On "Movement-Assisted Connectivity Restoration in Wireless Sensor and Actor Networks"

ShiGuang Wang, *Student Member*, *IEEE*, XuFei Mao, *Student Member*, *IEEE*, Shao-Jie Tang, Xiang-Yang Li, *Senior Member*, *IEEE*, JiZhong Zhao, *Member*, *IEEE Computer Society*, and GuoJun Dai

Abstract—In wireless sensor and actor networks (WSANs), a set of static sensor nodes and a set of (mobile) actor nodes form a network that performs distributed sensing and actuation tasks. In [1], Abbasi et al. presented DARA, a Distributed Actor Recovery Algorithm, which restores the connectivity of the interactor network by efficiently relocating some mobile actors when failure of an actor happens. To restore 1 and 2-connectivity of the network, two algorithms are developed in [1]. Their basic idea is to find the smallest set of actors that needs to be repositioned to restore the required level of connectivity, with the objective to minimize the movement overhead of relocation. Here, we show that the algorithms proposed in [1] will not work smoothly in all scenarios as claimed and give counterexamples for some algorithms and theorems proposed in [1]. We then present a general actor relocation problem and propose methods that will work correctly for several subsets of the problems. Specifically, our method does result in an optimum movement strategy with minimum movement overhead for the problems studied in [1].

Index Terms—Connectivity restoration, controlled node mobility, fault tolerance, wireless sensor and actor networks.

1 Introduction

In recent years, we have witnessed a growing research interest and practical applications of wireless sensor networks (WSNs). In a WSN, a group of sensors are deployed in a region to gather information about the physical world. The data could be collected, and then, aggregated up toward some sink nodes for further processing. To further improve the efficiency and effectiveness of wireless sensor networks, actor nodes have been introduced and integrated in the wireless sensor networks to form wireless sensor and actor networks (WSANs). Unlike small wireless sensor nodes, actors are usually resource-rich devices equipped with better processing capabilities, stronger transmission powers, and longer battery life. In WSANs (potentially mobile), actor nodes are deployed to perform appropriate actions upon the environment data collected by the sensors, which allows remote, automated interaction with the environment. Unlike traditional control system, in a WSAN network, an actor typically not only is capable of perform individual tasks (such as deactivating a land mine, extinguishing a fire, and rescuing a trapped survivor) in response to the environment, it is also able to collaborate with other actors in a networked system. Applications of wireless sensor and actor networks thus may include a group of mobile robots that will optimize their actions based on data collected from the network. A WSAN will perceive the environment from multiple disparate viewpoints based on the data gathered by a sensor network. Since the actor nodes will perform certain tasks immediately after collecting data from sensor nodes, the issue of real-time communication is very important in WSANs.

In many applications, the number of sensor nodes deployed in studying a phenomenon may be in the order of hundreds or even thousands [10], [17]. However, such a dense deployment is often not necessary for actor nodes because of the different coverage requirements and physical interaction methods of acting task. In addition, actor nodes are often capable of moving from one place to another place to perform a certain task in response to the changing environment. Hence, in order to provide effective sensing and acting, a distributed local coordination mechanism is necessary among sensors and actors. In most applications, it is often required that the network topology of a WSAN should be a connected graph. Here, each node (an actor or a wireless sensor node) is mapped to a vertex in a graph, and two vertices are connected if the corresponding nodes can communicate directly with each other using wireless channels without relying on relays by other nodes. A network is said to be k-connected if for every pair of nodes, there are k node disjoint paths connecting these two nodes. A k-connected network, for $k \geq 2$, is often said to be $(k-1)$-*fault-tolerant*, for fault tolerant for simplicity. When the

- *S. Wang, X. Mao, S.-J. Tang, and X.-Y. Li are with the Department of Computer Science, Illinois Institute of Technology, 10 West 31st Street, Chicago, IL 60616. E-mail: {swang44, xmao3, stang7}@iit.edu, xli@cs.iit.edu.*
- *J. Zhao is with the Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'An, P.R. China. E-mail: zjz@mail.xjtu.edu.cn.*
- *G. Dai is with the Institute of Computer Application Technology, Hangzhou Dianzi University, Hangzhou, P.R. China. E-mail: daigj@hdu.edu.cn.*

Manuscript received 29 Dec. 2009; revised 14 Jan. 2010; accepted 4 Apr. 2010; published online 18 May 2010.
Recommended for acceptance by I. Stojmenovic.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2009-12-0671.
Digital Object Identifier no. 10.1109/TPDS.2010.102.

network topology is not fault tolerant, an actor failure can cause the loss of multiple interactor communication links and may partition the network if alternate paths among the affected actors are not available. So, for a fault-tolerant network, *biconnectivity* is required. In a biconnected network, the removal of any node from the network will not partition the network into connected components.

In the recent work [1], Abbasi et al. presented DARA, a Distributed Actor Recovery Algorithm, which tries to efficiently restore the connectivity of the interactor network that has been affected by the failure of an actor. To address 1 and 2-connectivity requirements, two variants of the algorithm are developed by them (DARA-1C and DARA-2C). Due to the fact that the energy is limited in WSANs and normally movement of nodes consumes huge energy, compared to communication and computation processes, their basic idea is "*to identify the least set of actors that should be repositioned in order to reestablish a particular level of connectivity*." DARA strives to localize the scope of the recovery process and minimize the movement overhead imposed on the involved actors, although they did not explicitly prove and verify the performances of their algorithms in terms of this metric. Unfortunately, the paper [1] contains some significant discrepancies that affect the correctness of the algorithms. Here, we demonstrate it by presenting some counterexamples to some theorems and algorithms. We also present methods that can fix some of the discrepancies in this paper. We then formally define the connectivity-restoration problem and show that it can be solved in polynomial time if the number of candidate positions of nodes is at most $n + c$ for some constant c, for a network of n nodes. Observe that for all results presented in [1], after an actor node fails, they will move some actor nodes toward the old positions of some other actor nodes. Thus, for a network of n actor nodes, after one actor node failed, the number of candidate positions of actor nodes is still n. Thus, the method proposed by us will result in an optimal solution in the setting [1].

Designing wireless networks that are fault-tolerant, or k-connected, and reconfiguring the network (by moving some nodes or adding new relays) to be k-connected have been studied extensively in the literature. Li et al. [15], [16] showed how many nodes are needed to get a k-connected network with high probability with random node placement and proposed localized topology control method to retain k-connectivity. Yi et al. [18] studied the asymptotic distribution of the number of isolated nodes with Bernoulli failures in a random network.

Several results were proposed in the literature to get a new k-connected network after some node failures, by moving existing alive nodes or adding new nodes. Li and Hou [14] proved that it is NP-hard to add the minimum number of nodes to improve the network connectivity. Almasaeid and Kamal [3] proposed a method with objective of minimizing the number of additional nodes needed to repair the connectivity. Ibrahim et al. [12] also discussed network repair problem using minimum number of additional relays. Ibrahim et al. [11] characterized the network connectivity by Fiedler value, i.e., second smallest eigenvalue of the Laplacian matrix of the network. They used the semidefinite programming approach to find the best locations of relay nodes to improve the connectivity. Given a network, Kashyap et al. [13] proposed a polynomial-time method for placing some additional relay nodes for getting a k-connected graph. The approximation ratio of their method is $2D_{MST}$ for nodes in d dimensions, where D_{MST} is the maximum degree of a minimum degree minimum spanning tree of all nodes. Here, $D_{MST} = 5$ for two-dimensional nodes. Given the set of duty sensors in the plane and the upper bound of the transmission range, Cheng et al. [5] proposed two methods to compute the minimum number of relay sensors such that the induced topology by all sensors is globally connected. The first algorithm has a performance ratio of 3 and the second has a performance ratio of $2\frac{1}{2}$. See [6] for a survey of various methods for constructing minimum cost k-connected geometry networks.

Besides adding extra relay nodes, some approaches try to move the existing nodes to improve the connectivity of networks. Das et al. [7] proposed a seminal localized movement control algorithm for wireless mobile robot networks to establish/restore biconnectivity using p-hop neighbor information. Atay and Bayazit [4] used k-redundancy to check and repair the k-connectivity. Abbasi et al. [1] proposed several methods to move nodes around to improve the network connectivity. They assume that every actor node already has the information about all nodes within 2-hop.

This paper is organized as follows: We review some definitions and algorithms presented in [1] in Section 2, and discuss some of the results of [1] in Section 3. We also present our method that can optimally reallocate nodes with the minimum cost to restore the k-connectivity of the network if it is possible. We conclude the paper in Section 4.

2 Review of Results and Definitions in [1]

In this section, we briefly review the paper [1]. The paper [1] focused on the connectivity reestablishment problem in WSANs caused by single-node failure. In WSANs, there are two kinds of nodes: mobile/static sensor nodes and mobile actor nodes. Mobile actor nodes act as the backbone of a WSAN; all the sensor nodes transmit their data to the nearby actor nodes and actor nodes forward the data to the sink node where the data are processed. Due to an actor node's failure, the connectivity of actor nodes could be destroyed. So, how to reestablish the connectivity after one actor node damaged it naturally comes forth. Note that Abbasi et al. [1] did not consider the coverage problem; only connectivity reestablishment problem was focused.

In [1], the authors proposed two distributed algorithms DARA-1C and DARA-2C to address the 1 and 2-connectivity reestablishment problem, which we found not always valid. Both DARA-1C and DARA-2C are real time and perform cascaded relocation of actor nodes. The input of DARA-1C (respectively, DATA-2C) is the 1-connected (respectively, 2-connected) actor network graph. In DARA-1C, each actor node A_i maintains its 2-hop neighbors list, the set 2-hop-Neighbors(A_i), by using periodic heartbeat messages. It is assumed that all nodes will maintain an updated list of its 2-hop neighbors. Missing heartbeat messages can be used to detect the failure of actor nodes.

Note that not the failure of *any actor* node will destroy the connectivity; only a cut vertex of the network graph of the actor network will destroy it. Here, we call an actor node which is a cut vertex of the network graph simply as a cut vertex. A vertex v in a connected graph G is called *cut vertex* if the removal of v will partition G. However, the paper [1] did not mention how to distinguish cut and noncut vertices.

When a cut vertex A_f fails, its heartbeat message is lost, and thus, its neighbors will discover the failure and a recovery procedure happens. So, it is needed to choose an actor node A_{BC} (called *best candidate* node) and relocate it to restore connectivity. The best candidate A_{BC} is chosen from 1-hop neighbors of A_f, by the following criteria in order: *Least node degree, Closest proximity to the failed actor, and Highest actor ID*. The relocation method is as follows:

1. Node A_{BC} sends a "MOVING" message to its neighbors, in the set Dependents(A_BC,A_f), i.e., those 1-hop neighbors of A_{BC} that are not 1-hop neighbors of A_f, about its movement and the time it will take to reach to the new location.
2. Node A_{BC} moves to the exact location of the failed node A_f.
3. Node A_{BC} broadcasts "RECOVERED" message. Notice that since other nodes are still in their previous positions, this broadcast "RECOVERED" message can only be received by nodes in the same connected component as the node A_{BC} at the position of failed node A_f.

When relocation of A_{BC} causes disconnectivity of actor network, another round of relocation happens using the same method. The detection of network disconnectivity caused by the relocation of A_{BC} is performed as follows:

1. The dependent neighbors (children) of A_{BC} keep waiting until they receive the "RECOVERED" message indicating that the restoration process has been completed.
2. If they did not hear the "RECOVERED" message within the time period of $2T$, where T is the time needed by the node A_{BC} to travel for maximum possible distance r, they assume disconnectivity happens and do the restoration process just as their parent A_{BC} did.
3. Each node will move only once.

Note that different from the original relocation, here, the new best candidate in the children level is chosen among the dependent neighbors of the previous best candidate A_{BC}. In paper [1], the authors also talked about an optimization of DARA-1C for ring-like topologies; however, it does not increase the asymptotic bound of converge time.

For restoring 2-connectivity after a failure of an actor node, the authors proposed DARA-2C. First, the authors gave several definitions of key terms which are recalled as follows:

Definition 1 [1]. *The geometric convex hull [9] is the minimal convex set of points containing a nonempty finite set of points.*

Definition 2 [1]. *A network periphery is the convex hull of the nodes of the network. Since the nodes are placed in a plane, the network periphery is a simple closed polygonal chain, unless the nodes are collinear.*

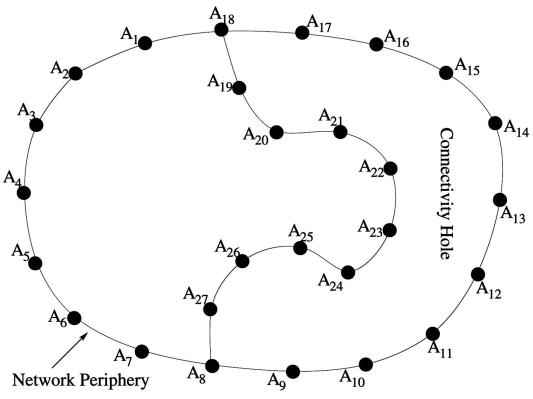


Fig. 1. The convex polygon constructed by vertices $A_1, A_2, \ldots, A_{18}$ is the convex hull of the network graph and is also the network periphery by [1, Definition 2]. By [1, Definition 3], the polygonal area constructed by vertices $A_8, A_9, \ldots, A_{27}$ is the connectivity hole of the network (which is a similar illustration as [1, Fig. 6]). The network boundary node is all the vertex $A_1, A_2, \ldots, A_{27}$, for that they are either located on the network periphery or on the connectivity hole.

Definition 3 [1]. *A connectivity hole [8] is an area in which the edges between nodes form a closed polygonal without links between nodes that are not adjacent on the polygonal chain.*

Definition 4 [1]. *A boundary node is one that is located at the network periphery or on the closed polygonal chain surrounding a connectivity hole.*

The definitions are illustrated in Fig. 1, which is a similar illustration as that in [1]. Observe that their original Definitions 1 (on convex hull), 2 (on network periphery), and 4 (on boundary nodes) are clear and self-contained. However, Definition 3 is not clear and confusing. For special network examples, such as the one illustrated in Fig. 1, the definitions are correct. Later, however, we will show that the definitions of network periphery and connectivity hole are not correct in all scenarios; some counterexamples will be given. Thus, the definition of boundary node is also incorrect. Their original definition of network periphery is not correct, but could be fixed. Unfortunately, we could not find a correct way to fix their original definition of connectivity hole, and consequently, the definition of boundary nodes, to make their algorithms correct.

The following two key theorems are the foundation based on which DARA-2C is proposed, which are recalled as follows:

Theorem 6 [1]. *In a 2-connected network, a failure of a node A_f can cause another node A_C to be a cut vertex only if A_f is a boundary node.*

Theorem 7 [1]. *In a 2-connected network, the biconnectivity lost due to a failure of a node A_f can be restored by repositioning the neighbors of A_f that are located at the network periphery or on the boundary of a connectivity hole.*

In DARA-2C, each actor node knows its 1-hop and 2-hop neighbors as in DARA-1C, which includes the IDs, locations, and node degrees of these neighbors. The paper [1] employs the localized boundary detection algorithm proposed by [19] to determine whether an actor node is a boundary node or

not. We point out here that the traditional network boundary detection method will *not* work for the definition of network boundary proposed in [1]. The reason is that traditional methods typically only detect the periphery of the network. We later will also point out that the definition of periphery [1] is also problematic. Abbasi et al. [1] also employ the cut-vertex detection method proposed by [2], which probabilistically determines whether a node is a cut vertex or not.

Now, we briefly review DARA-2C. As in DARA-1C, the failure actor node is detected by heartbeat message. Based on [1, Theorem 7], the restoration process will be initiated only on every actor node $A_j \in$ Neighbors(A_f), based on the following criteria: 1) A_f is a boundary node, 2) A_j is a boundary node, and 3) the failure of A_f introduces a cut vertex in the network. Nodes satisfy that these conditions are referred to as *candidates* thereafter. DARA-2C restores the 2-connectivity of a network by relocating one of these candidates. Two problems are addressed in DARA-2C as that in DARA-1C: 1) which candidate node A_{BC} (Best Candidate) is chosen to relocate and 2) how to relocate A_{BC}. A_{BC} is chosen by the following three criteria in order: 1) *Lowest node degree*, 2) *Least distance*, and 3) *Highest actor ID*. The relocation of A_{BC} works as follows:

1. If the cardinality of the candidate set is two, the selected A_{BC} will move toward the other candidate until it becomes within its radio range.
2. If the cardinality of the candidate set is three or more, the A_{BC} will move to the position of A_f.
3. Any node will move only once.

Similar to DARA-1C, when A_{BC} stops, it broadcasts a "RECOVERED" message, indicating the completion of the restoration process. The neighbors of A_{BC} will keep waiting for the "RECOVERED" message; if they received it, they conclude that they are still connected; otherwise, they will assume that they got disconnected and apply [1, Theorem 7] again as if A_{BC} stopped functioning. This means that a node $A_j \in \text{Neighbors}(A_{BC})$ will move only if it is a boundary node. The recovery process will be applied recursively to trigger the cascaded relocation of affected actors. Each of these boundary nodes will individually move toward A_{BC} and nonboundary neighbors of A_{BC} do not need to move.

3 Some Discrepancies in [1] and New Method

In this section, we will briefly discuss some discrepancies found in some methods presented in [1]. We will then fix some discrepancies in [1] and later formally propose the connectivity-restoration problem.

3.1 Discrepancies in Algorithm DARA-1C

The basic idea of the algorithm proposed in [1], called DARA-1C, for 1-connectivity case is reviewed in Section 2. We found that there is some problem when apply this algorithm. We describe one network example in Fig. 2. In [1], the authors stated the method to choose a new best candidate in the child level of the old best candidate A_{BC} as "Thus, these detached dependents identify a BC at the children level to relocate to the position of their parent," which means that the new best candidate is chosen from the

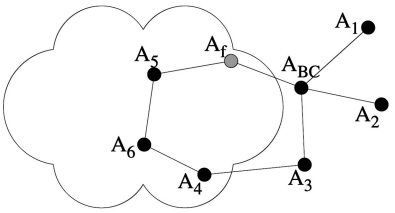


Fig. 2. The dependents set of A_{BC} is $\{A_1, A_2\}$, and the detached dependents set of A_{BC} after A_{BC} relocated is $\{A_1, A_2\}$, but A_1 and A_2 themselves do not know this set. The dependents set, denoted by Dependents(A_i, A_f) [1], is the set of actors that are neighbors of A_i but not neighbors of A_f. The detached dependents set of A_{BC} is the subset of Dependents(A_{BC}, A_f) which becomes not connected to the A_{BC} after A_{BC} relocated to the position of A_f.

set of all the detached dependents of the old best candidate A_{BC}. But *how can each detached dependent node know this detached dependents set?* First, the old best candidate A_{BC} cannot tell which dependent will be detached before its relocation, which is because that A_{BC} only maintains its 2-hop neighbors' information. In Fig. 2, A_{BC} does not know that A_3 is connected with A_f by another path, because that A_6 is not in the 2-hop neighbor table of A_{BC}. Second, the detached dependents could not be connected to the network after A_{BC} relocated and possibly they are not connected with one another either, which means that they *cannot* communicate with each other under some possible situation as A_1 and A_2 in Fig. 2. Thus, this detached dependents set cannot be learned by these detached dependents under some situation, which leads to the case that the selection of a new best candidate after A_{BC} relocated cannot be processed. In other words, detached dependents of A_{BC} cannot make a uniform decision on the new best candidate node. Thus, the goal of [1] ([1, Section 4.3.3]) "... preventing conflicts in the selection of the BC" could not be achieved under some situation.

We next make a brief summary: Any best candidate node like A_{BC} cannot be simply treated as a failure node. The main reason is for a real failure node A_f, all of its neighbors must have the same observation regarding the failure of A_f. On the other hand, since any node like A_{BC} does not really fail but just change its location, it will cause the inconsistency among the observation among its original neighbors. As shown in previous example, the detached dependents set cannot be computed exactly among all the detached dependents.

3.2 Our Method for Restoring 1-Connectivity

Here, we propose a new approach in order to fix this problem.

Before the node A_{BC} moves to some new location, it should unicast Dependents(A_{BC}, A_f) (as $\{A_1, A_2, A_3\}$ in Fig. 2) to all its dependents. And based on the criteria of selecting the best candidate, each dependent can get a priority number, for example, if A_3 should be the best candidate after A_{BC} moved, then A_3 got the number 1, and A_2 is the next candidate, then A_2 got the number 2. We use W_i to denote the number A_i got. Let T denote the time for traveling the distance $2r$. Our refined method, after A_{BC} moved to the new location, is as follows:

- Each node $A_i \in \text{Dependents}(A_{BC}, A_f)$ waits to receive the "RECOVERED" message in time T.

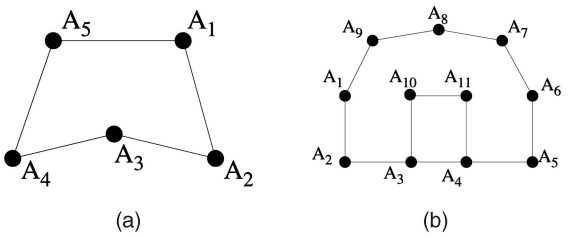


Fig. 3. The counterexamples for boundary nodes: (a) for network periphery and (b) for connectivity hole.

- If such a message is received, then this node does not move.
- If such a message is not received within time T, then wait for another $T \cdot W_i$ time.
  - If within the $T \cdot W_i$ time, it received the "RECOVERED" message, then it does not move.
  - Otherwise, at the end of $T \cdot W_i$ time, it moves to the location to A_{BC}, and then, broadcast a "RECOVERED" message.

It is not difficult to show that the neighboring nodes of the best candidate node will have the same view on what to do next and after the algorithm terminates, the final structure is indeed a connected network. We mainly let the candidate node A_{BC} to decide the priority number of each of its neighbors, and send the priority number information to all its neighbors before node A_{BC} moves. Consequently, they will have the same decision.

3.3 Discrepancies in Algorithm DARA-2C

The most significant discrepancy happens in [1, Section 5]. We found that based on the definitions of "boundary node" in [1], Theorem 6 is not valid at all. We first use Fig. 5 in paper [1] to illustrate the definition of "network periphery" and "connectivity hole." (refer to [1] to find Fig. 5.) In [9], the geometric convex hull of [1, Fig. 5] is the cycle defined by vertices sequence $A_1, A_6, A_4, A_{12}, A_3, A_{13}, A_7, A_5, A_{10}$. (Note that vertices A_2, A_{14} are not included.) The paper [1] uses the convex hull as the network periphery. The usage of convex hull to define network periphery as in [1] will cause some nodes that are *actually* on the network periphery be excluded.

Cited in [8], Abbasi et al. define the connectivity hole as "*the area in which the edges between nodes form a closed polygonal without links between nodes that are adjacent on the polygonal chain*," which is unclear to understand, so we here cite the original definition in [8]. In [8], connectivity hole is defined as *"an area where nodes cannot be placed and edges cannot exist,"* i.e., the connectivity hole is *independent* with the network topology. Connectivity hole used in [8] reflects a geographic region where we cannot physically put nodes in. Based on the definition of connectivity hole in [8], in [1, Fig. 5], all the nodes are not on the boundary of a connectivity hole. Abbasi et al. [1] give the definition of *boundary node* as the one located at the network periphery (defined as convex hull) or on the closed polygonal chain surrounding a connectivity hole (cited in [8]). With the definition of boundary node in [1], we provide two counterexamples for Theorem 6 [1], as shown in Fig. 3.

In Fig. 3a, the length of all the edge is 1 which is the maximum transmission range, and the distance between A_3 and $A_1(A_2)$ is $1+\epsilon$, where $\epsilon > 0$ is an arbitrarily small positive constant. By the definition of network periphery in [1], we know that in Fig. 3a, the network periphery is A_1, A_2, A_4, A_5, but the failure of A_3, not on the network periphery, can destroy the 2-connectivity of the network, which implies that Theorem 6 [1] is incorrect.

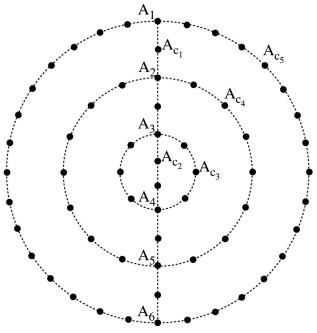


Fig. 4. When any vertex fails, there is a cut vertex.

It is clear that the network periphery in [1, Fig. 5] should be defined by the vertices sequence A_1, A_6, A_2, A_4, A_{12}, A_3, A_{13}, A_7, A_5, A_{14}, A_{10}, which is possibly what was proposed by the authors in [1].

Correct the discrepancy in paper [1]. Here, we give a correct definition of "network periphery."

Definition 1. *The* network periphery *is a subgraph of the network graph induced by the edges which can be connected to a point infinitely far away with curves that do not cross any edge or vertex of the network graph.*

With the corrected definition of network periphery, all the vertices in Fig. 3a are in the network periphery.

In Fig. 3b, the length of all the edges is 1, the distance between A_{10} and A_1(,A_9, A_8) is $1+\epsilon$, and the distance between A_{11} and A_6(, A_7, A_8) is $1+\epsilon$, where $\epsilon > 0$ is an arbitrarily small positive constant. So, by the definition of connectivity hole [8], there is no connectivity hole in Fig. 3b. However, the failure of A_{10} (or A_{11}) can destroy the 2-connectivity of the network, which again illustrates the incorrectness of [1, Theorem 6].

However, we found that it is very hard to correct the definition of connectivity hole in [1]. Maybe the authors in [1] wanted to define the connectivity hole as the faces which have more than three sides, for example, A_3, A_4, A_{11}, A_{10} in Fig. 3b. However, by this definition, almost all the vertices of the network graph except the ones on the network periphery are on the boundary of connectivity hole, which makes the definition meaningless. For example, in Fig. 3b, A_{10} and A_{11} are on the boundary of connectivity hole and they are the *only* two vertices not on the network periphery. It seems meaningless.

In Fig. 4, we give a series of input graphs in which when any vertex fails, there is some vertex that can become a cut vertex. In the example, there are several circles with the same center and a line segment crosses the center and intersects with each circle at two intersection points. On each intersection points, we deploy a vertex (such as $A_1, A_2, A_3, A_4, A_5, A_6$), and deploy other vertices uniformly on the circles and the line segment to guarantee connectivity (such as $A_{c_1}, A_{c_2}, A_{c_3}, A_{c_4}, A_{c_5}$).

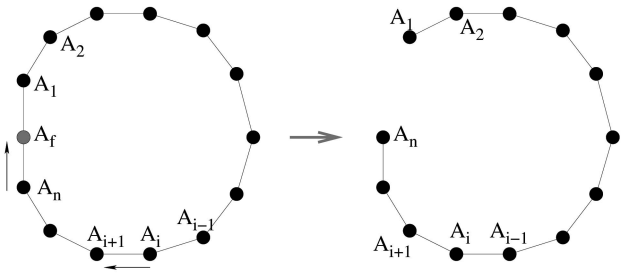


Fig. 5. A counterexample of DARA-2C. The algorithm will end with only 1-connected actor graph.

Although we cannot give an exact definition of connectivity hole, when there is some failed node A_f, we can get some properties based on the subgraph induced by the 1-hop neighbors of A_f. We denote the 1-hop neighbors of A_f as $N(A_f)$ and the subgraph induced by $N(A_f)$ as $G[N(A_f)]$. We claim the following lemmas:

Lemma 1. *If $G[N(A_f)]$ is 2-connected, the failure of A_f will not destroy the 2-connectivity of the network graph.*

Proof. For any two vertices u, v in the graph, there are two internally disjoint paths P_1, P_2 from u to v. There are totally three complementary cases: **Case 1)** both the two paths do not contain A_f, **Case 2)** P_1 contains A_f and P_2 does not contain any vertex in $N(A_f)$, and **Case 3)** P_1 contains A_f and P_2 contains some vertices in $N(A_f)$. We will analyze these three cases separately.

Case 1. After A_f fails, P_1 and P_2 will be the same.

Case 2. P_2 will be the same after A_f fails, and we can rebuild P_1 by using some nodes in $N(A_f)$ after A_f fails, since $N(A_f)$ is connected.

Case 3. P_2 will be the same after A_f fails. We denote the neighbors of A_f using by P_2 as $N_2(A_f)$. As in case 2, P_1 can be rebuilt by using some nodes in $N(A_f)$; we denote the neighbors of A_f using by P_1 as $N_1(A_f)$. We claim that we can always find $N_1(A_f)$ such that $N_1(A_f) \cap N_2(A_f) = \Phi$, here, Φ is empty. Because A_f was in P_1, so there are two nodes $A_1, A_2 \in N(A_f)$ in P_1. We know that P_1 and P_2 are disjoint paths, so $A_1, A_2 \notin N_2(A_f)$. Since $N(A_f)$ is 2-connected, then there are at least two disjoint paths between A_1 and A_2 in $N(A_f)$; one contains all the nodes in $N_2(A_f)$ and the other does not contain any node in $N_2(A_f)$. We choose the one not containing any node in $N_2(A_f)$ to rebuild P_1.

So, in all three cases, between any two nodes u and v, there are at least two disjoint paths, i.e., the graph after A_f fails is still 2-connected. □

Lemma 2. *If $G[N(A_f)]$ is only 1-connected (but not 2-connected) and a pair of nodes (A_f, A_c) is a cut, then $A_c \in N(A_f)$.*

Proof. We assume that (A_f, A_c) is a cut and $A_c \notin N(A_f)$, when $G[N(A_f)]$ is 1-connected. For that, (A_f, A_c) is a cut, there are two disjoint paths P_1 and P_2, connecting two vertices u and v such that $A_f \in P_1$ and $A_c \in P_2$. For that $A_c \notin N(A_f)$ and $G[N(A_f)]$ is 1-connected, we can rebuild P_1 by using some nodes in $N(A_f)$. Thus, u and v are still connected, which contradicts to the assumption that (A_f, A_c) is a cut. So, node $A_c \in N(A_f)$. □

Even if we assume that the definition of connectivity hole is valid and Theorem 6 in paper [1] holds, another significant discrepancy still exists. In the paper [1], the authors cited the localized algorithm in [19] to determine whether an actor node is a boundary node or not. However, the localized coverage boundary detection algorithm in [19] can only detect the nodes located at the network periphery and the nodes around the *coverage holes*. Note that the coverage hole is quit different from connectivity hole; coverage hole is the area which cannot be covered by the vertices and do not consider connectivity at all. So, [19] cannot be use here to detect the boundary nodes. Fig. 3b gives the counterexample. Because the algorithm in [19] can only detect the boundary of coverage hole of the *Region Of Interest* (ROI), in Fig. 3b, all the ROI is covered by nodes, so the connectivity hole *cannot* be detected in this example. Without detection of boundary nodes of WSAN, algorithm DARA-2C [1] cannot be implemented.

Even though we assume that there is some distributed algorithm that can detect the boundary nodes (on network periphery and the so-called connectivity hole), DARA-2C [1], based on Theorem 7 [1] (the argument is unconvincing by just given some example), cannot restore 2-connectivity under some specific input of the network graph, such as Fig. 5.

In Fig. 5, the length of all the edges is 1 and the network topology is a cycle which is 2-connected. Except the angle of $\angle A_1 A_f A_n = \pi$, all the other angles formed by the adjacent three nodes are greater than $\pi/2$ and smaller than π, for example, $\pi/2 < \angle A_{i-1} A_i A_{i+1} < \pi$. In the network graph illustrated in Fig. 5, A_f is failed at some time, then DARA-2C algorithm begins connectivity restoring process for that A_f is a boundary node. By DARA-2C, A_n is chosen as the best candidate (A_{BC}) and moves toward A_1 until A_n is within the communication range of A_1, which requires that A_n moves exactly to the location of A_f. After some period without hearing the "RECOVERED" message, the cascade node begins moving. The node moves just after A_n is A_{n-1} (in Fig. 5, we let $i = n - 2$), by DARA-2C, A_{n-1} moves toward the previous location of the A_{BC} which is A_n. Because the angle $\angle A_f A_n A_{n-1} > \pi/2$, A_{n-1} will be relocated at the previous location of A_n. The cascade moving continues, and for each A_i, the new location is exactly the previous location of A_{i+1}. So, when A_1 has moved to the previous position of A_2, the whole relocation process terminates because each node will move only once, and the resulted network graph is only 1-connected. Unfortunately, we could not come out with some method that can fix this algorithm.

3.4 Open Problems and Our Solution

In previous sections, we have showed that some methods presented in [1] will not work correctly always, and then, designed some methods that will give a correct result under certain scenarios. We find that the problem presented in [1] can be described as following in a more formal way:

Problem 1 (k-Connectivity restoration). *Assume that we are given a k-connected network $G = (V, E)$ (k is a given integer, say $k = 1$, or 2), in which each node $v_i \in G$ has a cost function $f_i(x)$, for example, $f_i(x) = a_i + b_i x$, that denotes the total cost of moving node v_i if we need to move node v_i of distance x.*

Assume that a node $A_f \in V$ failed, design a strategy to move nodes $V \setminus \{A_f\}$ to a set of possible locations U such that the network formed by nodes at their new locations will be k-connected. Our goal here is to minimize the total moving cost. Here, the set U could be infinite and no two nodes are allowed to be at the same position.

Clearly, if $f_i(x) = a_i + b_i x$, when $a_i = 1$ and $b_i = 0$ for all nodes v_i, then our goal becomes minimizing the number of moved nodes. When $a_i = 0$ and $b_i = 1$ for all nodes v_i, then our goal becomes minimizing the total moved distance of all nodes.

Notice that when we know that we must move nodes to the positions previously occupied by nodes in G, the problem can be easily solved using maximum weighted matching on a bi-partite graph H as follows: Given the graph $G = (V, E)$ of n nodes V and the failure of one node A_f, let $U_{-i} = V \setminus \{v_i\}$, where $v_i \in V$ is a node in V. Here, we use v_i also to denote the location of node v_i. Thus, a node w in U_{-i} also means that node w will take the same location as original. We then run method MCCR(G, $V \setminus \{A_f\}$, U_{-i}) (see Algorithm 1 for details) to check if there is a movement for k-connectivity restoration with the minimum cost. We will run the above approach for all possible sets U_{-i} (there are at most n such sets) such that the graph G reduced to U_{-i} is k-connected. Among the n solutions found, we then take the one that gives us the minimum cost. That solution will be the final optimum solution.

Algorithm 1. Minimum Cost k-Connectivity Restoration MCCR(G,$\bar{V}$,P)

Input: k-connected network $G = (V, E)$ with n nodes V, a set of alive nodes $\bar{V}$, and a set $P = \{p_1, p_2, \ldots, p_{|\bar{V}|}\}$ of possible new locations with size same as that of $\bar{V}$. A cost function $f : V \times P \rightarrow R^+$ specifying the cost moving a node from position in V to a position in P.

1: We first check if the graph defined on P based on the communication model is k-connected.
2: **if** it is not k-connected **then**
3: Return **Nil**, i.e., moving nodes to positions specified by P will not result in a k-connected network.
4: **else**
5: We build a bi-partite graph H on two sets of nodes $\bar{V}$ and P. The weight of an edge vu where $v \in \bar{V}$ and $u \in P$ is the cost $M - f(v, u)$, where $M > \max_{v \in V, u \in P} f(v, u)$ is a large number.
6: Find a maximum weighted matching in graph H. Obviously, the matching will have $|\bar{V}|$ edges. If vu is an edge in the matching, then we move node v to the position of u.
7: **end if**

It is easy to prove the following lemma:

Lemma 3. *The maximum weighted matching computed by Algorithm 1 gives an optimum moving strategy when the alive nodes are $\bar{V}$ and potential positions are P, with moving cost specified by f.*

This strategy can also be extended to solve the connectivity-restoration problem when there are multiple faulty nodes, and the cardinality of the possible positions U for all alive nodes, denoted as $\bar{V}$, is $\bar{n} + c$ for some constant c, where $\bar{n}$ is the number of alive nodes $\bar{V}$. For each subset $U_i \subset U$ that has exactly $\bar{n}$ positions, we run Algorithm MCCR($G, \bar{V}, U_i$) to find an optimal moving when potential positions are U_i. Constant c implies that the number of possible placements U_i of alive nodes is $\binom{\bar{n}+c}{\bar{n}} = O(\bar{n}^c)$, which, in turn, implies that we only need to solve $O(\bar{n}^c)$ number of subproblems mentioned in previous discussions using Algorithm 1. Then, we have the following theorem:

Theorem 1. *Assume that the cost of moving a node at position x to another position y is denoted by an arbitrary positive value $f(x, y)$. The connectivity-restoration problem can be solved in polynomial time for a network of n nodes when the cardinality of the possible new locations of all alive nodes is $n + c$ for some constant c and the number of failures is at most another constant c_1.*

4 Conclusion

We show that the algorithms proposed in [1] will not work correctly in many scenarios and we demonstrated this by presenting some counterexamples. We also fixed some discrepancies of [1]. We presented a centralized method that can solve the k-Connectivity restoration problem, including the problem studied in [1] as a special case, when the number of potential positions of alive nodes is only a constant more than the number of alive nodes. We would like to solve the general open problem discussed in the previous section, by designing both efficient centralized and efficient distributed methods with a certain performance guarantee.

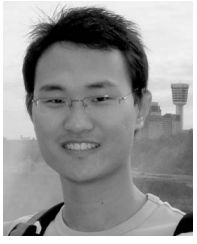
Acknowledgments

The research of authors is partially supported by the US National Science Foundation (NSF) CNS-0832120, National Natural Science Foundation of China under Grant No. 60828003, No. 60773042, and No. 60803126, the Natural Science Foundation of Zhejiang Province under Grant No. Z1080979, program for Zhejiang Provincial Key Innovative Research Team, program for Zhejiang Provincial Overseas High-Level Talents (100 Talents Program), the National Basic Research Program of China (973 Program) under grant No. 2010CB328100, the National High Technology Research and Development Program of China (863 Program) under grant No. 2007AA01Z180, and Tsinghua National Laboratory for Information Science and Technology(TNList).

References

[1] A.A. Abbasi, M. Younis, and K. Akkaya, "Movement-Assisted Connectivity Restoration in Wireless Sensor and Actor Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 20, no. 9, pp. 1366-1379, Sept. 2009.
[2] K. Akkaya, A. Thimmapuram, F. Senel, and S. Uludag, "Distributed Recovery of Actor Failures in Wireless Sensor and Actor Networks," *Proc. IEEE Wireless Comm. and Networking Conf.*, 2008.
[3] H.M. Almasaeid and A. Kamal, "On the Minimum K-Connectivity Repair in Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Comm.*, pp. 1-5, 2009.
[4] N. Atay and B. Bayazit, "Mobile Wireless Sensor Network Connectivity Repair with K-Redundancy," Technical Report WUCSE-2007-48, Dept. of Computer Science and Eng., Washington Univ. in St. Louis, 2007.

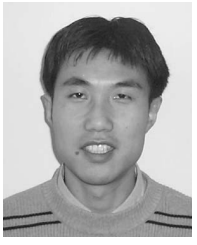
[5] X. Cheng, D. Du, L. Wang, and B. Xu, "Relay Sensor Placement in Wireless Sensor Networks," *Wireless Networks,* vol. 14, no. 3, pp. 347-355, 2008.
[6] A. Czumaj and A. Lingas, "Approximation Schemes for Minimum-Cost K-Connectivity Problems in Geometric Graphs," *Handbook of Approximation Algorithms and Metaheuristics,* Chapman and Hall, 2007.
[7] S. Das, H. Liu, A. Nayak, and I. Stojmenović, "A Localized Algorithm for Bi-Connectivity of Connected Mobile Robots," *Telecomm. Systems,* vol. 40, no. 3, pp. 129-140. 2009.
[8] G. Destino and G. de Abreu, "Network Boundary Recognition via Graph-Theory," *Proc. Fifth Workshop Positioning, Navigation and Comm.,* pp. 271-275, 2008.
[9] R. Diestel, *Graph Theory.* Springer, 2005.
[10] GreenObs Project, A Long-Term Kilo-Scale Wireless Sensor Network System in the Vast Fores, http://www.greenorbs.org/, 2009.
[11] A. Ibrahim, K. Seddik, and K. Liu, "Improving Connectivity via Relays Deployment in Wireless Sensor Networks," *Proc. IEEE Global Telecomm. Conf. (Globecom),* pp. 1159-1163, 2007.
[12] A. Ibrahim, K. Seddik, and K. Liu, "Connectivity-Aware Network Maintenance and Repair via Relays Deployment," *IEEE Trans. Wireless Comm.,* vol. 8, no. 1, pp. 356-366, Jan. 2009.
[13] A. Kashyap, S. Khuller, and M. Shayman, "Relay Placement for Higher Order Connectivity in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* 2006.
[14] N. Li and J.C. Hou, "Improving Connectivity of Wireless Ad Hoc Networks," *Proc. IEEE MOBIQUITOUS,* pp. 314-324, 2005.
[15] X.-Y. Li, P.-J. Wan, Y. Wang, C.-W. Yi, and O. Frieder, "Robust Deployment and Fault Tolerant Topology Control for Wireless Ad Hoc Networks," *Wiley J. Wireless Comm. and Mobile Computing,* vol. 4, no. 1, pp. 109-125, 2004.
[16] X.-Y. Li, Y. Wang, P.-J. Wang, and C.-W. Yi, "Fault Tolerant Deployment and Topology Control for Wireless Ad Hoc Networks," *Proc. ACM MobiHoc,* 2003.
[17] M. Lufeng, H. Yuan, L. Yunhao, Z. Jizhong, T. Shaojie, L. Xiang-Yang, and D. Guojun, "Canopy Closure Estimates with GreenOrbs: Sustainable Sensing in the Forest," *Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys),* 2009.
[18] C.-W. Yi, P.-J. Wan, X.-Y. Li, and O. Frieder, "Asymptotic Distribution of the Number of Isolated Nodes in Wireless Ad Hoc Networks with Bernoulli Nodes," *IEEE Trans. Comm.,* vol. 54, no. 3, pp. 510-517, Mar. 2006.
[19] C. Zhang, Y. Zhang, and Y. Fang, "Detecting Coverage Boundary Nodes in Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Networking, Sensing and Control (ICNSC),* vol. 6, 2006.



ShiGuang Wang received the BS degree from Nanjing University, P.R. China, in 2008. He is currently working toward the PhD degree in computer science at Illinois Institute of Technology. His research interests span algorithm and system design for wireless ad hoc and sensor networks, game theoretical study of networks, and RFID system. He is a student member of the IEEE.

XuFei Mao received the BS degree from Shenyang University of Technology, and the MS degree from Northeastern University, China. He is currently working toward the PhD degree in computer science at Illinois Institute of Technology. His research interests include design and analysis of algorithms for wireless networks and the design and implementation of large-scale wireless sensor networks. He is a student member of the IEEE.

Shao-Jie Tang received the BS degree from Southeast University, P.R. China, in 2006. He is currently working toward the PhD degree in computer science at Illinois Institute of Technology. His research field is on algorithm design, optimization, security of wireless networks as well as electronic commerce.



Xiang-Yang Li received the BEng degree in computer science and the bachelor's degree in business management from Tsinghua University, P.R. China, in 1995, and the MS and PhD degrees in computer science from the University of Illinois at Urbana-Champaign in 2000 and 2001, respectively. He has been an associate professor since 2006. He was an assistant professor of computer science at Illinois Institute of Technology from 2000 to 2006. He was a visiting professor of Microsoft Research Asia from May 2007 to August 2008. His research interests span wireless ad hoc and sensor networks, noncooperative computing, computational geometry, and algorithms. He is an editor of the *IEEE TPDS* and *Networks: An International Journal*, and was a guest editor of special issues for the *ACM Mobile Networks and Applications*, *IEEE Journal on Selected Areas in Communications*, and several other journals. In 2008, he published a monograph *Wireless Ad Hoc and Sensor Networks: Theory and Applications*, by Cambridge University Press. He is a senior member of the IEEE and a member of the ACM.

JiZhong Zhao received the BS and MS degrees in mathematics and the PhD degree in computer science, focused on Distributed System, in 2001 from Xi'an Jiaotong University, China, where he is a professor in the Computer Science and Technology Department. His research interests include computer software, pervasive computing, distributed systems, and network security. He is a member of the IEEE Computer Society and the ACM.

GuoJun Dai received the MS degree in 1991 and the PhD degree in electronic engineering in 1998 from Zhejiang University. He is a professor in the Computer Science Department, Institute of Computer Application Technology, Hangzhou DianZi University, P.R. China, where he is also the vice dean in the School of Computer Science. His research field is on wireless sensor networks, embedded systems, applications of CPS systems, and computer graphics.