Wireless Personal Communications (2021) 120:3277–3292
https://doi.org/10.1007/s11277-021-08613-x

k-Connected Relay Node Deployment in Heterogeneous Wireless Sensor Networks

Hemmat Sheikhi[1] · **Mohamad Hoseini**[2] · **Masoud Sabaei**[3]

Accepted: 9 June 2021 / Published online: 20 June 2021

Abstract
Wireless sensor networks (WSNs) are mostly implemented in outdoor and harsh environments; consequently, the probability of node failure is high and applying the fault tolerance mechanisms becomes crucial. The relay nodes placement is a mechanism in order to increase the connectivity level of WSNs. In this paper, a new method (KCN) is proposed to create *k*-connected WSNs based on injection of relay nodes in heterogeneous WSNs, where sensor nodes have different transmission radii. KCN consists of three algorithms. Algorithm 1 creates a minimum *k*-vertex connected graph based on graph theory and Algorithm 2 determines the candidate relay nodes and their locations for implementing each edge in the obtained graph. Algorithm 3 minimizes the deployed relay node count based on two new proposed lemmas. The main issues of concern here are the time complexity and the deployed relay node count. KCN has a low time complexity of $O(n^2)$, where n is the sensor node count. The simulation results demonstrate that KCN applies less relay nodes than the most closely related approach.

Keywords Wireless sensor networks · Relay node placement · *k*-vertex connected graph · *k*-connected network · Transmission radius

1 Introduction

Wireless sensor networks (WSNs) are applied to sense and monitor an environment. Sensor nodes are small devices with the ability to monitor an environment for various objectives. The transmission radius of sensor nodes is restricted, and therefore, they

✉ Hemmat Sheikhi
h.sheikhi@kut.ac.ir

Mohamad Hoseini
mhoseini@mpi-inf.mpg.de

Masoud Sabaei
sabaei@aut.ac.ir

[1] Faculty of Information Technology, Kermanshah University of Technology, Kermanshah, Iran

[2] Max Planck Institute for Informatics, Saarbrücken, Germany

[3] Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran

should be connected over a multi-hop network [1]. The inherent limitations of sensor nodes prevent WSNs functionality and this necessitates application of, therefore, topology management techniques therein. Several topology management techniques have been proposed for tolerating node failures in WSNs like node discovery, sleep scheduling, clustering, power saving, multi-path routing, etc. [2]. Multi-path routing can be facilitated through the relay node placement in WSNs.

In this paper, the proposed approach is based on the relay node placement to provide multi-path routing and fault tolerance with higher network connectivity in heterogeneous WSNs. A WSN can tolerate the failure of up to *k-1* nodes if and only if it contains at least k internally disjoint paths between every sensor pair. The relay nodes would applied to assure that there exist at least *k* vertex-disjoint paths between all sensor node pairs because the initial WSN may not contain enough paths. The relay nodes only receive/forward data from/to other nodes and cannot sense the environment. The problem of finding the minimum relay node count to establish k vertex-disjoint paths is a nondeterministic polynomial (NP)-hard problem [3].

This paper addresses relay node placement in the context of heterogeneous WSNs in order to reach more practical results. In this type of WSN, the sensor nodes have different transmission radii, while all of the relay nodes use an identical transmission radius [4]. The focus of this study is on creating a fault-tolerant heterogeneous WSN for suffering the failure of *k-1* sensor nodes by placing the least relay node count. To reach this aim a method (KCN) is proposed to establish k vertex-disjoint paths between all sensor node pairs.

The proposed method consists of two phases. The first phase is creating a *k*-vertex connected graph with the minimum length of edges. In [5] Chartrand et al. noted that "let G be a *k*-connected graph and *S* be any set of *k* vertices. If a graph *H* is obtained from *G* by adding a new vertex *w* and joining *w* to the vertices of *S*, then *H* is also *k*-connected". Algorithm 1 is developed to create a *k*-vertex connected graph with the minimum length of edges based on this lemma. The second phase is reducing the required relay node count for creating the *k*-connected network. In this phase, two algorithms are developed to implement the edges that are obtained in phase 1 with the least relay node count. Algorithm 2 determines the candidate relay nodes and their locations for implementing each edge. Algorithm 3 decides on whether to disregard a candidate relay node based on some heuristics. During making the decision on each candidate relay node, the possibility of playing its role through the sensor nodes and the previous placed relay nodes is investigated. The WSN remains *k*-connected, even if the algorithm decides to disregard all candidate relay nodes.

The contributions of this paper are summarized as follows:

- According to the mentioned lemma in [5], Algorithm 1 is presented to create a minimum *k*-vertex connected graph.
- An interesting technique is developed to reduce the time complexity of Algorithm 1 from $O(n^3)$ to $O(n^2)$.
- Algorithm 2 is developed to locate a set of candidate relay nodes for implementing each edge in the *k*-vertex connected graph.
- Lemmas 1 and 2 and Sub-lemma 1.1 with their proofs are stated to reduce the required relay node count.
- Algorithm 3, which uses Algorithms 1 and 2 and Lemmas 1 and 2, is developed to create *k*-connected WSNs with placing the least relay node count.

The rest of this paper is organized as follows: literature is reviewed in Sect. 2; the method is proposed in Sect. 3; the method performance is evaluated in Sect. 4 and the article is concluded in Sect. 5.

2 Literature Review

The problem of relay node placement in order to address fault tolerant WSNs is assessed by many prior studies with various views and under different situations in the networks [6]. In each study, a specific problem is addressed and some special assumptions are made for the networks are considered and a heuristic or meta-heuristic scheme is proposed to solve the problem in the given network. Some aspects of concern are: node structure (homogenous/heterogeneous), node location (constrained/unconstrained) and connectivity type (between all node pairs/between each node and the sink).

Sitanayah et al. [7] exploited a WSN, where all sensor nodes send/receive data through k disjoint paths with limited length to/from a set of sinks. Their proposed method consists of two phases. In the first phase, the k shortest paths between each sensor node and all of the sinks are identified through a heuristic method if possible and in the second phase, the relay nodes are added to the network applying a greedy algorithm in order to set paths are not found in the previous phase.

Nitesh et al. [8] introduced an algorithm that applies Jarvis March approach for relay node placement, where k relay nodes are placed in transmission radius of each sensor node (k-coverage) and there are s relay nodes with at least one path to the sink in transmission radius of each relay node (s-connectivity).

In some cases, the WSN divided into several separated sectors because of nodes failures. The key issue of concern here is to connect these separated sectors through the relay node placement. In the study of [9], relay nodes are placed to set a MST of sectors, and furthermore, maximize the connectivity degree of sectors with the finite relay node count. They implemented this method over grid and non-grid WSN.

WSNs consisting of the nodes with the same transmission radii are named homogeneous-WSN while WSNs are heterogeneous when the nodes have different transmission radii. The different transmission radii of the nodes lead to the asymmetric communication links between adjacent nodes in the network. There are two kinds of connection between nodes as follows: two-way connection and one-way connection. In the two-way connection, data transmission is carried out in both directions, while in the one-way connection; there only exists one direction transmission.

Han et al. [4] proposed connectivity-first algorithm (CFA) for addressing four problems in heterogeneous WSNs as follows: One-way partial fault-tolerant relay node placement, Two-way partial fault-tolerant relay node placement, One-way full fault-tolerant relay node placement, and Two-way full fault-tolerant relay node placement. In their study, the full fault-tolerant network is introduced as a WSN, where there exist k disjoint paths between every sensor and/or relay node pair while in the partial fault-tolerant WSN, k disjoint paths are only between every all sensor node pairs regardless of relay nodes. They apply a σ-approximation algorithm to generate a minimum k vertex connected spanning graph through placing the least relay node count.

Some methods based on the evolutionary algorithms like genetic, bee colony, biogeography based, ant colony and imperialist competitive algorithm are introduced to solve the k connectivity problem in WSNs [10–16]. These methods require much time to solve the

problem due to some reasons such as a prolonged time in exploring the problem space, generating the initial population and improving them.

3 The Proposed Method

In this section, KCN the proposed method for creating the *k*-connected heterogeneous WSNs is designed. The work model, notations and terms are presented in Sect. 3.1 and the proposed algorithms are described in Sects. 3.2, 3.3 and 3.4.

3.1 Work Model and Assumptions

The initial heterogeneous WSN only includes the sensor nodes with different transmission radii. These sensor nodes are scatter in a random manner on a two-dimensional are of interest. The problem is creating a fault-tolerant WSN by establishing k ($>=1$) vertex-disjoint paths between all sensor node pairs through deploying the least relay node count. A WSN can be modelled as a graph *G(V,E)*, where *V* is a set of vertices and *E* is a set of edges. The notations, terms and their semantics used in proposed algorithm are listed in Table 1.

3.2 The Minimum *k*-Vertex Connected Graph

The algorithm for creating the *k*-vertex connected graphs based on finding the shortest edges is developed here. According to the mentioned lemma in [5], a new method is proposed as a pseudo code in Algorithm 1.

Algorithm 1 is illustrated by an example in Fig. 1. There are 5 sensor nodes in S as depicted in Fig. 1a and the goal is to generate a minimum 3-vertex connected graph ($k=3$).

Table 1 Notations and semantics

Notations	Semantics
S	The set of sensor nodes
S_i	The sensor node i
n	The sensor node count
V	The set of graph vertices
E	The set of graph edges
E_{ij}	The edge between S_i and S_j
$\|E_{ij}\|$	The length of E_{ij}
T_i	The transmission radius of S_i
T_r	The transmission radius of all relay nodes
$\|\|X\|\|$	The cardinality of set X
k	Degree of connectivity
CR	The set of candidate relay nodes
R	The set of relay nodes

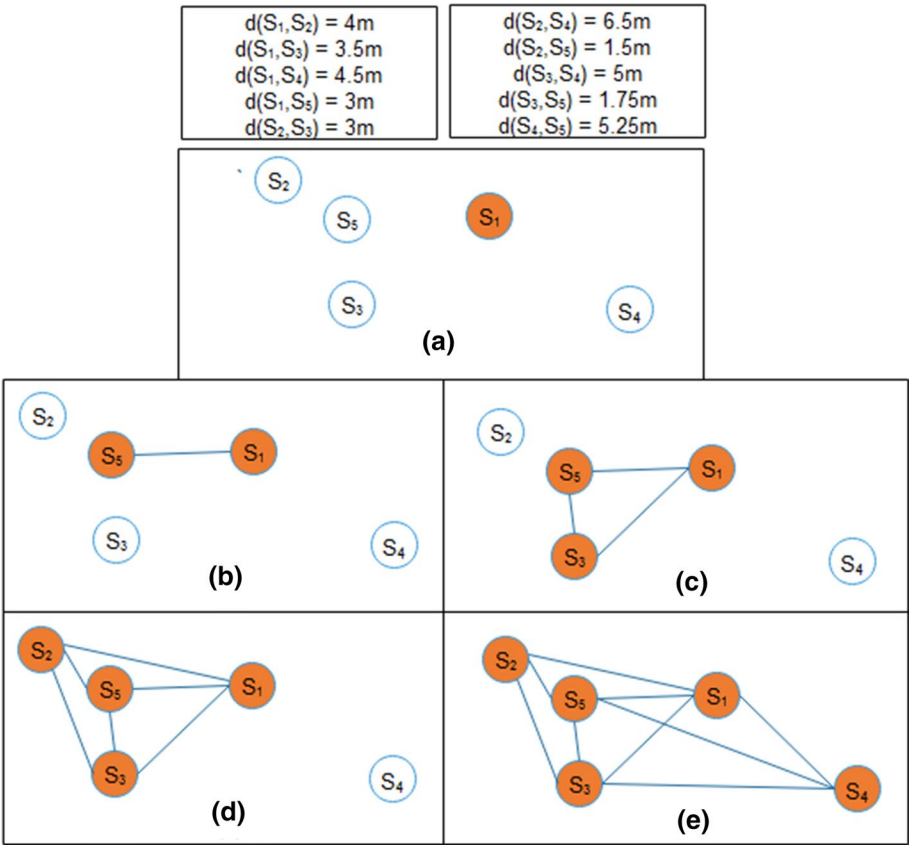

Fig. 1 An example of running Algorithm 1

The distances between all nodes in the graph are shown in Fig. 1a. S_1 is selected as the first node and added to V. Note that it is not important which sensor is the first node and any other sensor can be selected as the first sensor. In the first iteration of the example, m is 1, S_5 is selected to be added to V and E_{15} is added to E as shown in Fig. 1b. In the second iteration, m is 2, S_3 is added to V and two edges E_{31} and E_{35} are added to E (Fig. 1c). In the third iteration, m is equal to 3, S_2 is selected and added to V and three edges E_{21}, E_{23} and E_{25} are added to E (Fig. 1d). In the last iteration, m is equal to 3, S_4 is added to V and three edges E_{42}, E_{43} and E_{45} are add to E (Fig. 1e).

3.3 Determining the Candidate Relay Nodes

In this section, an algorithm is presented to locate a set of candidate relay nodes for implementing each edge in the k-vertex connected graph. There are three cases as follows:

1. $|E_{ij}|$ *is shorter than both transmission radius of* S_i *and* S_j: A two-way (duplex) connection is established currently and no need to deploy the redundant relay nodes.

2. $|E_{ij}|$ *is between the transmission radii of* S_i *and* S_j: A one-way (simplex) connection is established currently and relay node placement is required in order to establish the opposite (direction) connection.
3. $|E_{ij}|$ *is longer than both transmission radii of* S_i *and* S_j: There is no connection between two sensor nodes and relay node placement is required for establishing connection in both directions.

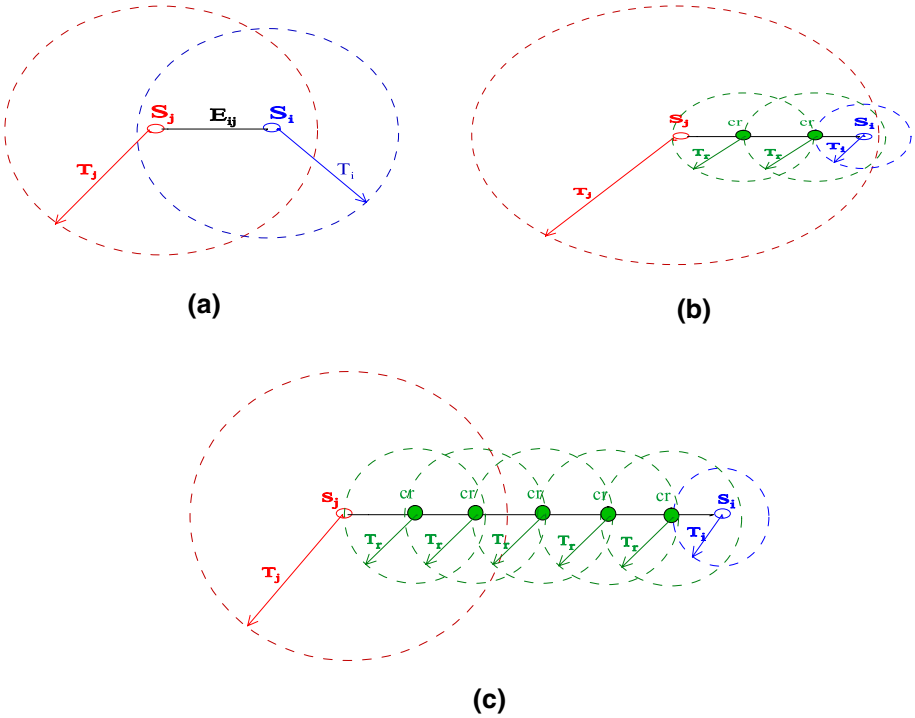

Fig. 2 The candidate relay nodes for implementing an edge (*cr* indicates a candidate relay node)

A pseudo code for locating the candidate relay nodes for each edge in the k-vertex connected graph is illustrated in Algorithm 2. To better understand Algorithm 2, three examples are presented in Fig. 2. As is shown in Fig. 2a, there is not any need to deploy a candidate relay node when distance between S_i and S_j is less than both T_i and T_j. Figure 2b, c show the required candidate relay nodes for cases 2 and 3, respectively. In Fig. 2, *cr* indicates a candidate relay node.

Algorithm 1: The algorithm for creating a minimum k-vertex connected graph

Input: the set of $\boldsymbol{n}$ sensor nodes $\boldsymbol{S}=\{S_1, S_2, \ldots, S_n\}$ and the integer $\boldsymbol{k}$ representing the degree of connectivity

Output: the *k*-vertex connected graph *G(V, E)*

1: $E \longleftarrow \emptyset$, $V \longleftarrow \emptyset$, $m = 1$, define a matrix *W[1..n][1..n]* where *W[i][j]* is the distance between S_i and S_j ;
2: add S_1 to *V*; //as the first node
3: define a matrix *nearest[2..n][1..k]* and an array *distances[2..n]*;
4: **for** (*i: 2...n*) **do**
5: *nearest[i][1]=1, nearest[i][2..k] = 0, distances[i]= W[i][1]*;
6: **end for**
7: **while** (||*V*|| < ||*S*||) **do**
8: find the node S_{min} in *S-V* where *distances[min]* is the minimum value among other nodes in *S-V*;
9: add the edges between S_{min} and the nodes, that their vertices are in *nearest[min][1..m]*, to *E*;
10: add S_{min} to *V*;
11: *m* = minimum{||*V*|| , *k*};
12: **for all** nodes S_i in *S-V* **do**
13: **for** *j:1..m* **do**
14: **if** (*W[i][min]* < *W[i][nearest[i][j]]*) **then**
15: shift right the elements of *nearest* from *nearest[i][j]* to *nearest[i][m-1]*;
16: *nearest[i][j] = min*;
17: **if** (*m==k)* **then**
18: *distances[i] = distances[i] - W[i][nearest[i][k]]*;
19: **end if**
20: *distances[i] = distances[i] + W[i][min]*;
21: **break**;
22: **end if**
23: **end for**
24: **end for all**
25: **end while**
26: **return** *G(V,E)*;

Algorithm 2: The **a**lgorithm for determining the candidate relay nodes and their locations for a specific edge

Input: the edge E_{ij}.
Output: The set of candidate relay nodes *CR*
1: $CR = \emptyset$;
2: **if** $|E_{ij}| <= \min\{T_i, T_j\}$ **then**
3: go to step 17;
4: **end if**
5: **if** $T_i < T_j$ **then**
6: $min \longleftarrow i$, $max \longleftarrow j$;
7: **else**
8: $\min \longleftarrow$ j, $\max \longleftarrow$ i;
9: **end if**
10: add one candidate relay node to *CR* and call it *cr1*. The location of *cr1* is on the straight line between S_{min} and S_{max} with the distance of $min\{T_{min}, T_r\}$ from S_{min};
11: **if** $|E_{ij}| - min\{T_{min}, T_r\} <= T_{max}$ **then**
12: add $\left\lfloor \frac{|E_{ij}| - min\{T_{min}, T_r\}}{T_r} \right\rfloor$ numbers of Candidate Relay nodes to *CR*. Their locations are on the straight line between *cr1* and S_{max} with identical distances from each other;
13: **else**
14: add one candidate relay node to *CR* and call it *cr2*. The location of c*r2* is on the straight line between S_{max} and *cr1* with the distance of $min\{T_{max}, T_r\}$ from S_{max};
15: add the $\left\lfloor \frac{|E_{ij}| - min\{T_{min}, T_r\} - min\{T_{max}, T_r\}}{T_r} \right\rfloor$ numbers of candidate relay nodes to *CR*. Their locations are on the straight line between *cr1* and *cr2* with equal distances;
16: **end if**
17: **return** *CR*;

3.4 Creating KFTN

The aim of this section is to propose an algorithm to implement a *k*-connected WSN based on the *k*-vertex connected graph the output of Algorithm 1. An edge in a graph can only connect two vertices, however, a relay node in a WSN may connect more than two sensor/relay nodes. The reason behind this fact is that a relay node has a transmission radius and all of the nodes in its transmission radius can send/receive data to/from it. An example of implementing two edges through only applying one relay node is shown in Fig. 3. Four sensor nodes *x, y, z* and *u* and their two corresponding edges E_{xy} and E_{zu} are indicated in Fig. 3a. Both of these edges can be established by one relay node *r* (Fig. 3b). This idea is the basis of the technique applied here to reduce the required relay node count. The terminologies which are used in this section are described below:

K-Fault-Tolerant-Network (KFTN): the WSN that remains connected after *k-1* nodes (sensor or relay) failures.

Sub(CR_t): the set of at least *k* sensor and relay nodes placed previously. The nodes of this set can play the role of the candidate relay node CR_t in setting up the edge between two sensor nodes.

$SCR_{ij} = [S_i, CR_1, CR_2, \ldots, CR_m, S_j]$: the sorted set of required candidate relay nodes to implement the edge between S_i and S_j. The set of candidate relay nodes obtained from Algorithm 2 is sorted based on the polar coordinates of the nodes over a straight line from S_i to S_j. The term *m* is use to indicate the number of candidate relay nodes for setting up the edge E_{ij}. If E_{ij} does not require any relay nodes then $SCR_{ij} = [S_i, S_j]$.

The flowing lemmas applied in Algorithm 3 are introduced as follow:

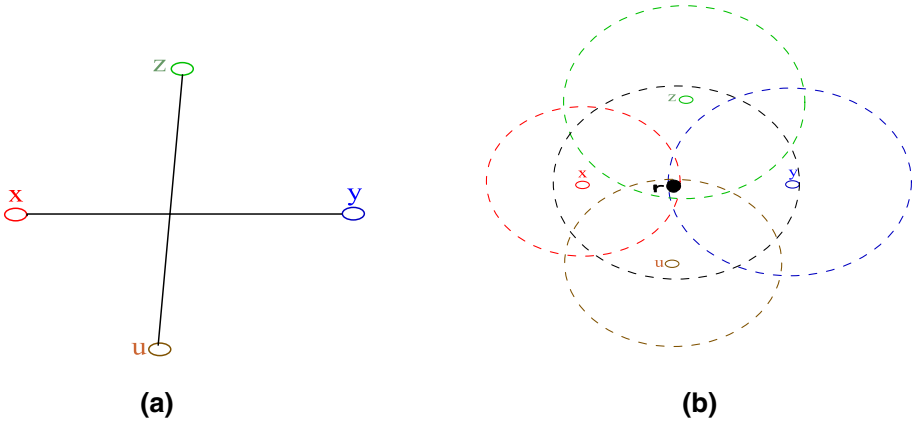

Fig. 3 Applying of one relay node for implementing two edges

Lemma 1 *Let E_{ij} be an edge in a KFTN and CR_t is a candidate relay node over this edge. If there exist at least k placed nodes (sensor or relay) where each of them has a two-way connection with both the previous and the next nodes of CR_t in the set SCR_{ij} then placing the CR_t can be disregarded without disconnecting E_{ij} and losing k-connectivity in the KFTN. These placed nodes are added to the set Sub(CR_t).*

Proof According to the mentioned assumptions $\left|\left| Sub\left(CR_t\right)\right|\right| \geq k$. Removing *k-1* nodes from the set *Sub(CR_t)* cannot disconnect the edge E_{ij}, because at least one node remains in the set *Sub(CR_t)* which connects the previous and the next nodes of CR_t over the edge E_{ij}. These removed nodes in the worst case lead to a *1FTN*. Thus, the network will stay as a *KFTN* even if the placement of CR_t is disregarded.

Lemma 1.1 *The Sub(CR_t) must have at least k placed nodes (sensor or relay).*

Proof Suppose the set *Sub(CR_t)* has *k-1* nodes. Removing all these *k-1* nodes in a *KFTN* can lead to a *1FTN* without respect to E_{ij} in the worst case. Since the connection between the previous node (CR_{t-1} or S_i) and the next node (CR_{t+1} or S_j) of CR_t over the edge E_{ij} is lost, E_{ij} would be disconnected, and consequently, the network is not connected any more. Therefore, substituting CR_t with at most *k-1* nodes does not assure that the network is *KFTN*.

An example of using Lemma 1 to reduce the required relay node count is shown in Fig. 4. Assume that $k=2$ and the edge between S_i and S_j (E_{ij}) exists in E (according to Algorithm 1) and three candidate relay nodes CR_1, CR_2 and CR_3 are required to implement E_{ij} (according to Algorithm 2). The sensor node S_z is one of the initial sensor nodes (set of S) and R_i is the relay node which has been deployed previously (according to Algorithm 3). As are represented in Figs. 4.a, 4.b and 4.c, two nodes S_z and R_i can play the role of CR_2 and connect CR_1 to CR_3, therefore $Sub(CR_2)=\{S_z, R_i\}$ and based on Lemma 1, CR_2 can be disregarded while the network would remain *2FTN*.

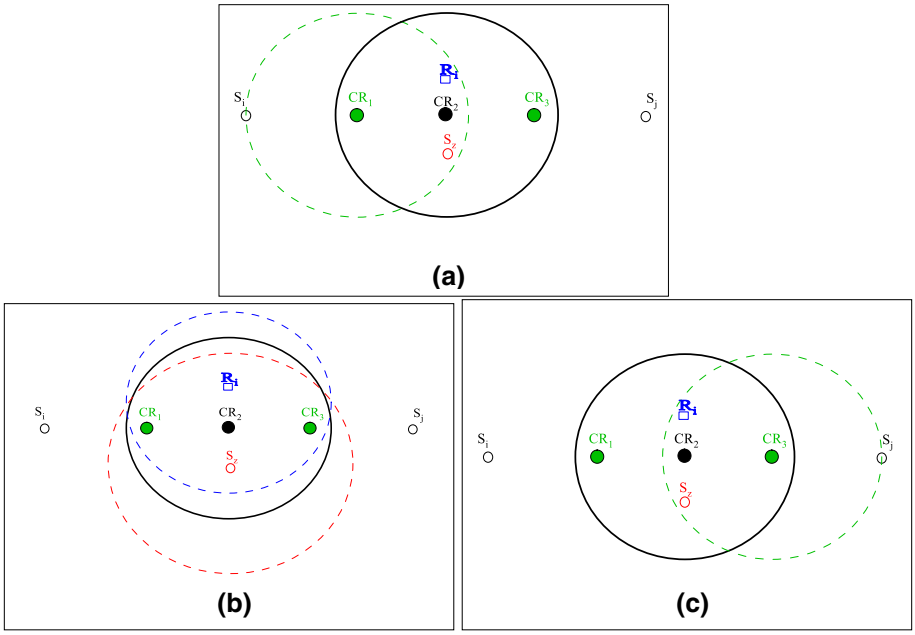

Fig. 4 An example of using Lemma 1

Lemma 2 *Let E_{ij} be an edge in a KFTN and CR_t is the candidate relay node over this edge and CR_{t-1} is disregarded based on Lemma 1. If there exist at least k deployed nodes (sensor or relay) that each one has a two-way connection with the next node of CR_t in the set SCR_{ij} and at least one node from $Sub(CR_{t-1})$ separately, then placing the CR_t can be disregarded without disconnecting E_{ij} and losing k-connectivity in the KFTN. These placed nodes are added to the set $Sub(CR_t)$.*

Proof The initial network is a *KFTN* and $||Sub(CR_t)|| \geq k$. Removing *k-1* nodes from $Sub(CR_t)$ cannot disconnect the edge E_{ij} and at least one node remains that can connect $Sub(CR_{t-1})$ to the next node of CR_t over the edge E_{ij}. In the worst case, the removed nodes lead to a *1FTN*. Thus, the network will be a KFTN after disregarding the placement of CR_t.

Figure 5 shows an example of applying Lemma 2 on the network in Fig. 4 after disregarding CR_2. This case considers how candidate relay node CR_3 would be disregarded. Assume that the sensor node S_t and the relay node R_j have been placed previously. As observed in Figs. 5.a and 5.b, S_t can connect R_i to S_j and vice versa as well as R_j can connect S_z to S_j and vice versa. As a result, $Sub(CR_3)=\{S_t, R_j\}$ and based on Lemma 2, CR_2 can be disregarded while the network would remain *2FTN*.

According to Lemmas 1 and 2 a new algorithm to implement a *KFTN* with placing the least relay node count is proposed as a pseudo code in Algorithm 3.

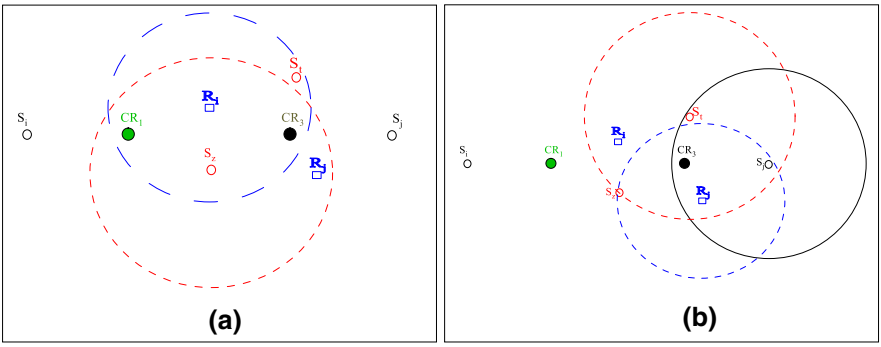

Fig. 5 An example of using Lemma 2

3.5 Time complexity

Algorithm 3 consists of two parts which run sequentially: the first part is line 1 which calls Algorithm 1 and the second part is a while-loop (starting from line 3). The runtime of these two parts are calculated firstly and then the summation of them is introduced as total runtime.

Algorithm 3: The algorithm for implementing a *KFTN*

Input: The set of sensor nodes S and the integer k presenting the degree of connectivity
Output: The set of relay nodes R.

```
1:  Call Algorithm 1 with the set S and get the minimum k-vertex connected graph G(V, E);
2:  R = Ø ;
3:  while (E ≠ Ø) do
4:      select one of the edges in E (assume E_ij);
5:      call Algorithm 2 for E_ij and get the set of candidate relay nodes CR;
6:      sort CR based on the polar coordinates and put all elements of CR into the ordered list SCR_ij
        along with S_i at the beginning and S_j at the end of the list;
7:      for t:1..||CR|| do
8:          Sub(CR_t) = Ø;
9:          if ( Sub(CR_t-1) == Ø OR t == 1 ) then
10:             if the placement of CR_t is disregarded based on Lemma 1 then
11:                 add the nodes with mentioned conditions in Lemma 1 to the set Sub(CR_t);
12:             else
13:                 add CR_t to the set R;
14:             end if
15:         else
16:             if the placement of CR_t is disregarded based on Lemma 2 then
17:                 add the nodes with mentioned conditions in Lemma 2 to the set Sub(CR_t);
18:             else
19:                 add CR_t to the set R;
20:             end if
21:         end if
22:     end for
23:     remove E_ij from E;
24: end while
25: return R;
```

The first part: Algorithm 1 has a while-loop (starting from line 7) which is repeated n (the number of sensor nodes) times because $||S||=n$ and in the beginning $||V||=0$. There is

a for-loop (starting from line 12) into the while-loop. This loop is iterated *n-1* times in the first cycle of the while-loop, *n-2* times in the second cycle and etc. In this for-loop, there is another for-loop (starting from line 13) repeated maximum k times in each cycle. Therefore, the runtime is cculated based on Eqs. (1 and 2).

$$T1(n) = k(n-1) + k(n-2) + \cdots + 2k + k \tag{1}$$

$$T1(n) = \frac{kn^2 - kn}{2} \tag{2}$$

The second part: As mentioned above, the second part of Algorithm 3 is a while-loop (starting from line 3). The number of iterations of this loop is c k-vertex connected graph returned through Algorithm 1 (presented in Eq. (3)).

$$||E|| = \underbrace{1 + 2 + \cdots + (k-1) + \overbrace{k + k + \cdots + k}^{n-k}}_{n-1} \tag{3}$$

Algorithm 2 does not have any loop and returns the candidate relay nodes for each edge. Assume that the number of candidate relay nodes for each edge is t (a constant value). The inner for-loop (line 7) is iterated t times. Therefore, the runtime of the second part is calculated according to Eq. (4).

$$T2(n) = \frac{2kn - k^2 - k}{2} \times t \tag{4}$$

The time complexity of Algorithm 3 is expressed through Eq. (5). Accorngly, the overall time complexity of KCN is $O(n^2)$.

$$T(n) = T1(n) + T2(n) = \frac{kn^2 - kn}{2} + \frac{2kn - k^2 - k}{2} \times t \tag{5}$$

4 Performance Evaluation

The performance of KCN is compared to CFA [4] and simple-KCN by using simulation. To the best of the authors' knowledge here, The CFA is the only method in the literature that creates k separate paths between all sensor pairs in heterogeneous WSNs. Therefore, both KCN and CFA focus on the same problem. The CFA method creates a k-vertex connected graph through adding edges with the highest contribution from the complete graph. The contribution of each edge is defined based on the sensor pairs that disjoint paths between them are fewer than k and the connectivity of them can be improved through this edge. The simple-KCN method creates the k-vertex connected graph and determines the set of candidate relay nodes for all edges in the graph and places all candidate relay nodes without disregarding any of them. MATLAB is used as the simulation environment.

In the simulation experiments, the area of interest size is 1000 m × 1000 m and the sensor nodes are scattered on this area in a random manner. Two types of simulations are considered in this work. In both the types, all sensor nodes have a random transmission

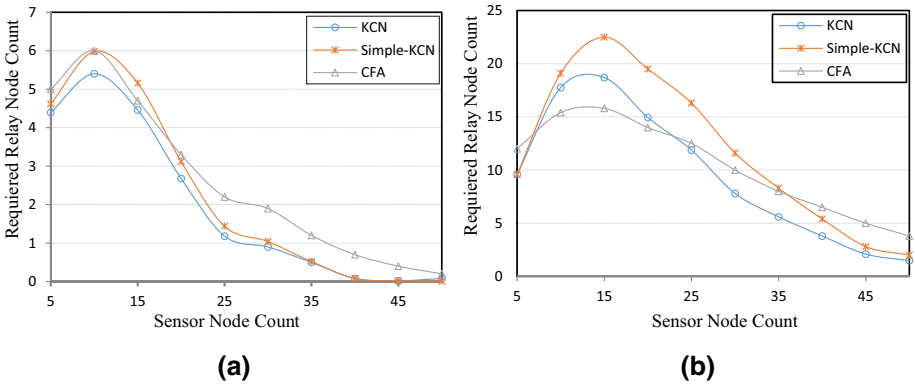

Fig. 6 The required relay node count. **a** $k=2$. **b** $k=4$

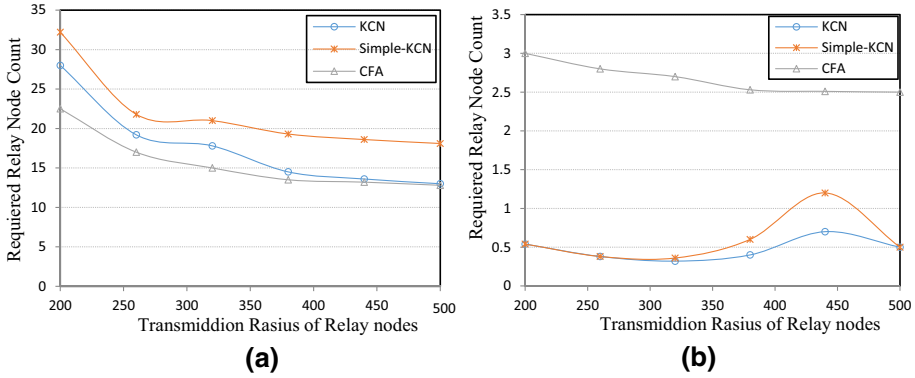

Fig. 7 The required relay node count. **a** $n=20$. **b** $n=60$

radius between 200 and 500 m and the average outputs of 50 runs are considered as the final simulation results.

In the first simulation, the transmission radius of all relay nodes is 350 m. The sensor node count is gradually increased from 5 to 50. The comparison between KCN with CFA and simple-KCN in terms of the required relay node count to assure that the WSN is *k*-connected are diagrammed in Fig. 6. There are some interesting tips. In Fig. 6a ($k=2$), KCN always requires less relay nodes than CFA and arrives at zero in 40, however, CFA in 50. In Fig. 6b ($k=4$), KCN has better results in many times but not all times. When the sensor node count is small or large (in both Fig. 6a, b), the KCN and simple- KCN have the same results. In a sparse WSN, the distances between sensors are too far and it is unlikely to disregard a candidate relay node. Also if the sensor node count is large, the connectivity of the initial network is high, and therefore, the difference of results of both KCN and Simple-KCN is slight.

In the second simulation, the transmission radius of all relay nodes is gradually increased from 200 to 500 m. Figure 7 compares the methods in terms of required relay node count for connectivity level $k=4$. Figure 7a shows the results for 20 sensor nodes and Fig. 7b shows for 60 sensor nodes. As shown, when the sensor node count is 20, CFA has

slightly better results, while for 60 sensors the results of KCN are significantly better than CFA. The reason behind this better result is that KCN tries to use the initial sensor nodes instead of deploying new relay nodes. Thus, the more the number of sensor nodes, the better the performance of KCN.

5 Conclusion

The problem of relay node deployment in heterogeneous WSNs is assessed here, where all sensor pairs must be connected by at least k separate paths. To solve this problem, the KCN method consisting of three heuristic algorithms, is proposed. Algorithm 1 creates the minimum k-vertex connected graph; Algorithm 2 determines the candidate relay nodes and Algorithm 3 minimizes the deployed relay node count. The method applies Lemmas 1 and 2 to decide on whether to disregard a candidate relay node based on some heuristics. To evaluate the performance of the proposed algorithm, two key issues are considered: the time complexity and the required relay node count. As proved, KCN has a low time complexity of $O(n^2)$, where n is the sensor node count. The simulation results confirm that the ability of KCN in reducing the required relay node count is higher than the most closely related approach. The future work will focus on reducing the edge count of the k-vertex connected graph in order to reduce the network cost.

References

1. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). Wireless sensor networks: A survey. *Computer Networks, 38*(4), 393–422.
2. Younis, M., & Senturk, I. F. (2014). Topology management techniques for tolerating node failures in wireless sensor networks: A survey. *Computer Networks, 58*, 254–283.
3. Bari, A., Jaekel, A., & Bandyopadhyay, S. (2007). Optimal placement of relay nodes in two-tiered, fault tolerant sensor networks. In *Proceedings of the the 12th IEEE symposium on computers and communications*, Las Vegas, USA, 159–164.
4. Han, X., Cao, X., Lloyd, E. L., & Shen, C. C. (2010). Fault-tolerant relay node placement in heterogeneous wireless sensor networks. *IEEE Transactions on Mobile Computing, 9*(5), 643–656.
5. Chartrand, G., & Zhang, P. (2012). *A first course in graph theory*. New York: Dover.
6. Younis, M., & Akkaya, K. (2008). Strategies and techniques for node placement in wireless sensor networks: A survey. *AD Hoc Networks, 6*(4), 621–655.
7. Sitanayah, L., Brown, K. N., & Sreenan, C. J. (2014). A fault-tolerant relay placement algorithm for ensuring k vertex-disjoint shortest paths in wireless sensor networks. *Ad Hoc Networks, 23*, 145–162.
8. Nitesh, K., & Jana, P. K. (2018). Relay node placement with assured coverage and connectivity: A Jarvis March approach. *Wireless Personal Communications, 98*(1), 1361–1381.
9. Alturjman, F. M., Hassanein, H. S., Alsalih, W. M., & Ibnkahla, M. (2011). Optimized relay placement for wireless sensor networks federation in environmental applications. *Wireless Communication and Mobile Computing, 11*(12), 1677–1688.
10. Gutierrez, J. M. L., Pulido, J. A. G., & Rodriguez, M. A. V. (2015). A new realistic approach for the relay node placement problem in wireless sensor networks by means of evolutionary computation. *Ad Hoc and Sensor Wireless Networks, 26*, 193–209.
11. Gupta, S. K., Kuila, P., & Jana, P. K. (2016). Genetic algorithm approach for k-coverage and m-connected node placement in target based wireless sensor networks. *Computer and Electrical Engineering, 56*, 544–556.
12. Hashim, H. A., Ayinde, B. O., & Abido, M. A. (2016). Optimal placement of relay nodes in wireless sensor network using artificial bee colony algorithm. *Journal of Network and Computer Applications, 64*, 239–248.
13. Gupta, G. P., & Jha, S. (2019). Biogeography-based optimization scheme for solving the coverage and connected node placement problem for wireless sensor networks. *Wireless Networks, 25*(6), 3167–3177.

14. Qasim, T., Zia, M., Minhas, Q. A., Bhatti, N., Saleem, K., Qasim, T., & Mahmood, H. (2018). An ant colony optimization based approach for minimum cost coverage on 3-D grid in wireless sensor networks. *IEEE Communications Letters, 22*(6), 1140–1143.
15. Sheikhi, H., & Barkhoda, W. (2020). Solving the k-coverage and m-connected problem in wireless sensor networks through the imperialist competitive algorithm. *Journal of Interconnection Networks, 20*(1), 201–205.
16. Barkhoda, W., & Sheikhi, H. (2020). Immigrant imperialist competitive algorithm to solve the multi-constraint node placement problem in target-based wireless sensor networks. *Ad Hoc Networks, 106*, 1–13.

Hemmat Sheikhi is currently a faculty member in the department of Information Technology at Kermanshah University of Technology, Kermanshah, Iran. He received his B.S. degree in Computer Engineering from Isfahan University, Isfahan, Iran in 2006 and M.S. degree in Information Technology Engineering from Amirkabir University of Technology, Tehran, Iran in 2009. His research interests are in the field of computer communications and Wireless Sensor Networks.

Mohamad Hoseini is currently Ph.D. student in Max Planck Institute for Informatics, Saarbrucken, Germany. He received his M.S. degree in Information Technology Engineering from Iran University of Science and Technology, Tehran, Iran in 2012 and B.S. degree in Information Technology Engineering from Shiraz University of Technology, Shiraz, Iran in 2009. His current research interest includes Wireless Sensor Networks and Online Social Networks.

Masoud Sabaei is an associate professor in the Department of Computer Engineering at the Amirkabir University of Technology in Tehran. Dr. Sabaei received his B.Sc. degree from the Esfahan University of Technology, Esfahan, Iran, and his M.Sc. and Ph.D. form Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, all in the field of Computer Engineering in 1992, 1995 and 2000, respectively. His research interests are wireless networks, software defined networks, and telecommunication network management.