower bounds, these problems are random walks, flooding, and aggregation.

It should be noted that there have been many other approaches such as connected dominating set-based (e.g., [26–28]) and clustering-based (e.g., [29, 30]) methods to select leads in wireless sensor networks.

3 Model and definition

The characteristics of dynamic networks are that the network topology changes over time due to the mobility of nodes. Generally, a dynamic network can be denoted by $G = \langle G_1, G_2, ..., G_t \rangle$, here $G_t = (V_t, E_t)$ is network topology at time t, in which V_t and E_t denote the sets of nodes and edges, respectively. All nodes follow the poisson distribution in two-dimensional space R^2, and each node has unique identifier *ID*.

3.1 Definition of the problem

Initially, all nodes have the same transmission power P, and each node can adjust it within the available range, so that we can ensure that nodes communicate with other nodes by the minimum power. In addition, assume that nodes can only enter or leave the network once for simplicity, but it moves to any directions more than once in the network. The moving rate of nodes is heterogeneous, denoted by v, and the moving direction is selected randomly in $[0, 2\pi]$, the maximum moving time is denoted by $t_{\max}$.

We partition the network based on Voronoi diagrams, the main reasons are listed as follows.

1. In previous works, they usually apply regular hexagon to divide the network, since it can make full use of the circular domain of nodes when we apply hexagon. Compared with the square mesh, hexagon will use less nodes to control the whole network.
2. Now we choose the Voronoi, which will use more less nodes than regular hexagon, and the application of the Voronoi diagram theory can effectively solve the closest point search.

Moreover, we extend the traditional Voronoi that only one node in one area, a leader node and several non-leader nodes are located in this area at the same time. The method we use to partition the network is described as follows.

First, we select k nodes randomly after initializing the network topology, then we partition the network as k areas, we call it the *Voronoi I*.

Second, we execute simple leader election algorithm that just depends on the *ID* of nodes in each area.

Finally, when the leader election algorithm terminates, we need to partition the network again. In this partition, we make the leaders be the center of the k areas, we call it the *Voronoi II*.

In the latter of this paper, the Voronoi area that we mentioned is *Voronoi II*. In addition, we give the following hypotheses about dynamic networks.

1. All nodes have the same sensing range and the communication range.
2. In order to improve the effectiveness of leader election algorithms, we assume that nodes keep stationary state within some rounds.
3. The communication among nodes is symmetric.
4. The number of nodes which move at the same time has an upper bound, denoted by N^{u}_{moving}.

The parameters are shown in the Table 1.

3.2 Model of node distribution

We divide the network into several regions and form a Voronoi cell that the leader is located in the center for each region. We use the property of Poisson Point Process (PPP), that is the independent thinning, to analyze energy consumption of nodes.

Denote the set of leaders and the set of non-leaders as poisson process Φ_r and Φ_b with densities λ_1 and λ_2, respectively.

Table 1 Symbols and description

Symbols	Description
n	The number of dead nodes
D	The diameter of network
P	The initial transmission power
λ_1	The density of leaders
λ_2	The density of non-leaders
k	The number of regions that partitioned by Voronoi
N_u	The number of neighbors of node u
k_{opt}	The number of optimal regions
p	The probability that a node successfully is elected as a leader
L	The sum of the distance between non-leaders and leaders
N	The number of non-leaders in the region
d	The average distances between non-leaders and leaders

Lemma 1 *Φ_r and Φ_b are independent, and their densities are λp and $\lambda(1-p)$, respectively.*

Denote the probability that any node can be elected a leader successfully be p. Thus it is concluded that $\lambda_1 = p\lambda$, $\lambda_2 = (1-p)\lambda$. The multi-leader election aims to divide the whole wireless network into several partitions, namely, one leader controls an area. Thus, the number of network partition is $k = np$.

Similar to the method in [31], we calculate the expected number of the non-leader after partitioning the network, and the expectation of the distance between non-leaders and leader.

For any voronoi cell, we define

$$S = \sum_{x_i \in V_j(\pi_2)} f(x_i). \tag{1}$$

when $f(x) = 1$, S denotes the number of non-leaders in the region, denoted by N.

when $f(x) = x$, S is the sum of the distance between non-leaders and leaders, denoted by L.

Considering Theorem 1 in [31], we can obtain the following conclusions:

$$E[S] = \lambda_2 \int_{x \in V_j(\pi_2)} f(x) e^{-\lambda_1 \pi x^2} dx, \tag{2}$$

$$E[N] = \lambda_2 \int_{x \in V_j(\pi_2)} e^{-\lambda_1 \pi x^2} dx = \frac{\lambda_2}{\lambda_1}, \tag{3}$$

$$E[L] = \lambda_2 \int_{x \in V_j(\pi_2)} x e^{-\lambda_1 \pi x^2} dx = \frac{\lambda_2}{2\lambda_1^{3/2}}. \tag{4}$$

The average distance between non-leaders and leaders in the region is

$$d = \frac{E[L]}{E[N]} = \frac{1}{2\sqrt{\lambda_1}}.$$

when $f(x) = x^2$,

$$E\left[\sum_{x_i \in V_j(\pi_2)} x_i^2\right] = \frac{\lambda_2}{\pi \lambda_1^2}.$$

3.3 Model and analysis of energy consumption

Generally speaking, a sensor has four modules, those are communication module, sleep module, detection module, and calculation module. The first module accounts for 90% the energy consumption, thus the key for reducing the energy consumption focuses on how to deal with the communication process of nodes properly. Similarly, we use the wireless node model in [32, 33] to model the energy consumption of nodes. In this way, we show that although the cost of energy is the same as other algorithms in one node, by using the designed algorithm, the total energy consumption is lower.

The energy consumption that node transmits data l, with distance d is calculated by

$$e_{Tx}(l,d) = \begin{cases} le_0 + l\varepsilon_{fs}d^2 & (d < d_0) \\ le_0 + l\varepsilon_{mp}d^4 & d \geq d_0). \end{cases} \tag{5}$$

In Eq. (5), constant e_0 is the standard energy consumption that a node transmits or receives 1 bit data. ε_{fs} is the specific power consumption of power amplifier in free space model, ε_{mp} is specific power consumption of power amplifier in multi-path attenuation model, and $d_0 = \sqrt{\frac{\varepsilon_{fs}}{\varepsilon_{mp}}}$ is the threshold of transmission distance, which is used to determine the attenuation model. This is, if the transmission distance $d < d_0$, we adopt the free space model; otherwise, the multi-path attenuation model is adopted.

The needed energy that a node receives data l is

$$e_{Rx}(l) = le_0. \tag{6}$$

When all the non-leaders send data l to a leader, the total energy consumption is

$$\begin{aligned} E_{no-le} &= E\left[\sum_{x_i \in V_j(\pi_2)} (e_{Tx}(l,x_i) + e_{Rx}(l))\right] \\ &= E\left[\sum_{x_i \in V_j(\pi_2)} \left((le_0 + l\varepsilon_{fs}x_i^2) + le_0\right)\right] \\ &= E\left[\sum_{x_i \in V_j(\pi_2)} \left(2le_0 + l\varepsilon_{fs}x_i^2\right)\right] \\ &= E\left[\sum_{x_i \in V_j(\pi_2)} 2le_0\right] + E\left[\sum_{x_i \in V_j(\pi_2)} l\varepsilon_{fs}x_i^2)\right] \end{aligned} \tag{7}$$

$$= 2le_0\frac{\lambda_2}{\lambda_1} + l\varepsilon_{fs}\frac{\lambda_2}{\pi\lambda_1^2}$$
$$= \frac{l\lambda_2}{\lambda_1}\left(2e_0 + \frac{\varepsilon_{fs}}{\pi\lambda_1}\right).$$

We need to elect a new leader once at least an old leader leaves the current area, and the residual energy of nodes can be determined by the following equations.

Let the initial energy of a node be E.

The energy consumption that a leader receives all messages transmitted in one round is

$$E_{leader} = \sum_{x_i \in V_j(\pi_2)} le_0 = le_0 E[N] = le_0\frac{\lambda_2}{\lambda_1}. \quad (8)$$

The energy that a non-leader sends message to a leader in one round is

$$E_{non-leader} = \begin{cases} le_0 + l\varepsilon_{fs}x_i^2 & (d < d_0). \\ le_0 + l\varepsilon_{mp}d^4 & (d \geq d_0). \end{cases} \quad (9)$$

The energy cost in the whole network is

$$E_w = k(E_{total} + E_{non-leader}). \quad (10)$$

Thus, the rest energy of a leader after a communication round is

$$E_{re-le} = E - E_{leader}. \quad (11)$$

Similarly, the remain energy of a non-leader is

$$E_{re-no} = E - E_{non-leader}. \quad (12)$$

After running n rounds, the rest energy of a leader is

$$E_{remain-le} = E = nE_{leader} = E - le_0\frac{n\lambda_2}{\lambda_1}, \quad (13)$$

and the rest energy of a non-leader is

$$E_{remain-no} = E - nE_{non-leader} = E - nle_0 - nl\varepsilon_{fs}x_i^2. \quad (14)$$

3.4 Model of node mobility

The mobility model of node is usually used to describe the mobile patterns, such as direction, rate, and acceleration changing with time. It plays an important role in determining the algorithmic performance and is a reasonable and ideal model to simulate the real world. Over the past years, many mobility models are designed to recreate the real-world scenarios. There are many kinds of random-based mobility models; in this paper, we consider random walk mobility model and random waypoint mobility model.

Random walk mobility model

The simplest mobility model is the random walk mobility model [34], and is also called Brownian motion; this model was first described in mathematical way in 1926 by Einstein. In this model, the moving node can move to a new position that the direction and rate v are randomly selected within the scope of $[0, 2\pi]$ and $[v_{min}, v_{max}]$, respectively.

Random walk model is a memoryless mobility process, because the future decision is independent of the previous statuses. In other words, the current rate is independent of previous rates, and the future rate is also independent of the current one. We simplify it by assuming that all nodes have the same moving rate in the simulation.

Random waypoint mobility model

Another random mobility model is the random waypoint mobility model [35]. This model was first proposed by Johnson and Maltz. It can be considered as the extension of random walk mobility model, and contains the pause times to change moving direction and rate. A moving node can keep stationary in one point for a period of time, we call this period as a pause time. Once the time is over, the moving node needs to select randomly a destination in the given area, and a rate v which is limited in $[v_{\min}, v_{\max}]$.

In the random waypoint model, the rate and the length of pause time are the two key parameters that determine the mobility behavior of nodes and network performance. It can be simplified to random waypoint mobility model without considering pause time.

4 Algorithm description

Due to the limited battery life of sensor nodes, it is necessary to design an energy efficient algorithm that can simultaneously reduce computation and communication overheads. To defeat this challenge, we propose algorithms VLE, NMDLE, and PSMLE that can satisfy these conditions simultaneously for the dynamic sensor networks with many sensor nodes. These algorithms deal well with the leader election problem, and every Voronoi area will have only a leader after algorithm termination. A leader can manage all the non-leader nodes in the current area and communicates with other leaders in different areas by one-hop or multi-hop way.

There are two cases that the leader cannot work, one is that a leader moves out of the current Voronoi cell, it will lose the role of a leader and become a non-leader; another is that the remaining energy of a leader could not satisfy the minimum energy constraint. When the leader is not available, then the area needs to elect a new leader timely. For trading off energy among nodes and achieving monitoring task, we calculate the residual energy of each node and sort all non-leaders by the descending order of *ID* after the moving procedure ending. Once the leader election begins, we can choose the node whose residual energy is the largest as the new leader, to break symmetry, when at least two nodes have the same remaining energy, we choose the one who has bigger *ID* as the new leader. Thus, we can save energy and prolong the lifetime of networks.

To further improve network performance, based on the sleeping module of sensor nodes to consider redundant nodes, we introduce the mechanism sleeping and waking up of nodes. When the density of nodes is greater than the upper bound of threshold, algorithm makes a part of non-leaders sleeping to save the energy; when the density of nodes is less than the lower bound of threshold, the algorithm will wake up some sleeping nodes to make sure that the whole area has a proper topology structure.

Symbols and parameters used in the algorithms are listed in Table 2.

4.1 Leader election algorithm

In this section, we first propose a simple leader election algorithm based on Voronoi division. The step of the algorithm is given as follows.

1. **Network division**
   In this step, we select k nodes randomly and divide the network into k Voronoi areas.
2. **Information exchange**
   Nodes have unique *ID* and have the same initial power P. For each area, each node broadcasts the information of its *ID* and distance to all one-hop neighbors. In this way, each node will keep the list of *ID* and distance information for its one-hop neighbors. Next, each node compares its own *ID* with those of its neighbors.
3. ***ID* sort**
   Each node sorts the *ID* including its own and all neighbors by the descending order.
4. **Leader election**
   If the *ID* of node u is the largest in its neighbors, and the remaining energy is larger than 10%E, then it will change its state to *leader* and control all its neighbors. If the remaining energy of the node equals to zero, the node dies, we use the N_{death} to denote the number of dead nodes. If the node is not available under two cases, that is, the *ID* of the node is smaller than the *ID* received from its neighbors, the node will change state to *nonleader*, and it will be controlled by the node who has the largest *ID*.

The pseudo code of Algorithm 1 is shown as follows.

Algorithm 1 Algorithm 1: Voronoi based Multi-leader Election Algorithm (VLE)

```
Initialization
Traveling all nodes in the network, and partitioning the network by Voronoi diagram
The states of nodes: leader, nonleader
Using $N_{death}$ to denote that the number of dead nodes
Phase I: Exchanging Message
1. for each node $u$ in the network do
2.   Node $u$ broadcasts ID and distance messages to one-hop neighbors
3.   Node $u$ receives message from its neighbors
4.   Node $u$ compares its ID with those of its neighbors
5. end for
6. Each node sorts all neighbors by the descending order of ID
Phase II: Area Leader Election
7. if ID of node $u$ is the largest in its neighbors and $E_{re-le} \geq 10\%E$ then
8.   $u$ changes to leader state
9.   $u$ broadcasts this message to its neighbors
10.  else if $E_{re-no} = 0$ then
11.    the node dies and $N_{death} = N_{death} + 1$
12.  else
13.    Change the state of $u$ to nonleader
14.  end if
15. end if
```

Table 2 Symbols and description

Symbols	Description
N_{death}	The number of dead nodes
E	The initial energy of nodes
E_{re-le}	The remaining energy of a leader
E_{re-no}	The remaining energy of a non-leader
itMovingID	The set of the it ID of moving nodes
t	The moving time of a moving node
t_{max}	The maximum time allowing to move
v	The moving rate of a moving node
NL_i	The non-leader node i
SR_i	The sensing radius of node i
T_i	The set of non-leaders with the ascending order of it ID
$\|T_i\|$	The size of T_i
δ	The minimum energy cost when sends one message

4.2 Distributed multi-leader election algorithms

Algorithm 2 is a distributed leader election algorithm. Different from Algorithm 1, this algorithm considers the moving of nodes. Algorithm 2 contains two processes and one mechanism, the two processes are about moving and new leader election, the mechanism is about nodes sleeping. In the process of moving, the node selects the direction of moving randomly, the node exchanges messages with its neighbors after moving. In the process of new leader election, if the moving node is a leader, the node who has the largest *ID* and the remaining energy is larger than 10%E will change the state to *leader*, or we can find a leader that meets the conditions according to the descending order of *ID* and energy constraint. Once the topology changes, we must check the network immediately due to ensure the stability of the network. If the moving node is a non-leader node, it needs to update the information of *ID* and distances with its neighbors, then algorithm restarts to execute the new leader election in the new area. When the process of leader election is over, the new leader will broadcast the message that it is the new leader to its neighbors. For the node sleeping mechanism, we proof that if one node and its neighbors cover the common area more than 68%, the leader will send sleeping message to other non-leader nodes. The sleeping nodes do not participate in the leader election. In order to maintain the connectivity of the network and trade off energy among nodes, the sleeping nodes are waken up after running 10 rounds.

We give the Algorithm 2 (NMDLE), the pseudo code are shown as follows.

In Algorithm 2, because of more energy is consumed when nodes transmit and receive messages, leaders or non-leader nodes are easy to die, then the area will not be monitored. In order to reduce the energy consumption of nodes and further prolong the lifetime of the network. We propose a novel but simple scheme of saving energy, that is, the multi-leader election algorithm based on periodic sleeping mechanism.

Algorithm 3 is applicable to deal with the case where only non-sleeping nodes are allowed to move and all nodes achieve moving process. In each Voronoi cell, we have two operations, first every non-leader node senses the other nodes within its half of sensing radius, sorts them by the ascending order of *ID* and stores them by set Γ; secondly, the first non-leader hold the token, and the other nodes in Γ come into sleeping. Then we divide the problem into three cases. The detail steps are given in Algorithm 3.

The advantages of Algorithm 3 are listed as follows.

1. Different from Algorithm 1, we introduce the mechanism of saving energy.
2. Different from Algorithm 2, Algorithm 3 is more accurate and efficient.
   Because the rate of coverage is higher than that in Algorithm 2, then when one node is in sleeping state, another node can replace it and monitor the area.
3. This mechanism is a novel method to save energy.
4. This mechanism achieves the goal that makes the balance of energy consumption among nodes for the first time, namely after running the algorithm several rounds, the residual energy of every node is similar, which is more beneficial to maintain network topology and extend network lifetime.

The pseudo code are given as following Algorithm 3.

Algorithm 2 Algorithm 2: Node Moving based Distributed Multi-leader Election Algorithm (NMDLE)

```
Initialization
The moving rate of node is $v$, the moving direction is selected randomly from $[0, 2\pi]$, the maximum time that allowing to move for any node is $t_{max}$
Operation of moving nodes
Using the function randint() to select the moving nodes randomly, that is,
MovingID={ID | corresponding to moving nodes}
Process of moving
Each moving node chooses a random moving direction, and moving towards, this direction $vt$ meters with rate $v$ and time $t$
After moving, each node exchanges messages with its new neighbors
Process of new leader election
1. if the previous state of moving node $u$ is leader then
2.    The Voronoi area that it managed waits for electing new leader after achieving moving
3.    Each node updates the information of ID and distance with neighbors
4.    Each node sorts all neighbors by the descending order of ID
5.    if there is no ID in neighbor set is larger than that of $u$ and $E_{re-le} \geq 10\%E$ holds for $u$ then
6.       $u$ does not change the state
7.    else
8.       Wait to receive message for leader
9.    end if
10. else
11.    if the node has the largest ID in neighbor set and $E_{re-le} \geq 10\%E$ then
11.       The state of node changes to leader
12.    else
13.       Wait for the message from leader
14.    end if
15. The leader broadcasts message that it is the leader to its neighbors
16. end if
Nodes sleeping mechanism
17. if at least one of neighbors of leader $u$ covers 68% area with it then
18.    The leader send sleeping message to those nodes and they are in sleeping state
19.    The sleeping nodes do not participate in the process of current leader election
20.    After running 10 rounds, the sleeping nodes are waken up
21. end if
```

5 Algorithm analysis

Lemma 2 *Chung et al.* [11] *For the problem of leader election, algorithms need $\Omega(Dn)$ rounds, that D is the diameter of networks, n is the number of nodes in the network.*

Algorithm 3 Algorithm 3: Periodic sleeping mechanism based Multi-leader election Algorithm (PSMLE)

```
For every Voronoi cell, we have the operation as follows:
1. Each non-leader node $NL_i$ senses the other non-leader nodes within half of sensing radius, i.e., $0.5 \cdot SR_i$, and then sorts all these nodes with the ascending order of ID, denoted by $T_i = \{NL_i, NL_j, ..., NL_k\}$, here $i < j < ... < k$.
2. Toss the token to node $NL_i$ , and other nodes in $T_i \backslash \{NL_i\}$ fall into sleeping state, whose the sleeping period is the size of $T_i$, namely $|T_i|$.
Case 1: Before the sleeping period of nodes in $T_i \backslash \{NL_i\}$ ending, node $NL_i$ participates in the moving process.
3. if $T_i$ does not change and the residual energy of $NL_i$ is larger than the energy threshold $\delta$ then
4.   $NL_i$ continues to work
5. else
6.   $NL_i$ handovers the token to the node $NL_j$ which has the biggest ID in $T_i \backslash \{NL_i\}$
7. end if
8. if the set $T_i$ changes then
9.   Wait to the sleeping nodes until they wakes up, then running the step 1
10. end if
Case 2: When the sleeping period of nodes in $T_i \backslash \{NL_i\}$ is over, node $NL_i$ does not participate in the moving process.
11. Update $T_i$ ( due to node moving can change $T_i$ after the sleeping period)
12. if the residual energy of $NL_i$ is larger than the energy threshold $\delta$ then
13.   $NL_i$ continues to work
14. else
15.   $NL_i$ handovers the token to the node $NL_j$ which has the largest ID in $T_i \backslash \{NL_i\}$
16. end if
Case 3: When the sleeping period of nodes in $T_i \backslash \{NL_i\}$ is over, node $NL_i$ participates in the moving process.
17. Update $T_i$ (other nodes change $T_i$ due to node moving after the sleeping period ending)
18. if the ID of $NL_i$ is still the largest and the residual energy of $NL_i$ is larger than the energy threshold $\delta$ then
19.   $NL_i$ continues to work
20. else
21.   $NL_i$ handovers the token to the node $NL_j$ which has the largest ID in $T_i \backslash \{NL_i\}$
22. end if
```

Theorem 1 *The message complexity of Algorithm* 1 *is* $O(N_u)$*, here* N_u *is the set of neighbors of* u.

Proof For any node u, during the processing of this algorithm, when nodes broadcast their own *ID*, they send a message to their one-hop neighbors, and each node will receive feedbacks from their neighbors, the number of nodes has an upper bound in each region, then the number of messages that they all need to send also is bounded, thus the message complexity of Algorithm 1 is $O(N_u)$. □

Theorem 2 *The time complexity of Algorithm* 1 *is* $O((1 + \log N_{\max})N_{\max})$ *rounds.*

Proof First, the time complexity of broadcasting its own *ID* and distance messages in Phase I is $O(1)$. Define $N_{\max}$ as the maximum number of neighbors for any node, the time complexity of comparing the messages with their neighbors and sorting the *ID* in Phase I is $O(N_{\max} \log N_{\max})$. Secondly, in Phase *II*, the time complexity is at most $N_{\max} - 1$ in the worst case, this is the first $N_{\max} - 1$ nodes which do not satisfy the condition $E_{re-le} \geq 10\%E$. Therefore, the total time complexity of the Algorithm 1 is $O((1 + \log N_{\max})N_{\max})$. □

Theorem 3 *The lower and upper bounds of the time complexity of Algorithm* 2 *are* $O(N_{\max} + N^u_{moving} \log N^u_{moving})$ *and* $O((1+\log N_{\max})N_{\max})$ *respectively,* N^u_{moving} *denotes the maximum number of moving nodes.*

Proof Algorithm 2 runs respectively in each Voronoi partition. According to one of the partitions, we can obtain the time complexity. Similarly, if the previous state of moving nodes is *nonleader*, the time complexity of electing new leader after moving is $O((1 + \log N_{\max})N_{\max})$. On the contrary, if previous state of moving nodes is *leader*, it only needs to compare its *ID* with other neighbors, the time complexity is at most $O\left(N_{\max} + N^u_{moving} \log N^u_{moving}\right)$. □

Theorem 4 *For each node, if its neighbors have one node which covers 68% area with it, then the leader sends the sleeping message to its neighbors.*

Proof Algorithm 2 uses the sleeping mechanism to constrain the number of sleeping nodes for saving energy. We can get this condition when communication circles of two nodes intersect from each other, we can calculate the area of coverage. We assume that the distance between the center of two circles is c, the radius of the two circles is r, the angle of the sector is 2α as shown in Fig. 1. The intersection section of the two circles and the method of calculating the area or coverage are given as follows.

$$\begin{aligned} S_{intersect} &= 2 \times (S_{sector} - S_{triangle}) \\ &= 2 \times \left[\frac{\pi \times 2 \arcsin \frac{\sqrt{r^2 - \frac{c^2}{4}}}{r} \times r^2}{2\pi} - \frac{1}{2} \times 2\sqrt{r^2 - \frac{c^2}{4}} \times \frac{c}{2} \right] \\ &= 2 \arcsin \frac{\sqrt{r^2 - \frac{c^2}{4}}}{r} \times r^2 - c \times \sqrt{r^2 - \frac{c^2}{4}} \end{aligned} \tag{15}$$

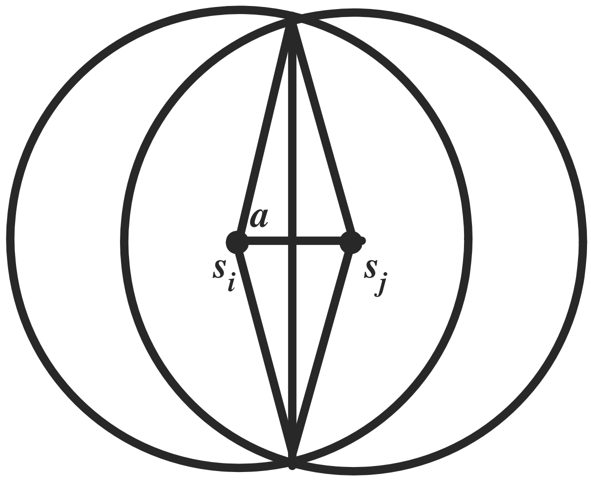

Fig. 1 The intersection section of the two circles

According to the above method, we can calculate the area of coverage.

5.1 The first case, the center distance is radius, that is, $c = r$.

For the scenario where the radius of the two circles is the same, in this case, we can calculate the coverage area simply (Fig. 1). The area of circle is πr^2. The area of coverage is

$$\frac{2\pi r^2}{3} - \frac{\sqrt{3}r^2}{2}.$$

Then we calculate the percentage of that the area of the coverage accounts for that of the circle, that is, 39%.

5.2 The second case, the center distance is 0.5 radius, that is $c = 0.5r$.

In this case, the area of circle is also πr^2. We let the c in formula (15) be $0.5r$, then we get the area of coverage is

$$2\arcsin\frac{\sqrt{15}}{4} \times r^2 - \frac{\sqrt{15}r^2}{8}.$$

Then we calculate the percentage of that the area of the coverage accounts for that of the circle, that is, 68%.

5.3 The third case, the center distance is 0.2 radius, that is, $c = 0.2r$.

In this case, the area of circle is also πr^2. We let the c in formula (15) be $0.2r$, then we get the area of coverage is

$$2\arcsin\frac{3\sqrt{11}}{4} \times r^2 - \frac{3\sqrt{11}r^2}{50}.$$

Then we calculate the percentage of that the area of the coverage accounts for that of the circle, that is, 87%.

Then we adopt the second result of the area of coverage to limit the sleeping nodes in Algorithm 2, that is, 68%. □

Theorem 5 *The time complexity of Algorithm* 3 *is*

$$O(N_{radius} \log N_{radius} + D),$$

where N_{radius} denotes the maximum number of nodes in communication radius.

Proof In the Algorithm 3, when the non-leader node senses the other non-leader nodes, and sorts these nodes, the complexity of time is $O(N_{radius} \log N_{radius})$. When the node which holds the token dies or the residual energy of it is lower than energy threshold δ, it will handover the token to the next node, which has the largest *ID*, the time complexity is $O(D)$, here D is the diameter of the network. Thus, the time complexity of Algorithm 3 is $O(N_{radius} \log N_{radius} + D)$. □

6 Evaluation

In this section, we evaluate the performance of our proposed multi-leader selection algorithms via simulation. We initialize a network where nodes are deployed in a 300 m× 300 m region. Additionally, we consider the following parameters.

1. The initial battery energy, moving rate and moving time of each node are uniform. The energy consumptions of sending and receiving a data also are same and equal to 0.00000005.
2. The communication and sensing ranges are between 87.7058 m and 43.879 m, respectively.
3. We divide the lifetime of the network into rounds. Every 10 rounds, at most 10 nodes participate in moving process. And the energy consumption of once moving equals to 0.000001.

Other parameters are listed in the following Table 3.

First, we consider the performance of Algorithm 1, as shown in Fig. 2. To valuate the impact of executing rounds of Algorithm 1 on energy consumption, we consider networks where the number of nodes is 100, the maximum moving time is 20 s. The red solid line shows the result that no nodes participate in moving. In this case, multi-leader only receives data from their neighbors, energy consumption considered includes two parts: energy of sending data and energy of receiving data. However, once nodes move to any direction with maximum distance 300 m and rate 5 m/s, resulting in more energy consumption. In Fig. 2, we can see that remaining energy of nodes decreases with the executing rounds increasing and the number of moving nodes is harmful to remaining energy. This is because that, on the one hand, more

Table 3 Parameters and values

Parameters	Values
The initial energy of nodes E_0	0.5
The number of sending data l	4000
The cost energy of nodes sending data ETX	50*0.000000001
The energy of multi-path attenuation model EDA	5*0.000000001
The energy cost of communication e_0	ETX+EDA
The frequency of nodes moving	10 rounds
The maximum number of moving nodes	0 15
Specific power consumption of power amplifier ε_{fs}	10* 0.000000000001
Specific power consumption of power amplifier ε_{mp}	0.0013* 0.000000000001
The energy cost of nodes receiving data ERX	50*0.000000001
The threshold of transmission distance d_0	87.7058 m
The moving rate	5 m/s
The maximum moving time	80 s

the number of moving nodes is, more energy consumption is. But when the executing rounds are less than 40, the difference between energy consumptions of 5 moving nodes and 7 moving nodes is small. This is because that although 7 moving nodes can result in more 0.000002 energy consumption, 5 moving nodes have more number of neighbors and cost more energy.

Next, we consider the impact of moving time on energy consumption. For each curve in Fig. 3, we can see that for the same number of moving nodes, energy consumption presents the rising trend on the whole apart form the red solid line. This is because that the larger moving time is, the larger moving distance is, nodes have more number of neighbors, resulting in more energy consumption.

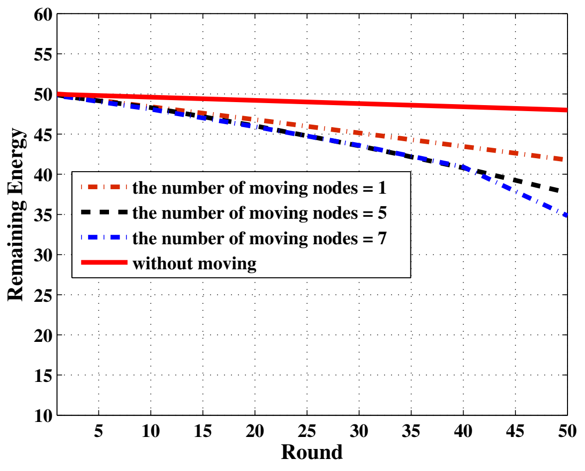

Fig. 2 The relation between remaining energy and the number of rounds

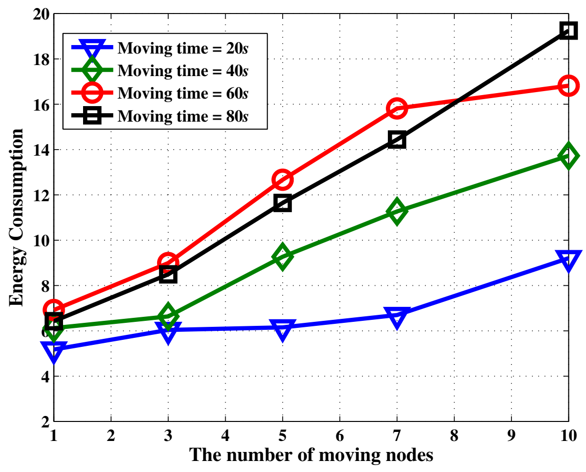

Fig. 3 The relation between energy consumption and the number of moving nodes

This suggests that it is desirable to have a significantly less number of neighbor nodes and a lower energy consumption if we select a small moving time. For the red solid line, when the number of moving nodes is less than 8, the energy consumption of 60 s moving time is larger than that of 80 s. This shows that the number of neighbors may diminish when the moving distance is larger than the optimum, thus the latter costs less energy consumption. This gives an insight into we can select properly the moving time and the less moving time may result in more energy consumption.

To understand the relationship between the number of moving nodes and execution rounds of Algorithm 1, we show in Fig. 4 how the execution rounds vary with different number of moving nodes where moving time is 80 s and other parameters are set as before. It is can be seen that the execution rounds decrease with the number of moving nodes increasing and drop dramatically for larger number of moving nodes. But the difference between execution rounds of 5 moving nodes and 7 moving nodes is very small, this phenomenon is similar to Fig. 3.

In order to understand the impact of moving time and the number of moving nodes on energy consumption, their relationship is shown in Fig. 5. Figure 5 shows that, on the one hand, when the number of moving nodes is less than 4, energy consumption is small for any moving time, and when the moving time is less than 40 s, energy consumption has similar result. Energy consumption increases significantly, when the moving time is about 60 s and the number of moving nodes is about 6, but it drops when the number of moving nodes decreases or increases. Similarly, when the moving time is larger than 80 s, energy consumption shows an descending trend, as shown in the middle position of Fig. 5. This shows that

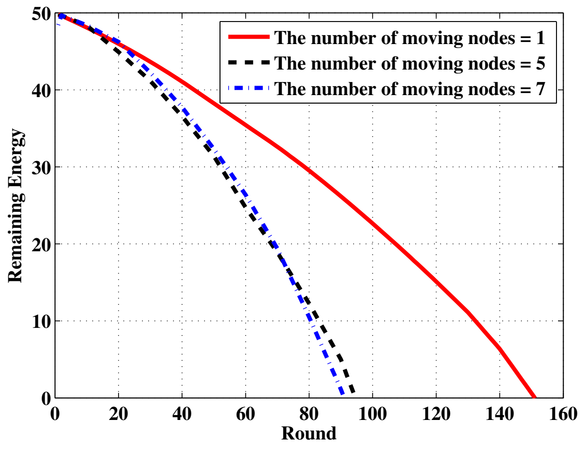

Fig. 4 The relation between remaining energy and the number of rounds

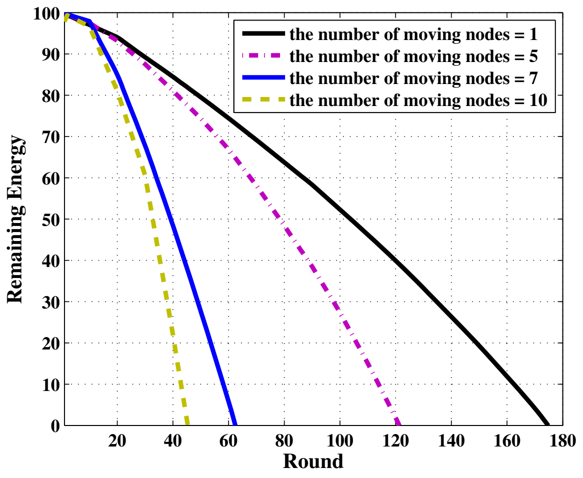

Fig. 6 The relation between remaining energy and the number of rounds

nodes which have 60 s moving time can move to the area which has a larger node density than that of other different moving times. When the moving time is 80 s and the number of moving nodes is 10, energy consumption has the maximum, since the larger number of moving nodes can result in more energy consumption.

Considering the impact of the number of nodes on execution rounds again for another node size in Fig. 6. The simulation is done with the number of nodes being 200, moving time being 80 s, other parameters are set before. Compared with Fig. 4, when the number of moving nodes is 7, execution rounds of Algorithm 1 decreases 31.1%, and execution rounds for 10 moving nodes is less than that of 7 moving nodes. This shows that the node density is not direct proportion to the execution rounds of Algorithm 1. That is, increasing node density can improve the number of execution rounds for less moving nodes, an inverted result exists for larger number of moving nodes. The main reason is that the nodes have more neighbors for larger node density and then cost more energy, resulting in less execution rounds.

Considering the impact of the number of nodes and the number of moving nodes on energy consumption, the simulation is done with the moving time being 80 *s*, execution rounds being 30. In Fig. 7, it is noticed that when the number of moving nodes is less than 9, energy consumption increases with the number of nodes increasing for the same number of moving nodes, since nodes have more neighbor nodes for larger node density. Each non-leader node needs to transmit data to its leader and leaders also receive more data, thus energy consumption boosts. When the number of moving nodes is larger than 9 and the number of nodes is larger than 150, energy

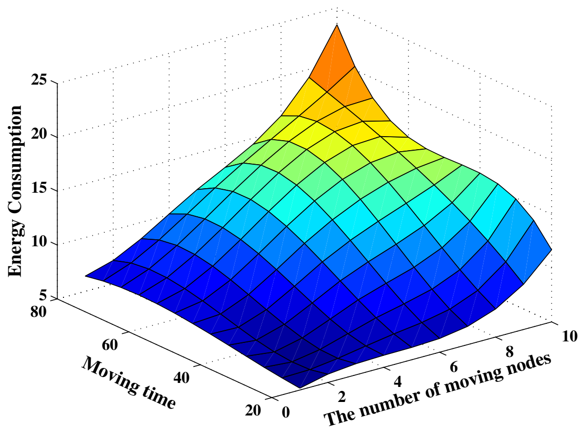

Fig. 5 The relation between energy consumption and moving time, the number of moving nodes

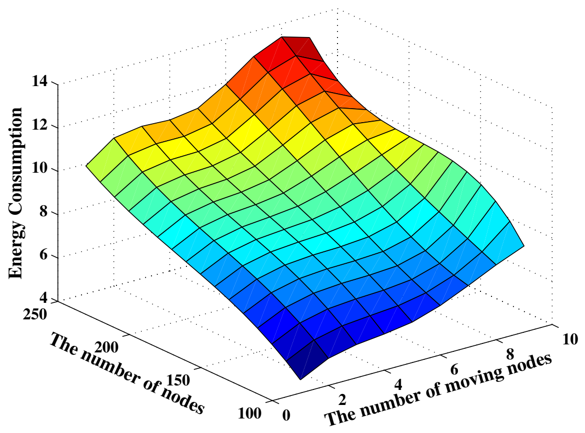

Fig. 7 The relation between energy consumption and the number of nodes, the number of moving nodes

consumption decreases with node density increasing, this is because that although the number of moving nodes is larger, those nodes move to the area which has less nodes and thus each node has less neighbors. Under the same number of moving nodes, with the number of nodes increasing (from 100 to 250), the average roses on energy consumption are 115.2, 109.4, 62.3, 85.5, and 62.5%, respectively.

Compared with Fig. 5, energy consumption presents a rising trend with different number of nodes, but energy consumption has the maximum when the moving time is 60 s.

To understand the performance of Algorithm 2, we consider the impact of the number of moving nodes and the number of nodes on energy consumption, as shown in Fig. 8. It can be seen that energy consumption increases with the number of nodes increasing. Under different node densities, the average roses on energy consumption are 82.8, 56.7, 26.2, and 49.3% when the number of moving nodes is 1, 3, 5, 7, and 10, respectively. Compared with Fig. 7, the main difference includes two parts. One is that energy consumption obtained by Algorithm 2 is less than that of Algorithm 1, which shows the saving energy mechanism is effective, when the number of nodes is set to 200, the average roses on energy consumption descend 19.7, 27.5, 22.3, 23.8, and 23.4%. Another is that although energy consumption decreases, with the number of nodes increasing (from 100 to 250), the average roses on energy consumption for the same number of moving nodes are 152.6, 156.5, 110.4, 115.1, and 80.2%, respectively.

In the following, we valuate the performance of Algorithm 3, as shown in Fig. 9. Similar to Figs. 7 and 8, energy consumption increases with the number of nodes increasing. Compared with Figs. 7 and 8, the whole plane is smooth, which shows that under the same number of nodes, the number of moving nodes has less effect on energy consumption. And each node has similar energy consumption, thus some leaders and non-leaders can not *die* because of costing more energy by receiving data or sending data. Moreover, it is noticed that all energy consumption is less than those obtained by Algorithm 1, when the number of node is 100, the average roses on energy consumption for different number of moving nodes are 30.4, 43.8, 45.8, 56.2, and 61.6%, respectively. And compared with energy consumption obtained by Algorithm 2, the average roses on energy consumption are 20, 28.1, 39.6, 49.7, and 57.4%, respectively. In Algorithm 3, for the same number of moving nodes, with the number of nodes increasing, the average roses on energy consumption rise 6.8, 11.6, 7.1, 6.8, and 1.8%, the range is less than those of Algorithms 1 and 2. Therefore, the saving energy scheme in Algorithm 3 not only saves more energy, but ensures that each node has similar energy consumption, improving network performance.

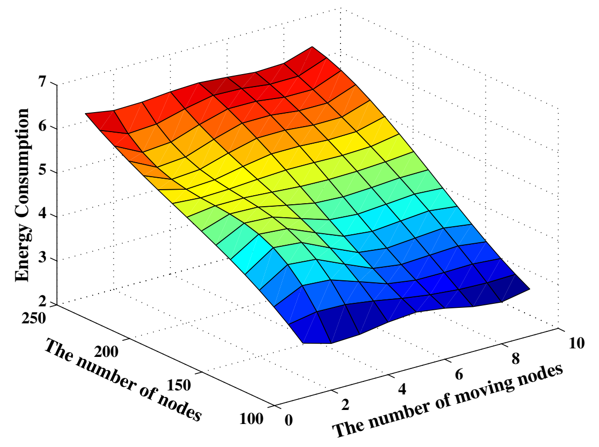

Fig. 9 The relation between energy consumption and the number of nodes, the number of moving nodes

In Fig. 10, the simulation is done with the number of moving nodes being 1, moving time being 80 s, execution rounds being 100, we consider the performance of the proposed algorithms. We can see that Algorithm 3 is more better to save energy than Algorithms 1 and 2. Furthermore, Fig. 10 also reveals the results in Figs. 7, 8, and 9 and the advantages of Algorithm 3.

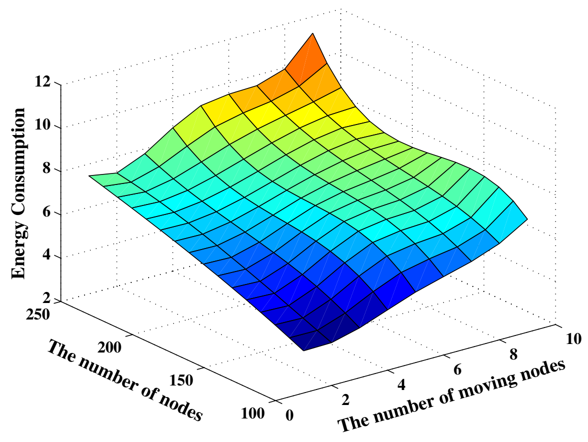

Fig. 8 The relation between energy consumption and the number of nodes and the number of moving nodes

7 Conclusions

In this paper, we propose a new approach to partition the network based on Voronoi, and then run the leader election algorithm in each cell. We consider the fault-tolerant problem, that is, the algorithms of VLE + NMDLE + PSMLE. The innovation points are that we use Voronoi diagram to partition the network, consider the node energy consumption model, and calculate the optimal

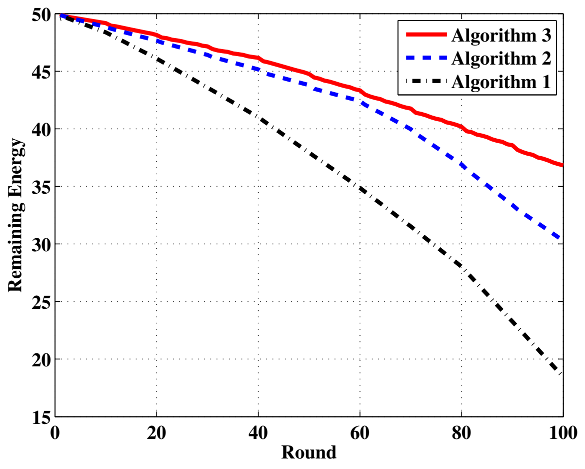

Fig. 10 The relation between remaining energy and the number of rounds

number of partition. Thus, in any network we are able to compute the optimal number of partitions, which will not result in wasting resources and can save energy consumption of the node. On the process of fault tolerance, we choose the current active nodes whose *ID* is largest locally and the remaining energy satisfies minimum energy constraint. Finally, we reduce the energy consumption of the communication module and prolong the network lifetime based on sleeping scheme of nodes. In future work, we will focus on more effective sleeping scheme to reduce energy cost.

Acknowledgements
This work is supported by NSF of China under Grants 61373027 and 61672321, GIF of Shandong University of Science and Technology under Grants SDKDYC170107 and SDKDYC170345, and Macao Science and Technology Development Fund under Grant No. 005/2016/A1.

Authors' contributions
KY, MG, and GL designed and analyzed the algorithms; KY, HJ and GL designed and performed the simulations. KY, MG, HJ, and GL wrote the paper. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Author details
[1]College of Computer Science and Engineering, Shandong University of Science and Technology, 266000 Qingdao, Shandong, China. [2]School of Information Science and Engineering Qufu Normal University, 276826 Rizhao, Shandong, China.

Received: 14 June 2017 Accepted: 15 October 2017
Published online: 13 November 2017

References
1. J Li, S Cheng, Z Cai, J Yu, C Wang, Y Li, Approximate holistic aggregation in wireless sensor networks. ACM Trans. Sens. Netw. **13**(2), 1–24 (2017)
2. A Mainwaring, D Culler, J Polastre, R Szewczyk, J Anderson, in *Proc. WSNA'02 workshop*. Wireless sensor networks for habitat monitoring, (2002), pp. 88–97
3. J Yu, B Huang, X Cheng, M Atiquzzaman, Shortest link scheduling algorithms in wireless networks under the sinr model. IEEE Trans. Veh. Technol. **66**(3), 2643–2657 (2017)
4. Z He, Z Cai, S Cheng, X Wang, Approximate aggregation for tracking quantiles and range countings in wireless sensor networks. Theor. Comput. Sci. **607**(3), 381–390 (2015)
5. S Cheng, Z Cai, J Li, H Gao, Extracting kernel dataset from big sensory data in wireless sensor networks. IEEE Trans. Knowl. Data Eng. **29**(4), 813–827 (2017)
6. R Rana, C Chou, S Kanhere, N Bulusu, W Hu, in *Proc. IEEE IPSN*. Ear-phone: an end-to-end participatory urban noise mapping system, (2010), pp. 105–116
7. D Work, O Tossavainen, Q Jacobson, A Bayen, in *Proc. ACC*. Lagrangian sensing: traffic estimation with mobile devices, (2009), pp. 1536–1543
8. Z He, Z Cai, J Yu, X Wang, Y Sun, Y Li, Cost-efficient strategies for restraining rumor spreading in mobile social networks. IEEE Trans. Veh. Technol. **66**(3), 2789–2900 (2017)
9. H Attiya, J Welch, *Distributed computing: fundamentals, simulations and advance topics*. (Wiley, 2004). ISBN:0471453242
10. H Chung, P Robinson, J Welch, in *Proc. FOMC'10 Workshop*. Regional consecutive leader election in mobile ad-hoc networks, (2010), pp. 81–90
11. H Chung, P Robinson, J Welch, in *Proc. ACM SIGACT/SIGMOBILE FOMC'11 Workshop*. Optimal regional consecutive leader election in mobile ad-hoc networks, (2011), pp. 52–61
12. N Malpani, J Welch, N Vaidya, in *Proc. of the 4th international workshop on Discrete Algorithms and methods for mobile Computing and Communications*. Leader election algorithms for mobile ad hoc networks, (2000), pp. 96–104
13. R Ingram, T Radeva, J Walter, J Welch, in *Proc. SPDP*. An asynchronous leader election algorithm for dynamic networks, (2009), pp. 1–12
14. K Hatzis, G Pentaris, P Spirakis, V Tampakas, R Tan, in *Proc. ACM SPAA*. Fundamental control algorithms in mobile networks, (2009), pp. 251–260
15. P Parvathipuram, V Kumar, G Yang, in *Proc. ICDCIT*. An efficient leader election algorithm for mobile ad hoc networks, (2004), pp. 32–41
16. S Vasudevan, J Kurose, D Towsley, in *Proc. IEEE ICNP*. Design and analysis of a leader election algorithm for mobile ad hoc networks, (2004), pp. 350–360
17. A Boukerche, K Abrougui, in *Proc. IWCMC*. An efficient leader election protocol for mobile networks, (2006), pp. 1129–1134
18. F Kuhn, N Lynch, C Newport, The abstract mac layer. Distrib. Comput. **24**(3), 187–206 (2011)
19. F Kuhn, R Oshman, Dynamic networks: models and algorithms. ACM SIGACT News. **42**(1), 82–96 (2011)
20. J Augustine, G Pandurangan, P Robinson, T Amitabh, in *Proc. ACM-SIAM SDA*. Towards robust and efficient computation in dynamic peer-to-peer networks, (2012), pp. 551–569
21. E Korach, S Moran, S Zaks, in *Proc. ACM SPDC*. Tight lower and upper bounds for some distributed algorithms for a complete network of processors, (1984), pp. 199–207
22. Y Afek, E Gafni, in *Proc. ACM SPDC*. Time and message bounds for election in synchronous and asynchronous complete networks, (1985), pp. 186–195
23. S Kutten, G Pandurangan, D Peleg, A Trehan, in *Proc. ACM SPDC*. On the complexity of universal leader election, (2013), pp. 100–109
24. S Kutten, G Pandurangan, D Peleg, A Trehan, in *Proc. ICDCN*. Sublinear bounds for randomized leader election, (2013), pp. 348–362
25. M Dinitz, J Fineman, S Gilbert, N Calvin, in *Proc. ISDC*. Smoothed analysis of dynamic networks, (2015), pp. 513–527
26. J Yu, N Wang, G Wang, Constructing minimum extended weakly-connected dominating sets for clustering in ad hoc networks. J. Parallel Distrib. Comput. **72**(1), 35–47 (2012)
27. J Yu, N Wang, G Wang, D Yu, Connected dominating sets in wireless ad hoc and sensor networks-a comprehensive survey. Comput. Commun. **36**(2), 121–134 (2013)
28. J Yu, Q Zhang, D Yu, C Chen, G Wang, Domatic partition in homogeneous wireless sensor networks. J. Netw. Comput. Appl. **37**, 186–193 (2014)
29. X Gu, J Yu, D Yu, G Wang, Y Lv, Ecdc: An energy and coverage-aware distributed clustering protocol for wireless sensor networks. Comput. Electr. Eng. **40**(2), 384–398 (2014)

30. J Yu, Y Qi, G Wang, Q Guo, X Gu, An energy-aware distributed unequal clustering protocol for wireless sensor networks. Int. J. Distrib. Sensor Netw. **2011**, 1–8 (2011)
31. S Foss, S Zuyev, On a Voronoi aggregative process related to a bivariate poisson process. Adv. Appl. Probab Univ. Sheffield. **28**(4), 965–981 (1996)
32. J Yu, Y Qi, G Wang, A cluster-based routing protocol for wireless sensor networks with nonuniform node distribution. AEU-Int. J. Electron. Commun. **66**(1), 54–61 (2012)
33. J Yu, L Feng, L Jia, X Gu, D Yu, A local energy consumption prediction-based clustering protocol for wireless sensor networks. Sensors. **14**(12), 23017–23040 (2014)
34. T Camp, J Boleng, V Davies, A survey of mobility models for ad hoc network research. Commun. Mob. Comput. **2**(5), 483–502 (2002)
35. W Navidi, T Camp, Stationary distributiongs for the random waypoint mobility model. IEEE Trans. Mob. Comput. **3**(1), 99–108 (2004)