JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 39, No. 3, March 2016

Message-Passing Strategy for Decentralized Connectivity Maintenance in Multiagent Surveillance

Derya Aksaray*
Boston University, Boston, Massachusetts 02215
A. Yasin Yazıcıoğlu†
Massachusetts Institute of Technology, Cambridge, Massachusetts 02139
and
Eric Feron‡ and Dimitri N. Mavris§
Georgia Institute of Technology, Atlanta, Georgia 30332

DOI: 10.2514/1.G001230

In multiagent surveillance missions, a group of agents monitors some points of interest to provide situational awareness. For agents with local communication capabilities, one way to transmit the surveillance information to a base is streaming by the instantaneous data via multihop communications over a connected network. However, a connected communication network may become disconnected if some agents leave the surveillance area (for example, for refueling). This paper presents a locally applicable, efficient, and scalable strategy that guarantees a connected communication network between the base and the agents in the face of any agent removal. The proposed decentralized strategy is based on a sequence of local replacements, which are initiated by the agent leaving the network. It is shown that the replacement sequence always ends with the relocation of an agent, for which the absence from its current position does not disconnect the network. Furthermore, the optimality (that is, the minimum number of replacements) of the proposed scheme is improved by incorporating a local criticality notion in the decision mechanism. Finally, the cases are considered in which some agents are not allowed to execute a replacement, and it is shown that the proposed strategy maintains a connected communication network, even in the presence of such constraints.

I. Introduction

OVER the last decade, advances in networking and computing technologies along with new manufacturing techniques have created a new paradigm shift toward multiagent systems (MASs) in engineering applications. A MAS involves a set of agents, each of which can be a robot, a satellite, or a sensor, to name a few (e.g., [1–3]). Recently, there is a significant interest in using MASs for target tracking (e.g., [4]), environmental monitoring (e.g., [5]), persistent surveillance (e.g., [6]), coverage control (e.g., [7]), and several others. In a MAS, the connectivity of the communication network plays a significant role to achieve collaboration among the agents through some local interactions. For instance, multiple spacecraft can synchronize their attitudes with each other via local interactions if they maintain a formation with a connected communication network (e.g., [8,9]). Alternatively, a group of unmanned aerial vehicles (UAVs) can stream the surveillance data back to the base, if they have a connected communication network including the base (e.g., [10,11]).

Even though a MAS begins a mission with a connected communication network, such systems have an inherent possibility of agent removal due to failure, refueling, or some other reasons (e.g., [12]). In the face of an agent removal, the communication network may become disconnected. In networked systems, disconnection can be avoided or fixed by proactive (e.g., [13–15]) or reactive (e.g., [16,17]) approaches, respectively. In proactive approaches, a robust network topology is designed a priori such that the network can tolerate a certain number of agent removals. Note that relying only on proactive approaches can be impractical in applications where a large number of agents may eventually be removed from the network. In reactive approaches, a control strategy is developed for the network to reconfigure itself in the face of agent removals. A recovery process can be characterized as centralized or decentralized based on the information leveraged in the decision scheme. In a large-scale system, the availability of global information to individual agents is usually not feasible. Therefore, a decentralized strategy becomes more desirable than a centralized one due to practicality and scalability concerns.

This paper introduces a decentralized recovery scheme that is applicable to networked systems involving mobile agents. The proposed scheme is mainly based on a sequence of local replacements, and it solves the replacement control problem, which was initially posed in [18]. In our solution, each agent is assumed to have a unique identity (ID), which is known to its immediate neighbors. Before an agent leaves the group (e.g., due to reaching a critical fuel threshold), we assume that it creates a message with its individual ID and passes it to one of its neighbors as a request for that neighbor to replace the leaving agent. Whenever an agent receives the message, first, it appends its own ID to the message and sends it to one of its other neighbors, for which the ID is not included in the message yet (i.e., a neighbor who has not yet received the message). Then, it executes the requested replacement. A strategy based on consecutive message passing is similar to token-based techniques used in various algorithms, such as the one in [19], to record the agents involved in a process.

This work represents significant progress beyond [18], where the replacements by some minimum-degree neighbors were presented as a solution. The contributions of this paper are as follows:

1) We generalize the connectivity maintenance scheme as the message-passing strategy (MPS), and we show that the MPS recovers connectivity even in the case of agents using a minimum amount of information, i.e., only neighbor IDs.

2) We inspect the performance and optimality of the MPS and relate it to the graph structure.

3) We incorporate a local criticality notion to improve the performance of the MPS, and we provide a sufficient condition on the graph

Received 23 December 2014; revision received 25 August 2015; accepted for publication 29 August 2015; published online 13 November 2015.

*Postdoctoral Associate, Department of Mechanical Engineering. Member AIAA.

†Postdoctoral Associate, Laboratory for Information and Decision Systems.

‡Dutton/Ducoffe Professor of Aerospace Software Engineering, School of Aerospace Engineering. Associate Fellow AIAA.

§Boeing Regents Professor, School of Aerospace Engineering. Fellow AIAA.

topology for the equivalence of the local criticality to the global criticality.

4) The strategy proposed in [18] and the MPS have an underlying assumption that a replacement is executed whenever a request is received by an agent. In this paper, we also consider the cases where some agents are not allowed to execute a replacement (e.g., due to monitoring a special region). We call these agents the constrained agents, and we propose a variant of the MPS that ensures the recovery of the network connectivity in a decentralized fashion without moving the constrained agents. We use the proposed strategy in a multiagent surveillance scenario, where the base (or bases) is modeled as the constrained agent (or agents). We show that a connected communication network is maintained between the base (or bases) and the agents in the face of any agent removal.

The organization of this paper is as follows: Sec. II presents a brief review of graph theory and some related work in the literature. Section III motivates and defines the multiagent surveillance problem. Section IV presents the MPS, which guarantees the decentralized recovery of network connectivity in arbitrary agent removal through a sequence of local replacements. Section V presents a variant of the MPS that ensures the recovery of the network connectivity, even in the cases where some agents reject a replacement request. Finally, Sec. VI concludes the paper by providing a brief summary of the proposed ideas.

II. Background

A. Graph Theory Preliminaries

An undirected graph $\mathcal{G} = (V, E)$ consists of a set of nodes V and a set of undirected edges E. For any subset of the nodes ($X \subset V$), $\mathcal{G}_X$ refers to a subgraph induced by the nodes in X. In other words, $\mathcal{G}_X$ consists of all the nodes in X and their corresponding edges. In a graph, a k-length path $\boldsymbol{p}$ is a sequence of nodes $(p_0, p_1, \ldots, p_k)$ such that the edge between any p_i and p_{i+1} belongs to E. A path in $\mathcal{G}$ is called simple if it does not have any repeated nodes. An undirected graph $\mathcal{G}$ is connected if there exists a path between any two nodes of the graph.

Let v_j and v_k be any two nodes in $\mathcal{G}$. Then, the distance between v_j and v_k is denoted as $d(v_j, v_k)$, and it is equal to the length of the shortest path between them. The diameter of $\mathcal{G}$ is defined as the largest distance (i.e., longest shortest path) over all pairs of nodes of $\mathcal{G}$. The neighbor set of node v_i is denoted as $\mathcal{N}_{v_i}$, and it is the set including all adjacent nodes that are connected to v_i; that is,

$$\mathcal{N}_{v_i} = \{v_j | (v_i, v_j) \in E\} \tag{1}$$

The degree of v_i is defined as the cardinality of $\mathcal{N}_{v_i}$. Furthermore, the δ-hop neighborhood of v_i is denoted as $\mathcal{N}^{\delta}_{v_i}$, and it is the set including nodes that are, at most, a distance δ away from v_i; that is,

$$\mathcal{N}^{\delta}_{v_i} = \{v_j | d(v_i, v_j) \leq \delta\} \tag{2}$$

B. Related Work

Recently, a great amount of interest has been devoted to the analysis of multiagent systems using graph theory. In such analyses, multiagent networks are denoted as interaction graphs, where the nodes represent the agents (such as robots, sensors, or individuals) and the edges represent the direct interactions between them. Note that connected interaction graphs play an important role in achieving various multiagent coordination tasks (e.g., [1,20,21]).

The literature on the graph theoretical connectivity control of mobile systems against edge failure includes, but is not limited to, optimization-based connectivity control (e.g., [1]), continuous feedback connectivity control (e.g., [22]), or control based on the estimation of the algebraic connectivity (e.g., [23]). In these studies, the authors mainly consider some perturbations of the edges, and they assume a constant number of nodes.

Various aspects of agent loss problems in networked systems have been studied in the literature. In [13] and [14], the main focus is on the design of robust network topologies that can tolerate a certain number of agent removals. In [14] and [24], the authors proposed self-repair strategies that created new connections among the neighbors of the failing agent. Alternatively, some researchers consider mobile agents and propose some agent movements for connectivity restoration. Some relevant works on distributed connectivity recovery of wireless sensor networks in agent failure include [16,17,25–27]. The authors of [16] and [25] proposed a set of cascaded replacements, which was similar to what we propose for the recovery process. Compared to their works, which involved extended numerical results on the subject, we treat the problem from an analytical perspective and additionally consider some agents that might refuse to execute a replacement. Alternatively, the authors of [17] proposed a block movement of agents instead of cascaded replacements, and their algorithm restored connectivity without extending the shortest paths among agents after a failure. Their scheme was based on the shortest-path routing table, for which the population greatly affected the resulting agent movements.

In networked systems, an agent is called critical if its removal causes a disconnection. Determining agent criticality is an important step for the development of efficient control strategies because a recovery process is required only in the face of a critical agent removal. However, agent criticality is global information that is usually not locally observable. Therefore, many studies in the literature usually rely on some global information for the initiation of the process (e.g., [16,25]).

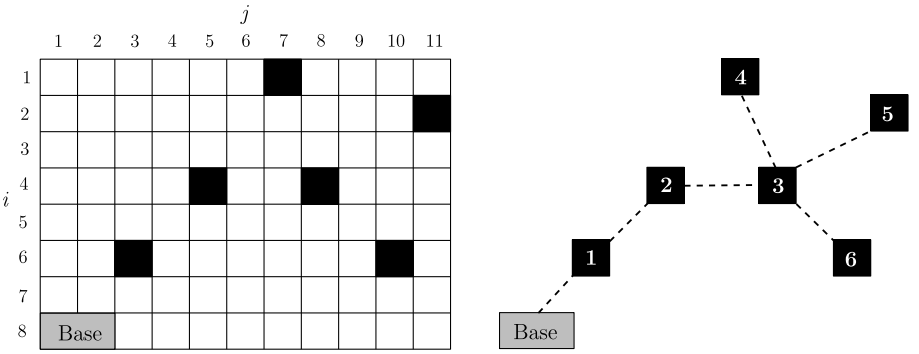
An approach to develop more localized algorithms for determining criticality is using information only in the k-hop neighborhood. For instance, the authors of [26,27] proposed a connectivity restoration framework, where each agent constructed a routing table from its k-hop neighborhood to estimate its neighbors' criticality. Like [26] and [27], we also use a notion as k-criticality. Complementing the aforementioned studies, which discuss false detection of criticality and connectivity restoration via simulations, we present a graph topological sufficient condition for the equivalence of k-criticality to the global criticality. Moreover, this paper additionally differs from the existing studies by proposing a decentralized strategy that guarantees the connectivity maintenance by considering some constraints in the agent mobility. Such constraints model the cases where the corresponding agent has low remaining fuel or is currently monitoring an area that is essential for the overall mission. Hence, our proposed strategy results in a network reconfiguration without moving such constrained agents.

Finally, in many studies, agent failure in sensor networks is studied based on the assumption of agents sending periodic heartbeat messages (e.g., [17,16,26]). Accordingly, if a heartbeat message is not received by the neighbors of agent i for some period of time, then agent i is assumed dead by the neighbors. Note that our main focus is on the removal of agents due to reasons such as reaching an energy threshold or changing mission strategy. These types of reasons are mostly seen in surveillance missions, where agents may be required to return to the base for refueling (e.g., [6,28]) or may leave the group for tracking a target (e.g., [11,10]). As such, we study how the remaining agents need to reconfigure themselves in a decentralized fashion if a particular agent voluntarily decides to leave the system. Such a voluntary leave of an agent is referred to as agent removal throughout the paper.

III. Problem Formulation

A. Environment Model

This paper addresses a multiagent surveillance problem, where the major objective is to increase the situational awareness about a target area. To achieve this goal, a base sends a group of agents to the target area and receives instantaneous information from the agents. Suppose that a desired region of interest is discretized as a two-dimensional (2-D) grid and each grid cell has a relative weighting that indicates the importance of the corresponding region. In a large surveillance area, the weightings of each grid cell may not necessarily be uniform because some cells may represent more essential regions,



a) b)

Fig. 1 Environment represented as a) 2-D grid, and b) connected graph.

such as the oil rigs on the sea, the high-population zones in a city, or the disaster areas in an environment.

In this paper, the surveillance area is represented as in Fig. 1a, where the black cells denote the high-priority zones with a weighting of one (i.e., the monitoring points), the white cells represent the rest of the points with a weighting of zero (i.e., the points that do not require monitoring), and the gray cell denotes the base. In this setting, the base sends some agents (e.g., UAVs) to the field to gather information about the points of interest. Note that the agents and the base have limited communication capability over the field because the continuous wireless communication may not be always feasible, reliable, or secure due to weather effects, jamming attacks, or service delays [29]. In the presence of limited communications, the environment can also be represented as a graph, which takes into account not only the geographical coordinates of the monitoring points and the base but also the communication ranges of the agents. For instance, suppose that six agents having a communication range of R will be located on the grid cells in Fig. 1a. Accordingly, if the distances between any pair of grid cells are less than R, then the agents located on the corresponding cells can communicate with each other. Hence, the limited communication capabilities can be incorporated by representing the surveillance area as a graph, shown in Fig. 1b.

In an ideal case, if the agents and the base have a connected communication network and each node is monitored by an individual agent, then the base has the maximum situational awareness about the field. However, in long-run missions, an agent has an inherent possibility of leaving the surveillance area due to refueling, maintenance, or a strategy change. The removal of an agent becomes a disturbance to the overall system, and it causes a degradation in situational awareness due to preventing the regular updates of some particular nodes. For instance, Fig. 2a illustrates a scenario where each UAV monitors a single node and sends data back to the base via a connected communication network. Suppose that UAV 4 needs to return to the base for refueling. As seen from Fig. 2b, the removal of UAV 4 causes not only an unoccupied node but also a disconnection among UAVs 3, 5, 6, and the base. Here, even though UAVs 3, 5, and 6 monitor their assigned nodes, they cannot send information back to the base due to the disconnection.

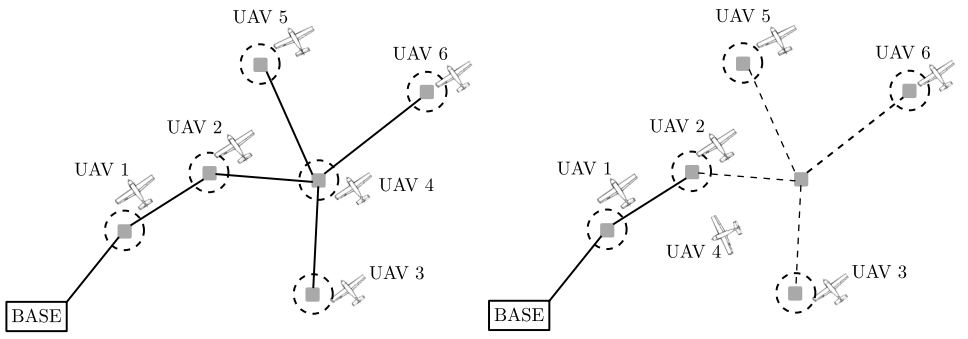
B. Replacement Control Problem

In this study, we consider a discrete time problem that consists of m agents, $V = \{v_1, \ldots, v_m\}$, occupying the nodes, $V^* = \{1, 2, \ldots, n\}$, of a connected graph $\mathcal{G}^*$, where $n \geq m$. For instance, Fig. 3 illustrates an example to $\mathcal{G}^*$, in which a node represents a monitoring region and an edge implies the communication capability of two agents if they are located on the corresponding nodes. Note that an agent can communicate with the base if some particular nodes are occupied (e.g., in Fig. 3, it is necessary to occupy node 14 to communicate with the base).

In this setting, each node is occupied by a single agent. Moreover, the nodes are located far from each other, and the communication range of each agent is assumed larger than its monitoring range. Therefore, agents are urged to stay as close as possible to the nodes. Accordingly, the communication network of the agents becomes a subgraph of $\mathcal{G}^*$, denoted by $\mathcal{G}$. For instance, Fig. 4 shows some examples for the communication networks of agents located on the nodes of a connected graph.

In the case of an agent removal, a disconnection may happen in $\mathcal{G}$, as shown in Fig. 4b. To recover connectivity, we assume that the agents are able to move on the space until $\mathcal{G}$ becomes connected. Implicitly, we therefore assume that the ability to communicate from one agent to another is synonymous with being able to move across the corresponding locations. Such an assumption can be granted by making sure that any transmission from one agent to its neighbors contains that agent's position. Under such assumptions, an agent is capable of moving only to a neighboring node in a single time step. As such, v_8 in Fig. 4b can either stay in its current position (i.e., 10) or move to 7, 11, or 12 in the next time step.

In this problem, multiple agent removals may occur; however, we assume that a single agent is removed at a particular instant. For $k \geq 1$, let T_k^r be the time instant for the kth removal. We assume that the time instances for agent removals are sufficiently spaced and



a) b)

Fig. 2 Communication network: a) connected, and b) disconnected.

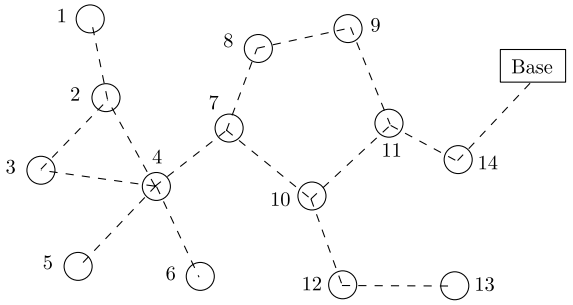


Fig. 3 Connected network structure $\mathcal{G}^*$.

satisfying $T^r_{k+1} - T^r_k \geq \tau$, where τ is greater than the maximum time needed to reconfigure the network after a removal.

Assuming that agents are homogeneous with respect to network operations and task coexecution can be achieved between any pair of agents, one way to recover connectivity in this problem is to replace the removed agent by one of the remaining agents. For instance, if the removal of an agent causes a disconnection, then one of its neighbors may replace it to recover the connectivity. Similarly, if that replacement also causes a disconnection, then another replacement is required. In this manner, the replacements can be executed until a connected network is obtained. Here, any feasible solution has to recover connectivity after a finite number of replacements, and it is desired to realize the minimum number of replacements. Furthermore, it is preferable to develop a scheme that can be executed by agents making local decisions with limited information. Accordingly, we revise the replacement control (RC) problem initially posed in [18] as follows:

Let a group of agents have a connected communication network. Design a decentralized control scheme such that, for any agent removal, the agents reach a connected communication network via as few as possible local replacements.

In the following two sections, we propose various decentralized strategies as solutions to the RC problem. In Sec. IV, we consider an unconstrained problem, where any agent can execute a replacement when a request is received. Accordingly, the agents reconfigure themselves to recover the connectivity among each other in the face of an agent removal. Then, we use the results of the unconstrained problem later in Sec. V to solve the RC problem in constrained scenarios, where some agents are not allowed to replace an agent. These cases mainly represent the problems that require not only a connected network among the agents but also connectivity to some specific points. Finally, we use the results of Sec. V in a multiagent surveillance scenario, where a set of agents and a pair of bases maintain a connected communication network in the face of agent removals.

IV. Decentralized Connectivity Maintenance with Unconstrained Agents

A. Message-Passing Strategy

Given a connected graph $\mathcal{G}$, where the nodes in $\mathcal{G}$ represent the agents, the RC problem finds a sequence of replacements to recover connectivity. For any solution of the RC problem, the sequence of replacements needs to end with the relocation of a noncritical node since the removal of such nodes from their previous positions does not require any further replacements.

Definition 1 (node criticality): Let v_i be a node in $\mathcal{G}$ and E_i be the edges incident to v_i. Then, v_i is critical in $\mathcal{G}$ if the graph, $\mathcal{G}' = \mathcal{G} - (v_i, E_i)$, obtained by removing v_i and E_i is disconnected; otherwise, v_i is noncritical.

A connected undirected graph always has a finite number of noncritical nodes [30]. Thus, the goal of the RC problem is to find one of them. The following remark presents a trivial sufficient condition for a node to be noncritical in a graph.

Remark 1: Given a connected graph $\mathcal{G} = (V, E)$, let $v_i \in V$ be a leaf node such that $|\mathcal{N}_{v_i}| = 1$. Then, v_i is noncritical in $\mathcal{G}$ because any simple path involving v_i either starts or ends with v_i. Hence, its removal does not cause a disconnection between any two nodes.

In the RC problem, a replacement is assumed to occur between a node and one of its neighbors. Therefore, the sequence of replacements can be represented as a path from the removed node to a noncritical node. Note that, for a connected undirected graph $\mathcal{G}$, there always exists a path from any node in $\mathcal{G}$ to a noncritical node.

A centralized controller can provide the optimal solution to the RC problem by finding a shortest path between the removed node and a noncritical node. However, the optimal solution is obtained by assuming the availability of the overall graph structure. The goal of our work is to find a decentralized scheme that can perform "close to optimal."

Definition 2 (maximal simple path): Let $\mathcal{G} = (V, E)$ be a connected undirected graph, and let $\mathcal{N}_{v_i}$ denote the neighbors of $v_i \in V$. Suppose that $\boldsymbol{p} = (p_0, p_1, \ldots, p_k)$ is a simple path of length k. Then, $\boldsymbol{p}$ is a maximal simple path if $\mathcal{N}_{p_k} \subseteq \{p_0, p_1, \ldots, p_k\}$.

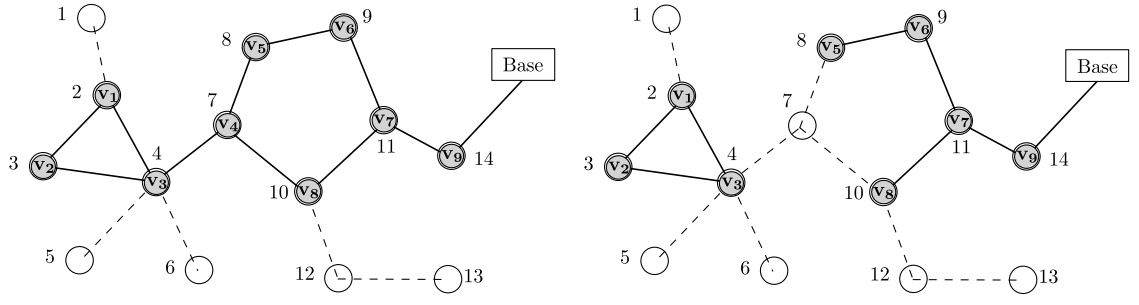
Theorem 1 [18]: For any connected undirected graph $\mathcal{G}$, a maximal simple path always ends with a noncritical node.

Corollary 1: For any connected $\mathcal{G} = (V, E)$ and arbitrarily removed $p_0 \in V$, a sequence of replacements along a maximal simple path, $(p_0, p_1, \ldots, p_k)$, such that any $p_{i+1} \in \mathcal{N}_{p_i} \setminus \{p_0, p_1, \ldots, p_i\}$ for $i < k$, recovers the graph connectivity.

Proof: The maximal simple path $(p_0, p_1, \ldots, p_k)$ is the replacement path where any p_i is replaced by p_{i+1} for $0 \leq i \leq k-1$. Note that, after the replacements are completed, the graph will have a new structure as if p_k was removed from the system. From Theorem 1, we know that p_k is noncritical, so its removal from its previous location does not cause any loss of connection in $\mathcal{G}$. □

In light of the preceding facts, we introduce a decentralized connectivity maintenance scheme called the message-passing strategy. Let p_0 be any arbitrary node that is removed from $\mathcal{G}$. The objective of the MPS is to find a sequence of replacements, which is initiated by p_0 and ends with a noncritical node, by using only local information. In this manner, the replacements result in a reconfiguration such that the graph becomes identical to the initial graph minus a noncritical node, which is the final node in the replacement sequence.

The outline of the MPS is as follows: Before the removal of p_0, p_0 first creates a message including its own node ID as $\{p_0\}$ and checks whether it is a leaf node. If it is a leaf node, then it is noncritical (from



a) Connected communication network **b) Disconnected communication network**

Fig. 4 Representations of a) a connected communication network $\mathcal{G}$; and b) a disconnection in $\mathcal{G}$ due the removal of v_4.

Remark 1) and its removal does not cause a disconnection. Otherwise, it selects a node p_1 from $\mathcal{N}_{p_0} \setminus \{p_0\}$. Then, p_0 sends the message to p_1, which will replace p_0. In this respect, whenever a node p_i receives a message, $\{p_0, \ldots, p_{i-1}\}$, it appends its individual node ID to the message as $\{p_0, \ldots, p_{i-1}, p_i\}$. Then, it sends the message to one of its neighbors from the set $\mathcal{N}_{p_i} \setminus \{p_0, \ldots, p_i\}$ before replacing p_{i-1}. Eventually, the message-passing process, for which the pseudocode is displayed in Algorithm 1, stops when $\mathcal{N}_{p_i} \setminus \{p_0, \ldots, p_i\} = \emptyset$ or p_0 is a leaf node.

Proposition 1: MPS always stops, and it stops at a noncritical node.

Proof: Let $p_0 \in V$ be any arbitrary node that will be removed from $\mathcal{G}$. If p_0 is a leaf node, the MPS stops at p_0. Otherwise, p_0 generates a message as $\{p_0\}$, and the message is modified as $\{p_0, p_1, \ldots, p_i\}$ whenever it is received by $p_i \in V$. Note that the message passing continues if $\mathcal{N}_{p_i} \setminus \{p_0, \ldots, p_i\} \neq \emptyset$, and it will eventually stop (e.g., containing all nodes in $\mathcal{G}$). Therefore, $\mathcal{N}_{p_i} \subseteq \{p_0, \ldots, p_i\}$ is eventually satisfied (e.g., when $\{p_0, \ldots, p_i\} \to V$), and the MPS stops at p_i. From Theorem 1, p_i is noncritical because $\mathcal{N}_{p_i} \setminus \{p_0, \ldots, p_i\} = \emptyset$. Consequently, the MPS always stops at a noncritical node. □

Corollary 2: The message obtained when the MPS stops is a set of ordered nodes, which corresponds to either a single leaf node or a maximal simple path. Hence, the MPS guarantees connectivity maintenance in the removal of any arbitrary node from $\mathcal{G} = (V, E)$.

Proof: The message obtained from the MPS is either $\{p_0\}$ or $\{p_0, \ldots, p_i, \ldots, p_k\}$. If it is $\{p_0\}$, then $|\mathcal{N}_{p_0}| = 1$, implying that p_0 is a leaf node, and the connectivity maintenance is an immediate result. If the message is $\{p_0, \ldots, p_i, \ldots, p_k\}$, it involves consecutive pairs of nodes, $(p_i, p_{i+1}) \in E$; thus, the message always represents a path in $\mathcal{G}$. Additionally, the message never involves repeated nodes because each p_i selects p_{i+1} from $\mathcal{N}_{p_i} \setminus \{p_0, \ldots, p_i\}$. Thus, the path is always simple. Finally, the MPS stops whenever $\mathcal{N}_{p_k} \setminus \{p_0, \ldots, p_i, \ldots, p_k\} = \emptyset$. From Definition 2, the ordered nodes in the message are a maximal simple path. From Corollary 1, the replacements based on a maximal simple path always guarantee connectivity recovery because the final graph is reconfigured as the initial graph without p_k. □

An illustration for the MPS is displayed in Fig. 5, where there is initially a connected graph with seven nodes. As is seen in Fig. 5b, the removal of v_0 will create a disconnection in the graph. If each node runs the MPS, then a possible replacement path is $\{v_0, v_2, v_4\}$, such that v_2 replaces v_0 and v_4 replaces v_2. Note that $\{v_0, v_2, v_4\}$ is not the only replacement path, e.g., $\{v_0, v_1, v_5\}$. Consequently, the nodes reconfigure themselves to recover connectivity, and the resulting graph becomes the initial graph without a noncritical node (e.g., v_4).

Algorithm 1 Message-passing strategy

Input: An arbitrary node p_0 from $\mathcal{G}$
Output: Connectivity maintenance in the removal of p_0
Assumption: Each node shares its unique node ID with its neighbors.
1: **initialization:** $p_i \leftarrow p_0$; $\mathcal{N}_{p_i} \leftarrow \mathcal{N}_{p_0}$; message$\leftarrow(p_0)$;
2: **if** $|\mathcal{N}_{p_0}| = 1$
3: no replacements required;
4: **else**
5: **while** $\mathcal{N}_{p_i} \setminus$ message $\neq \emptyset$
6: $p_{i+1} \leftarrow v$ s.t. $v \in \mathcal{N}_{p_i} \setminus$ message;
7: p_i sends message to p_{i+1} and replaces p_{i-1};
8: $p_i \leftarrow p_{i+1}$; $\mathcal{N}_{p_i} \leftarrow \mathcal{N}_{p_{i+1}}$; message$\leftarrow$(message, p_i);
9: **end while**
10: **end if**

B. Performance of MPS

Given a networked system, reactive schemes for connectivity maintenance result in some changes in the graph topology. While maintaining the graph connectivity, an important aspect is to avoid causing significant changes in the graph properties, such as the total number of edges or the maximum node degree. Note that the total number of edges and the maximum node degree can be directly related to the overall and individual communication loads, for which the increase is not desirable for a network system containing agents with limited energy capacity.

Proposition 2: A sequence of replacements along a maximal simple path, $(p_0, p_1, \ldots, p_k)$, on $\mathcal{G}$, such that any $p_{i+1} \in \mathcal{N}_{p_i} \setminus \{p_0, p_1, \ldots, p_i\}$ for $i < k$, guarantees no increase in the total number of edges and the maximum node degree in the face of an arbitrary node removal.

Proof: Let p_0 be the removed node, $\boldsymbol{p} = (p_0, p_1, \ldots, p_k)$ be the replacement path, and $\mathcal{G}'$ be the new graph structure after the replacements. Then, this corollary is proven in two parts:

1) In the removal of p_0, $\boldsymbol{p}$ results in $\mathcal{G}'$, which corresponds to the removal of p_k and its adjacent edges from $\mathcal{G}$. As a result, the total number of edges decreases as the agents are removed.

2) Let p_0 be the node that has the maximum degree $d_{\max}$ in $\mathcal{G}$. Based on $\boldsymbol{p}$, p_1 replaces p_0 after p_0 is removed. Now, if $k = 1$, then p_1 is a noncritical node that will not be replaced. As a consequence, the degree of p_1 becomes $d_{\max} - 1$ after the replacement. If $k \neq 1$, then p_1 will be replaced by p_2. Hence, the degree of p_1 becomes $d_{\max}$ after the replacements. In both cases, p_1 becomes the node with the maximum degree in $\mathcal{G}'$ after replacing p_0. Finally, let $\tilde{v}$ be the node with the maximum degree such that $\tilde{v} \neq p_0$. In the face of p_0 removal, either no replacements occur in the neighborhood of $\tilde{v}$ or the replacements in the neighborhood of $\tilde{v}$ may cause, at most, one reduction in $d_{\max}$. As a result, the maximum node degree in $\mathcal{G}'$ becomes either $d_{\max}$ or $d_{\max} - 1$. □

The optimal solution resulting in the minimum number of replacements for the RC problem can be obtained by a centralized controller by finding the shortest path between the removed node and a noncritical node on the graph. Note that such a centralized controller requires the complete information about the graph. The objective of the MPS is to solve the RC problem by only using some local and partial information. Due to using limited information, the MPS may not necessarily guarantee the optimal solution for any graphs. In the following results, we will discuss the performance of MPS.

Proposition 3: In any undirected connected graph, $\mathcal{G} = (V, E)$, the maximum number of replacements that can occur via the MPS is $|V| - 1$.

Proof: From Corollary 2, the MPS results in a message that is the sequence of nodes represented as a maximal simple path $\boldsymbol{p}$. Let $|\boldsymbol{p}| > |V|$, and then at least one node appears multiple times in $\boldsymbol{p}$;

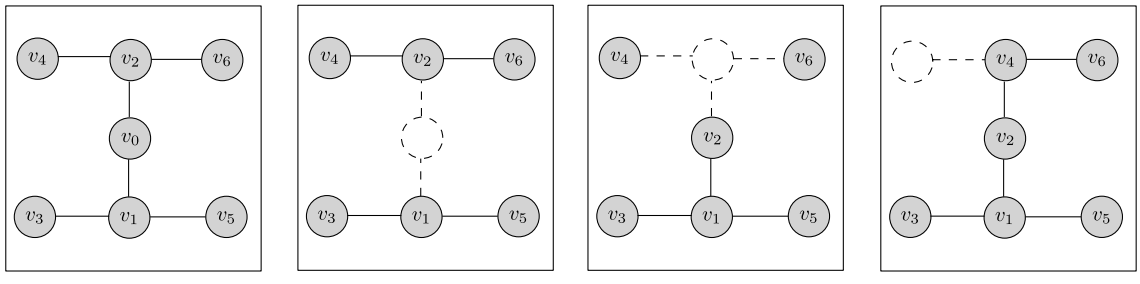


Fig. 5 Representations of a) an initially connected graph; b) v_0 leaving the system; c) v_2 replacing v_0; and d) v_4 replacing v_2.

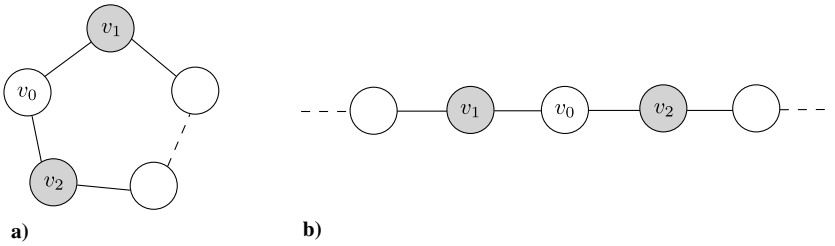


Fig. 6 Representations of a) an infinite cycle graph; b) an infinite path graph.

thus, $\boldsymbol{p}$ is not simple. This is a contradiction; hence, $|\boldsymbol{p}| \leq |V|$, which implies an upper bound for the number of replacements as $|V| - 1$. □

A tighter bound for Proposition 3 can be obtained in a tree graph, in which any two nodes are connected by exactly one simple path.

Proposition 4: In a tree graph, $\mathcal{G} = (V, E)$, the maximum number of replacements that can occur via the MPS is $\Delta - 1$, where Δ is the diameter of $\mathcal{G}$.

Proof: In a tree graph, a noncritical node is always a leaf node, and a critical node always has a degree of two. Thus, the diameter of a tree graph corresponds to the length of the longest maximal simple path. Let $\{p_0, p_1, \ldots, p_{\Delta-1}, p_\Delta\}$ denote to the longest maximal simple path, where p_0 and p_Δ are leaf nodes and the nodes in between are critical. If p_0 is the removed node, then the MPS does not initiate replacements. If p_1 is the removed node, then the maximum number of replacements based on the MPS may occur along the sequence $\{p_1, \ldots, p_{\Delta-1}, p_\Delta\}$, resulting in $\Delta - 1$ replacements. □

A biconnected graph is a connected graph that does not have any critical nodes, so the removal of any single node cannot disconnect the graph.

Remark 2: In biconnected graphs, the MPS cannot achieve the optimal solution for connectivity maintenance. Using the MPS, the removal of an arbitrary node initiates some replacements, even though none are necessary.

Note that the MPS may not always result in the minimum number of replacements in any node removal. For instance, if the removed node is not a leaf node, but is noncritical, the MPS still initiates a sequence of replacements as depicted in Remark 2. On the contrary, a centralized strategy certainly identifies the criticality of a node due to the availability of the overall graph structure.

Remark 3: The node criticality cannot be always determined with local information. As shown in Fig. 6, let $\mathcal{G}_1$ and $\mathcal{G}_2$ be an infinite cycle and infinite path graphs, respectively. Suppose that any node in $\mathcal{G}$ knows its δ-hop neighborhood. Let v_0 be the removed node. In Fig. 6, v_0 is noncritical in $\mathcal{G}_1$ but critical in $\mathcal{G}_2$. Note that, for any finite δ, v_0 has the same neighborhood in $\mathcal{G}_1$ and $\mathcal{G}_2$; hence, it cannot differentiate its criticality by just looking at its δ-neighborhood.

Since the criticality cannot be always determined locally, the MPS may sometimes initiate a sequence of replacements when a noncritical node is removed. This occurs due to the limitation of local information in the computations. However, it is important to emphasize that, for any undirected connected graph, connectivity maintenance in the presence of any node removal is guaranteed by the MPS by using only some local information. In this respect, for all $i \geq 0$, p_i selects the node p_{i+1} from $\mathcal{N}_{p_i} \setminus \{p_0, \ldots, p_i\}$ so that p_{i+1} will replace p_i. Here, a question arises as to which node from $\mathcal{N}_{p_i} \setminus \{p_0, \ldots, p_i\}$ should be selected to increase the efficiency of the MPS. For instance, a random selection scheme requires very little information to be shared among the nodes, or a node selection based on the minimum degree as introduced in [18] may capture the leaf node neighbors. Consequently, as the information possessed by a node and shared in the neighborhood increases, the solution is expected to approach the optimal solution.

C. δ-MPS

In this section, we introduce a variant of the MPS that uses δ-criticality information for each node. Here, the δ-criticality is defined as follows:

Definition 3 (δ-criticality): For any positive integer $\delta \geq 1$, a node v_i is called δ-critical if the subgraph, induced by the δ-neighborhood of v_i, is disconnected; otherwise, v_i is δ-noncritical. A node is zero-critical if its degree is equal to one.

Some examples of δ-criticality are illustrated in Fig. 7. For instance, v_0 in Fig. 7a is one-critical because the subgraph induced from its one-hop neighborhood (i.e., $\mathcal{N}^1_{v_0} = \{v_1, v_2, v_3\}$) is disconnected, as shown in Fig. 7b; whereas it is two-noncritical, as shown in Fig. 7c, because the subgraph induced from its two-hop neighborhood (i.e., $\mathcal{N}^2_{v_0} = \{v_1, v_2, v_3, v_4, v_5\}$) is connected.

Let $\mathcal{G} = (V, E)$ be a connected graph, and let $v_i \in V$ be δ-noncritical. Suppose that a simple path $\boldsymbol{p}^{nm}$ connects any arbitrary two nodes $v_n, v_m \in V$ and includes v_i as an intermediate node. In $\boldsymbol{p}^{nm}$, v_i appears between two of its neighbors, i.e., $\boldsymbol{p}^{nm} = (v_n, \ldots, v_j, v_i, v_k, \ldots, v_m)$. In the removal of v_i, there exists another path $\boldsymbol{p}^{jk}$ consisting of some nodes within δ-hops of v_i, since v_i is δ-noncritical by definition. Hence, the removal of v_i does not cause a disconnection between v_j and v_k as well as between v_n and v_m.

Remark 4: A δ-noncritical node is always noncritical in $\mathcal{G}$.

In light of Remark 4, δ-criticality is used in the MPS as in Algorithm 2. In this respect, each node knows whether itself and its immediate neighbors are δ-noncritical. Note that a node can obtain its own δ-criticality by sending a query to each node in its δ-neighborhood to understand who is linked to whom within the δ-neighborhood. In the δ-MPS, whenever a node p_i receives a message, it appends its own individual ID to the message likewise MPS. Then, it selects a δ-noncritical neighbor from the candidate set, $\mathcal{N}_{p_i} \setminus$

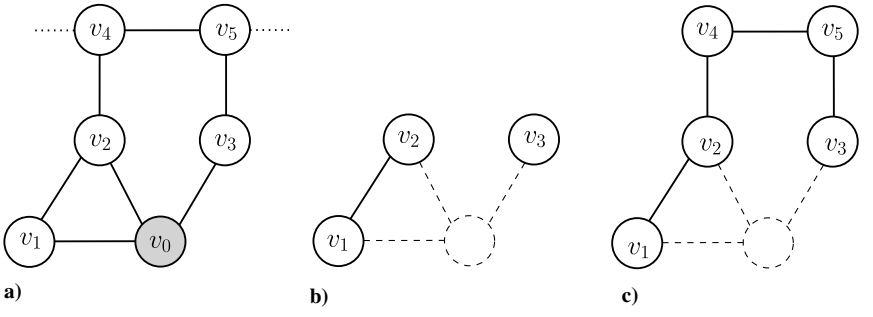


Fig. 7 Representations of a) v_0 being removed from the graph, b) v_0 as one-critical, and c) v_0 as two-noncritical.

Algorithm 2 δ-MPS

Input: An arbitrary node p_0 from $\mathcal{G}$
Output: Connectivity maintenance in the removal of p_0
Assumption: Each node shares both its ID and δ-criticality with its neighbors.
1: **initialization:** $p_i \leftarrow p_0$; $\mathcal{N}_{p_i} \leftarrow \mathcal{N}_{p_0}$; message$\leftarrow(p_0)$;
2: **if** p_0 is δ-noncritical
3: no replacements required;
4: **else**
5: **whereas** $\mathcal{N}_{p_i} \setminus$ message $\neq \emptyset$
6: **if** there exists a δ-noncritical node $v \in \mathcal{N}_{p_i} \setminus$ message;
7: $p_{i+1} \leftarrow v$ such that v is one of the δ-noncritical nodes;
8: **else**
9: $p_{i+1} \leftarrow v$ such that v is randomly selected from $\mathcal{N}_{p_i} \setminus$ message;
10: **end if**
11: p_i sends message to p_{i+1} and replaces p_{i-1};
12: $p_i \leftarrow p_{i+1}$; $\mathcal{N}_{p_i} \leftarrow \mathcal{N}_{p_{i+1}}$; message$\leftarrow$(message, p_i);
13: **end while**
14: **end if**

$\{p_0, \ldots, p_i\}$. In the case where the candidate set does not contain a δ-noncritical node, p_i selects a random node from the candidate set.

It has been shown in Remark 4 that a δ-noncritical node is globally noncritical in $\mathcal{G}$. Now, a question arises as to when a δ-critical node assures global criticality. In this respect, Proposition 5 presents a sufficient condition that guarantees global node criticality by relating δ to a graph structure.

Definition 4: A chordless cycle in $\mathcal{G}$ is a cycle such that no two nodes of the cycle are connected by an edge that does not itself belong to the cycle.

Proposition 5: Let $c_{\max}$ be the length of the longest chordless cycle in a graph. Then, any δ-critical node is globally critical in $\mathcal{G}$ if $\delta \geq \lfloor c_{\max}/2 \rfloor$, where $\lfloor \cdot \rfloor$ is the standard floor operator.

Proof: This proposition is proven in two parts:

Case 1 ($c_{\max} > 0$): Let $\delta \geq \lfloor c_{\max}/2 \rfloor$, and let v be a δ-critical node. Then, the graph $\mathcal{G}'$ induced by the nodes in $\mathcal{N}_v^\delta$ is disconnected after the removal of v. For the sake of contradiction, assume that v is a globally noncritical node in $\mathcal{G}$. Then, there exists $v_m, v_n \in \mathcal{N}_v^\delta$ such that there is no path between v_m and v_n on $\mathcal{G}' - \{v\}$, but there is a shortest path $\boldsymbol{p}^*$ between v_n and v_m on $\mathcal{G} - \{v\}$. Note that $\boldsymbol{p}^*$ contains at least one node outside $\mathcal{N}_v^\delta$. Let $\boldsymbol{p}^{vn}$ and $\boldsymbol{p}^{mv}$ denote the shortest paths from v to v_n and v_m to v, respectively. Then, $(\boldsymbol{p}^{vn}, \boldsymbol{p}^*, \boldsymbol{p}^{mv})$ is a chordless cycle. Let β be the length of this chordless cycle. Then, $\boldsymbol{p}^*$ contains a node that is $\lfloor \beta/2 \rfloor$ hops away from v, where $\lfloor \beta/2 \rfloor > \delta$. Since $c_{\max} \geq \beta$ by definition, we obtain $\lfloor c_{\max}/2 \rfloor \geq \lfloor \beta/2 \rfloor > \delta$. However, this contradicts with $\delta \geq \lfloor c_{\max}/2 \rfloor$.

Case 2 ($c_{\max} = 0$): In this case, $\mathcal{G}$ is a tree graph. In a tree graph, all the leaf nodes are noncritical and all the other nodes are globally critical. Furthermore, all the nodes in a tree, other than the leaf nodes, are δ-critical for any value of $\delta \geq 0$ (from Definition 3). Hence, any δ-critical node is globally critical. □

Corollary 3: If δ is selected based on Proposition 5, then the replacement sequence generated via the δ-MPS involves only one noncritical node, which is the last node on the replacement sequence.

Proof: Based on the δ-MPS, a message travels from a δ-critical node to a neighboring δ-critical node until finding a δ-noncritical node. If δ is selected based on Proposition 5, a δ-critical node is globally critical. Hence, the replacement sequence generated via δ-MPS contains only one noncritical node, which is the last node on the sequence. □

In executing δ-MPS, if δ is selected such that any δ-critical node is globally critical, then only globally critical nodes generate or forward replacement request messages. Nonetheless, this is not sufficient to ensure the minimum number of replacements to recover connectivity. For instance, if there is no δ-noncritical node in the immediate neighborhood of a node v, then v selects a neighbor randomly for its replacement. As such, the δ-MPS does not always guarantee a shortest path to a noncritical node due to such randomizations.

In Fig. 8, we illustrate two cases to discuss the optimality gap of the δ-MPS. For the tree graph shown in Fig. 8a, Proposition 5 suggests that, if $\delta \geq 0$, then δ-criticality implies global criticality. Assume that v_0 is removed from the graph, and let δ be one. From the perspective of v_0, selecting v_1 or v_2 is indifferent because v_0 can only see $\mathcal{G}_{\mathcal{N}^1}$, which contains the highlighted nodes in Fig. 8a. Hence, one-MPS can result in the shortest maximal simple path as (v_0, v_1, v_3) or the longest maximal simple path as $(v_0, v_2, \ldots, v_{n-1}, v_n)$. In Fig. 8b, assume that the longest chordless cycle is a triangle and $\delta = 2$ in accordance with Proposition 5. Like the previous case, v_0 is removed from the graph and selecting v_1 or v_4 is indifferent for v_0 (since v_1 and v_4 are both two-critical.) Note that selecting v_1 leads to a replacement sequence as (v_0, v_1, v_2), whereas selecting v_4 causes a much longer route as $(v_0, v_4, \ldots, v_{n-1}, v_n)$.

Consequently, the optimality of the δ-MPS depends on the lengths of the maximal simple paths between critical and noncritical nodes. On any graph $\mathcal{G}$, the optimality of the δ-MPS is quantified by the maximum possible deviation (in terms of the number of replacements) from the optimal centralized solution in any agent removal.

Remark 5: For any connected $\mathcal{G} = (V, E)$, let $V_c \subset V$ be the set of critical nodes. For each $v_i \in V_c$, let $l_{\max}(v_i)$ and $l_{\min}(v_i)$ be the lengths of the shortest and the longest maximal simple paths from v_i to any noncritical node, respectively. If δ is selected based on Proposition 5, then the δ-MPS deviates from the optimal centralized solution by, at most,

$$\max_{v_i \in V_c} (l_{\max}(v_i) - l_{\min}(v_i)) \tag{3}$$

D. Simulation Results

The main objective of the Monte Carlo simulations is to provide a statistical comparison for the performances of the MPS and δ-MPS with respect to the optimal centralized solution. Here, the centralized solution solves the RC problem with the minimum number of replacements by identifying the closest noncritical node to the removed node and then executing the replacements along the shortest path between them.

In all simulations, we consider connected random geometric graphs consisting of 50 nodes. The nodes are randomly and uniformly distributed on an area having a size of 75×75 units^2.

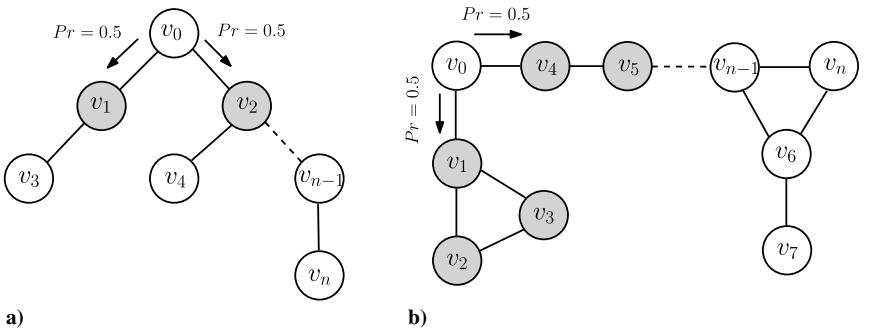


Fig. 8 Possible neighbor selections of v_0 via a) one-MPS, and b) two-MPS, which may lead to longest or shortest replacement sequences.

Table 1 Based on 200 simulations, the mean costs[a] of various strategies for connectivity maintenance in graphs with 50 nodes

Graph properties			Mean cost[a]				
R (unit)	Mean $\bar{d}$[b]	Mean Δ	Centralized	RCS	MPS	One-MPS	Two-MPS
8	2.00	38.45	5.275	12.025	11.695	10.915	10.755
9	2.09	30.62	2.745	8.370	7.470	5.790	5.455
10	2.19	26.35	1.710	6.040	5.490	3.240	2.810
11	2.65	20.97	0.725	6.330	4.570	0.895	0.835
12	3.55	16.41	0.330	8.400	7.400	0.595	0.405
13	4.11	14.94	0.225	9.585	8.720	0.410	0.305
14	4.56	13.29	0.170	11.880	9.770	0.330	0.240
15	5.14	11.64	0.120	14.930	12.085	0.240	0.205
16	5.56	10.26	0.050	16.330	14.005	0.210	0.125
17	6.47	9.02	0.040	19.545	16.935	0.165	0.075
18	7.10	8.17	0.030	23.985	19.165	0.120	0.065
19	7.80	7.49	0.025	24.125	20.340	0.105	0.050
20	8.50	6.85	0.010	29.690	23.790	0.050	0.030

[a]Number of agent replacements.
[b]Average node degree.

Different graph properties are obtained by varying the radius of connection R, which is assumed to be the same for each agent. For any given R, 200 connected graphs are generated. In each simulation, a randomly selected node is removed from the graph, and the RC problem is solved via a centralized controller, then via the replacement control strategy (RCS) [18], then via the MPS, and then via the δ-MPS for $\delta = 1, 2$. The strategies are compared based on the resulting costs (i.e., the number of agent replacements). The results are illustrated in Table 1, which shows the mean costs of the strategies to solve the RC problem for various graph structures. Moreover, the distributions of the graph properties and the costs of the replacements are displayed for the cases of $R = 8$, 12, and 16 in Figs. 9, 10, and 11, respectively.

For graphs consisting of equal numbers of nodes, as the radius of connection increases, it is more likely to have nodes with more connections, which results in smaller graph diameters. Such graphs are, in general, more robust to agent removal, and the average number of replacements needed to recover connectivity is reduced. This is illustrated in Figs. 9, 10, and 11, where the cost distribution of the centralized solution is more skewed toward zero as the diameter decreases.

Simulations indicate an interesting result when the RCS and MPS are compared with each other. Even though the RCS is using more information than the MPS (i.e., each agent running the RCS shares its degree and ID in its neighborhood, whereas each agent using the MPS shares only its ID), it performs slightly worse than the MPS. We justify this result as follows: suppose that the removed agent is p_0 and the closest noncritical agent to p_0 is p_k, for which the distance to p_0 is k. Let $\boldsymbol{Pr}_{\mathrm{MPS}}$ and $\boldsymbol{Pr}_{\mathrm{RCS}}$ denote the probabilities of finding a k-length replacement path as $(p_0, p_1, \ldots, p_k)$ via the MPS and RCS, respectively. As such,

$$0 < \prod_{0<i<k-1} \frac{1}{|\mathcal{N}_{p_i}|} \leq \boldsymbol{Pr}_{\mathrm{MPS}} \leq \frac{1}{|\mathcal{N}_{p_0}|} \tag{4}$$

$$0 \leq \boldsymbol{Pr}_{\mathrm{RCS}} \leq 1 \tag{5}$$

where $\mathcal{N}_{p_i}$ is the neighbor set of agent p_i. For example, if p_k is a leaf node (i.e., $k = 1$), the randomized nature of the MPS causes the selection of p_k with a probability of $1/|\mathcal{N}_{p_0}|$, whereas the RCS certainly selects p_k. Nonetheless, there can be cases where $\boldsymbol{Pr}_{\mathrm{RCS}}$ becomes zero due to the strict selection of minimum degree neighbor, whereas $\boldsymbol{Pr}_{\mathrm{MPS}}$ is always bounded away from zero. For instance, let p_0 in Fig. 12 be the removed node. Note that the gray nodes are the closest noncritical nodes to p_0. Accordingly, p_0 running the RCS never creates a message that ends with one of the gray nodes, since p_0 tends to send the replacement message to the node with a degree of two. On the contrary, p_0 using the MPS has a probability of 0.5 to create a message ending with one of the gray nodes.

In the MPS and RCS, a sequence of replacements is always executed unless a leaf node is removed from the graph. Thus, the MPS and RCS may result in a large number of unnecessary replacements, especially when the majority of the nonleaf nodes are noncritical (e.g., well-connected graphs). As is seen from Table 1, the mean costs of the MPS and RCS dramatically increase as the radius of connection (i.e., R) increases. On the other hand, the results show that using δ-criticality, even for $\delta = 1$, significantly improves the performance of the MPS. Moreover, the simulations illustrate that the increase in δ improves the performance. This is expected, since higher values of δ imply that more information about the graph

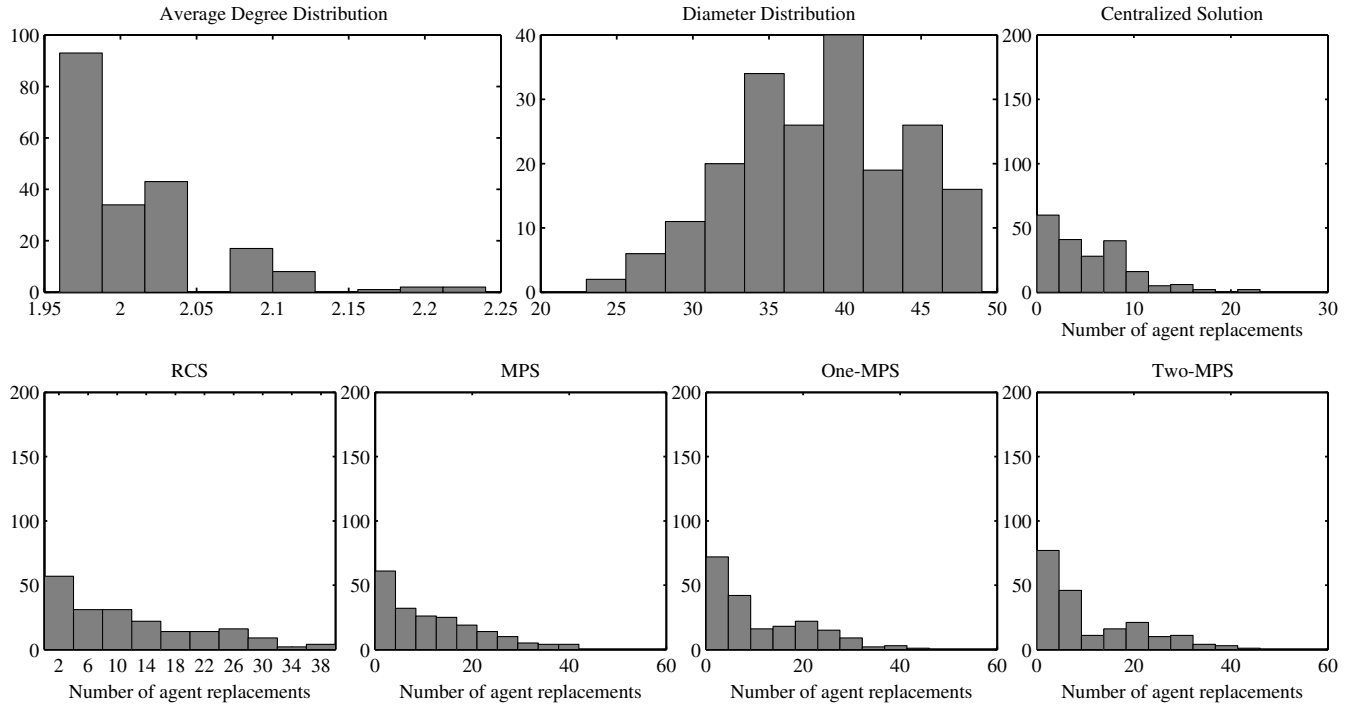


Fig. 9 For $R = 8$, the distributions of graph properties and mean costs for various strategies to solve the RC problem.

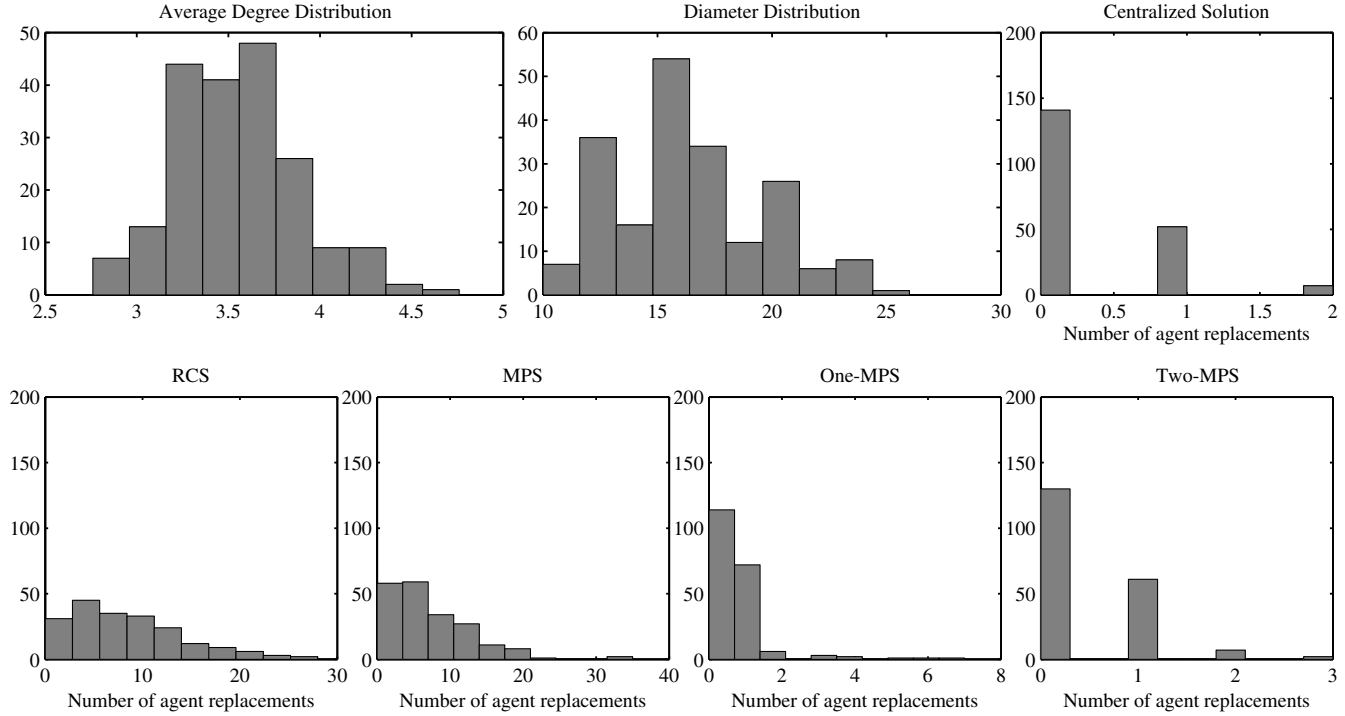


Fig. 10 For $R = 12$, the distributions of graph properties and mean costs for various strategies to solve the RC problem.

topology is available to each agent. In particular, the effect of δ on the performance becomes more significant as the sparsity of the inspected graph increases. Finally, we observe that using the one-MPS over the MPS results in a great performance improvement in Table 1, whereas using the two-MPS over the one-MPS does not improve the optimality significantly. Hence, the simulation results suggest that the one-MPS is an efficient and local approach to solve the RC problem.

V. Decentralized Connectivity Maintenance with Constrained Agents

Given a connected graph $\mathcal{G}$, where the nodes in $\mathcal{G}$ represent the agents, the δ-MPS results in a sequence of replacements to recover connectivity in the face of an arbitrary node removal. However, the δ-MPS assumes that any node in $\mathcal{G}$ will pursue a replacement once a replacement request message is received. In this section, we take into account the cases where a node in $\mathcal{G}$ may not execute a replacement due to some ongoing individual operation or a constraint based on its location on $\mathcal{G}^*$. In this paper, such nodes are called constrained nodes and defined as follows:

Definition 5 (node constraint): A node in $\mathcal{G}$ is constrained if it cannot be involved in a replacement sequence unless it is the removed node. Otherwise, it is unconstrained.

To ensure a sequence of replacements among the unconstrained nodes, a node should not move immediately after it sends a replacement request message as in the δ-MPS. Let v_0 be the removed node,

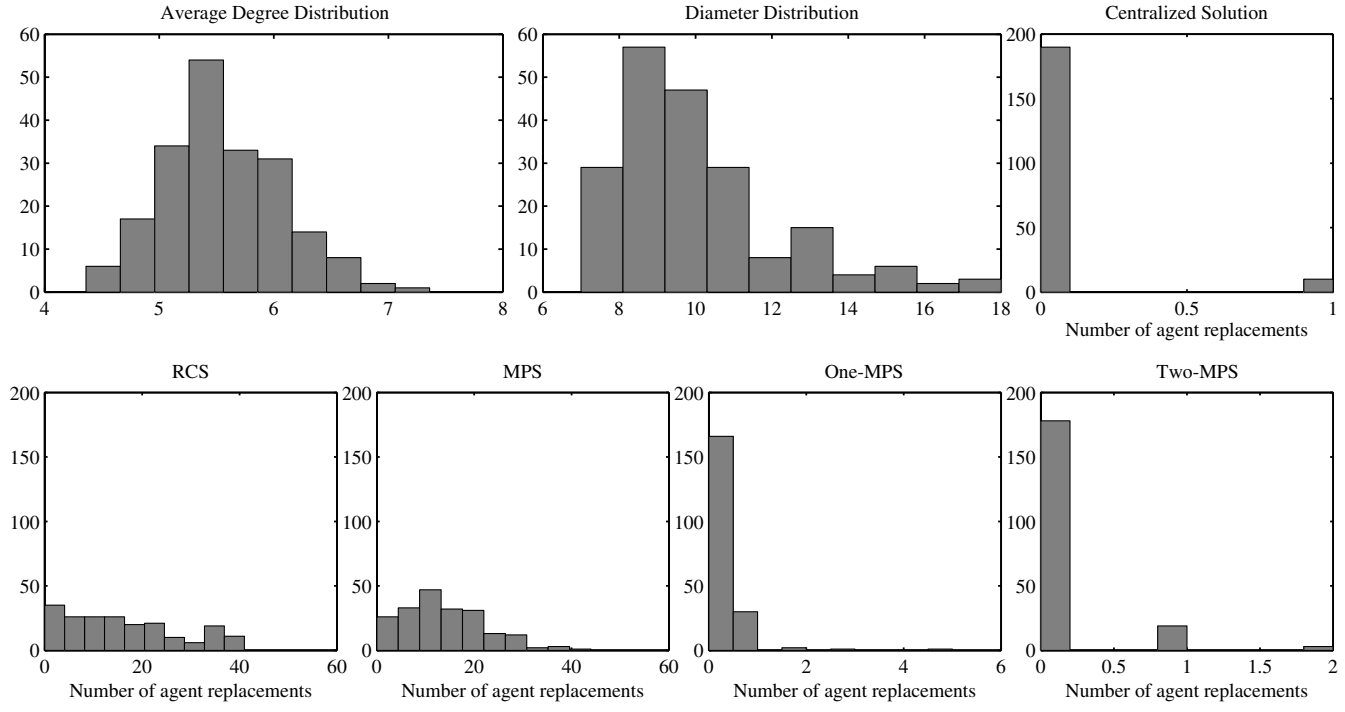


Fig. 11 For $R = 16$, the distributions of graph properties and mean costs for various strategies to solve the RC problem.

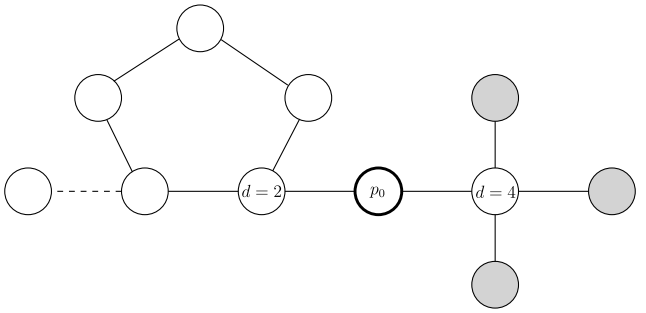


Fig. 12 Representation of a removed node, p_0, and the closest leaf nodes highlighted in gray.

and let v_3 be the constrained node in Fig. 13a. Suppose that v_0 initiates the replacements by sending a replacement request message to v_1 and it leaves the system immediately. Then, v_1 sends the replacement request message to v_2 according to the δ-MPS and replaces v_0. Similarly, v_2 sends the replacement request message to v_3, which is its only neighbor that has not received the message earlier. Accordingly, v_2 replaces v_1 by assuming that v_3 will eventually replace v_2. However, v_3 is a constrained node that will not move from its current position. As a result, the replacements result in a reconfiguration, as in Fig. 13b, illustrating a graph disconnection. To avoid such cases, the feasibility of a replacement sequence should be verified before the replacements are actually executed. As such, we propose a variant of the δ-MPS, namely, *the constrained* δ-MPS, which guarantees connectivity maintenance, even in the presence of constrained nodes.

A. Constrained δ-MPS

In the constrained δ-MPS for $\delta \geq 0$, each node knows its own constraint in addition to its individual ID and δ-criticality. Moreover, it can share this information with its immediate neighbors. Similar to the δ-MPS, the message passing for replacement request initiates with the removal of a δ-critical node and continues until it reaches a δ-noncritical node. Different than the δ-MPS, 1) there are two types of messages, namely, the request and the approval messages; 2) the request message contains the request sequence as well as the unfeasible node set; 3) a node moves only if an approval message is received; 4) if the removed node is δ-noncritical but constrained, the message passing initiates; 5) if a δ-critical constrained node receives a request message, it forwards it without including its individual ID to the message; and 6) if a δ-noncritical constrained node receives a request message, the replacements are not executed. Instead, it modifies the message by deleting its ID from the request sequence, including its ID to the unfeasible node set, and sends the message to the last node in the request sequence (i.e., the node that has previously sent the message to itself). This process continues until the message reaches a node that has a neighbor not in the request sequence and not in the unfeasible node set.

The pseudocode of the constrained δ-MPS is presented in Algorithm 3. Let p_0 be the removed node. If it is δ-noncritical and unconstrained, no replacements are executed (lines 2–3). Otherwise, the process is as follows: For any $i \geq 0$, suppose that p_i receives (or generates if $i = 0$) a message containing a request sequence, Request $= (p_0, \ldots, p_{i-1})$, and an unfeasible node set, notFeasible $\subseteq V$. If there exists a δ-noncritical and unconstrained node in $\mathcal{N}_{p_i} \setminus$ (Request $\cup$ notFeasible), then p_{i+1} is selected as one of the δ-noncritical and unconstrained nodes in that set (line 8). Otherwise, p_{i+1} is selected randomly (line 10). Then, p_i sends the message to p_{i+1}.

If p_{i+1} is unconstrained, it modifies the request sequence by adding its individual ID (line 14). Otherwise, it does not change the message and possibly sends the message as received to another node in the next time step (equivalent to message forwarding). Furthermore, if $p_i + 1$ is a constrained node and does not have any other neighbors to send the message (lines 15 and 16), it includes its ID to the unfeasible node set (line 17), deletes its ID from the request sequence (line 17), and sends the modified message to the last node in the request sequence (line 18). Accordingly, the message passing continues until it reaches a node that has a neighbor not in the request sequence and not in the unfeasible node set. Note that, in the worst case, the while loop between lines 16 and 19 leads to sending back the modified message to the removed node. In that case, the removed node selects another neighbor to initiate the message passing. Finally, the main while loop (between lines 6 and 21) breaks in two cases: 1) whenever all nodes are in the unfeasible node set; and 2) the message is received by a δ-noncritical unconstrained node. Accordingly, the approval message is created based on the request sequence (line 22). Then, as the approval messages are passed among the agents, the replacements are executed (line 24). Eventually, p_0 is removed from the graph (line 28).

Definition 6 (connected component): A connected component is a maximal connected subgraph of $\mathcal{G}$.

Remark 6: Each node belongs to exactly one connected component of $\mathcal{G}$.

Lemma 1: Let v' be any critical node in a connected graph $\mathcal{G}$. If the removal of v' partitions $\mathcal{G}$ into N connected components, then each connected component has at least one noncritical node.

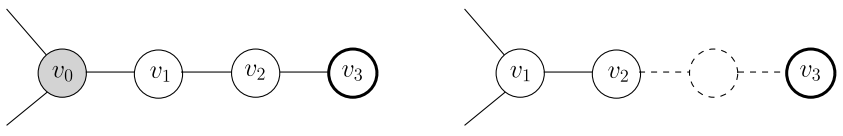
Proof: Let $\mathcal{G}_1, \ldots, \mathcal{G}_N$ denote the connected components of $\mathcal{G}$ after the removal of v_i. Note that, for each $\mathcal{G}_i$, at least one node in $\mathcal{G}_i$ is an immediate neighbor of v'. Let $(p_0, p_1, \ldots, p_m)$ be a maximal simple path where $p_0 = v'$ and any p_{i+1} is selected from $\mathcal{N}_{p_i} \setminus \{p_0, p_1, \ldots, p_i\}$, as in the MPS. Note that, if the MPS is initiated by v' and p_1 is a node in $\mathcal{G}_i$, then the rest of the nodes on the sequence are also in $\mathcal{G}_i$. Moreover, the last node on the sequence is noncritical because the MPS always stops at a noncritical node, as shown in Proposition 1. This implies that any $\mathcal{G}_i$ for $1 \leq i \leq N$ contains at least one noncritical node. □

Lemma 2: Let $\mathcal{G} = (V, E)$ be a connected graph, and let V^{nc} be the set of noncritical nodes in $\mathcal{G}$. For any node set $X \subset V$, let $Y \supseteq X$ be the set of nodes in a smallest connected subgraph containing X. Then, $V^{\text{nc}} \nsubseteq Y$, if $Y \subset V$.

Proof: If $Y \subset V$, then there are two possible cases for $V \setminus Y \neq \emptyset$:
1) If $(V \setminus Y) \subseteq V^{\text{nc}}$, then $V^{\text{nc}} \nsubseteq Y$ is an immediate result.
2) If $(V \setminus Y) \nsubseteq V^{\text{nc}}$, then there exists a critical node v' in $V \setminus Y$. Note that the removal of v' results in at least two connected components. Moreover, one of those connected components, $\mathcal{G}_i$, does not contain any nodes from Y. Following Lemma 1, $\mathcal{G}_i$ has at least one noncritical node. Hence, if $V \setminus Y$ has a critical node, then it also contains a noncritical node. Thus, $V^{\text{nc}} \nsubseteq Y$. □

Theorem 2: Let $\mathcal{G} = (V, E)$ be a connected graph, let V' be the set of all the constrained nodes in $\mathcal{G}$, and let $Y \supseteq V'$ be the set of nodes in a smallest connected subgraph containing V'. Assume that δ is selected according to Proposition 5. For any single node removal, the constrained δ-MPS recovers connectivity without violating the constraints if, and only if, $Y \subset V$.

Proof: First, we will show that, if the constrained δ-MPS recovers connectivity without violating the constraints, then $Y \subset V$. For the sake of contradiction, assume that $V = Y$ before the node removal.



a) Before replacements **b) After replacements**

Fig. 13 Representation of the δ-MPS not guaranteeing connectivity maintenance in the face of a node removal (e.g., v_0) if there exists a constrained node (e.g., v_3).

Algorithm 3 Constrained δ-MPS

Input: An arbitrary node p_0 from $\mathcal{G}$
Output: Connectivity maintenance in the removal of p_0
Assumption: Each node shares its unique node ID, δ-criticality, and constraint with its neighbors.
1: **initialization:** $p_i \leftarrow p_0$; $\mathcal{N}_{p_i} \leftarrow \mathcal{N}_{p_0}$; RequestSeq$\leftarrow(p_0)$; ApprovalMes$\leftarrow\emptyset$; notFeasible$\leftarrow\emptyset$;
2: **if** p_0 is a δ-noncritical and unconstrained node
3: no replacements required;
4: **else**
5: RequestMes$\leftarrow$(RequestSeq, notFeasible)
6: **while** $\mathcal{N}_{p_i} \setminus$ (RequestSeq $\cup$ notFeasible) $\neq \emptyset$
7: **if** there exists a δ-noncritical unconstrained node $v \in \mathcal{N}_{p_i} \setminus$ (RequestSeq $\cup$ notFeasible)
8: $p_{i+1} \leftarrow v$ such that v is one of the δ-noncritical unconstrained nodes; **break**;
9: **else**
10: $p_{i+1} \leftarrow v$ such that v is randomly selected from $\mathcal{N}_{p_i} \setminus$ (RequestSeq $\cup$ notFeasible);
11: **end if**
12: p_i sends RequestMes to p_{i+1}; $p_i \leftarrow p_{i+1}$; $\mathcal{N}_{p_i} \leftarrow \mathcal{N}_{p_{i+1}}$;
13: **if** p_i is an unconstrained node
14: RequestSeq$\leftarrow$(RequestSeq, p_i);
15: **else**
16: **while** $\mathcal{N}_{p_i} \setminus$ (RequestSeq $\cup$ notFeasible) $= \emptyset$
17: notFeasible$\leftarrow$notFeasible $\cup \{p_i\}$; delete p_i from RequestSeq;
18: p_i sends RequestMes to p_{i-1}; $p_i \leftarrow p_{i-1}$;
19: **end while**
20: **end if**
21: **end while**
22: ApprovalMes$\leftarrow$RequestSeq; $n = |$ApprovalMes$|$;
23: **while** $n > 1$
24: p_n sends ApprovalMes to p_{n-1} and replaces p_{n-1};
25: $n \leftarrow n - 1$;
26: **end while**
27: **end if**
28: p_0 is removed;

Then, $\mathcal{G}$ corresponds to the smallest connected graph containing all the nodes in V', which implies that all noncritical nodes are in V'. Thus, if δ is selected according to Proposition 5, then any node removal in $\mathcal{G}$ initiates the message passing via the constrained δ-MPS. However, the replacement request message never reaches a noncritical unconstrained node because such nodes do not exist in $\mathcal{G}$. Accordingly, the approval message will contain only the removed node. Hence, there will not be any replacements, and either a critical node or a constrained node will be removed from the graph.

Now, we will show that, if $Y \subset V$, then the constrained δ-MPS recovers connectivity without violating the constraints. Following Lemma 2, if $Y \subset V$, there always exists a noncritical unconstrained node v_{nc} in $V \setminus Y$. Since $\mathcal{G}$ is initially connected, there always exists a simple path from any node in $V \setminus \{v_{\text{nc}}\}$ to v_{nc}. For the sake of contradiction, suppose that the constrained δ-MPS does not recover connectivity without violating the constraints. Then, when Algorithm 3 reaches line 22, all the nodes other than p_0 are in the unfeasible node set. However, in order to be added to the unfeasible node set, a node first receives a request message. However, if v_{nc} receives the request message, then line 8 in Algorithm 3 is executed and the main while loop is broken. Hence, v_{nc} can never be added to the unfeasible node set, which leads to a contradiction. □

B. Complexity

In practical scenarios, a message-passing algorithm becomes applicable if the number of messages to be sent by the nodes does not dramatically increase with respect to the graph topology. Increasing the number of messages over a graph increases the information flow over the graph, which possibly results in optimal solutions. However, it also increases the time delays, energy consumption, susceptibility to disturbances on communication, etc. Hence, there is usually a tradeoff between the communication load and the optimality in cooperative networked systems. In Theorem 3, we provide an upper bound on the number of messages via the constrained δ-MPS.

Theorem 3: Let $\mathcal{G} = (V, E)$ be a connected graph. After an arbitrary single node removal, the overall number of messages to recover connectivity via the constrained δ-MPS is, at most, $2|V| - 2$.

Proof: In the constrained δ-MPS, the following holds:

1) If a request message does not reach a δ-noncritical constrained node, each node receives only one request message and one approval message. Accordingly, the request message flows along a simple path from the removed node to a δ-noncritical unconstrained node, and the approval message traverses this path in the opposite direction.

2) If a request message reaches a δ-noncritical constrained node, a refusal message is created and such a node never gets another message, since all the unfeasible nodes are recorded in the refusal message. This mainly prevents any cyclic travel of the message over the graph.

Thus, in both cases, the messages are transferred on the edges of the tree, for which the nodes have received a request message. In the worst case, such a tree contains all the nodes in the graph, so it can have, at most, $|V| - 1$ edges. Note that each edge is traversed only twice. Hence, the overall messages passed among the nodes cannot exceed $2|V| - 2$. □

C. Simulation Studies

In this section, we perform a simulation study to compare the resulting performances of the δ-MPS and the constrained δ-MPS. Consider a rectangular surveillance area with the dimensions of 60×50 km, which involves six regions of interest and two bases, as in Fig. 14. Assume that each region is monitored by a single UAV at the beginning of the mission, and let each UAV have a maximum communication range of 20 km. In this setting, it is desired to have a connected communication network between the UAVs and the bases for efficient data processing. To this end, we model the bases as some pseudo-UAVs, which mainly correspond to the constrained nodes in the communication network. Based on the described setting, a randomly selected UAV is removed from the surveillance area at each time step, and the evolutions of the communication network via no control strategy, one-MPS, and the constrained one-MPS are inspected as illustrated in Figs. 15–18.

At $t = 1$, UAV 5 leaves the surveillance area. As seen from Fig. 15, using no control strategy causes a graph disconnection because it is a critical agent. In one-MPS, UAV 5 creates a request message, sends it to UAV 4, and leaves the surveillance area. UAV 4 modifies the message, sends it to the pseudo-UAV 3, and replaces UAV 5. Since the

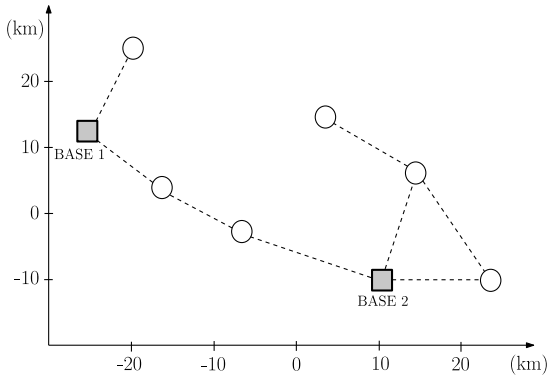


Fig. 14 Surveillance area involving two bases and six regions of interest.

pseudo-UAV 3 is the constrained agent, it does not move, and the message passing stops. As illustrated, executing replacements immediately without checking the feasibility causes a graph disconnection. On the other hand, the constrained one-MPS recovers the connectivity with two replacements, such as the replacement of UAV 5 by UAV 4 and the replacement of UAV 4 by UAV 2. Note that UAVs 4 and 2 were not directly connected to each other at the beginning. However, when the pseudo-UAV 3 receives the request message from UAV 4, it forwards the message to UAV 2 due to using the constrained one-MPS.

At $t = 2$, UAV 6 leaves the surveillance area. Based on the one-MPS, one replacement is executed, i.e., UAV 7 replaces UAV 6. Since the graph is disconnected at $t = 1$, it is still disconnected at $t = 2$, as shown in Fig. 16. However, the number of connected components via the one-MPS does not increase after the replacements. In other words, UAVs 1, 4, 7, and 8 still stay connected after the removal of UAV 7. Similarly, the replacement of UAV 6 by UAV 7 occurs via the constrained one-MPS. Since the graph was connected at $t = 1$, it stays connected at $t = 2$. At $t = 3$, UAV 2 leaves the surveillance area. No replacements occur via the one-MPS. On the other hand, the constrained one-MPS results in three replacements, i.e., UAV 2 is replaced by UAV 4, UAV 4 is replaced by UAV 7, and UAV 7 is replaced by UAV 1, as shown in Fig. 17. Note that, by using the constrained one-MPS, the pseudo-UAV 3 never receives a request message from UAV 2 because it is noncritical and constrained. Moreover, we observe a replacement between UAVs 4 and 7, which is due to the message forwarding via the pseudo-UAV 8.

At $t = 4$, UAV 4 leaves the surveillance area and two replacements occur via the constrained one-MPS, i.e., UAV 7 replaces UAV 4 and UAV 1 replaces UAV 7, as illustrated in Fig. 18. Note that the connectivity is not ensured by the no-control strategy and the one-MPS in the presence of multiple constrained agents. On the other hand, the constrained one-MPS recovers the connectivity until $t = 4$. However, if a UAV leaves at $t = 5$, then the constrained one-MPS cannot recover the connectivity, since there would not be sufficient UAVs to maintain a connected network between the bases (as shown in Theorem 2).

Remark 7: Let n_T and n_B be the number of agents and the number of bases in the environment, respectively. Let k be the number of nodes in the smallest connected component containing all the bases. If the bases are modeled as the constrained agents in the communication network and $n_T - 1 \geq k - n_B$, then executing the

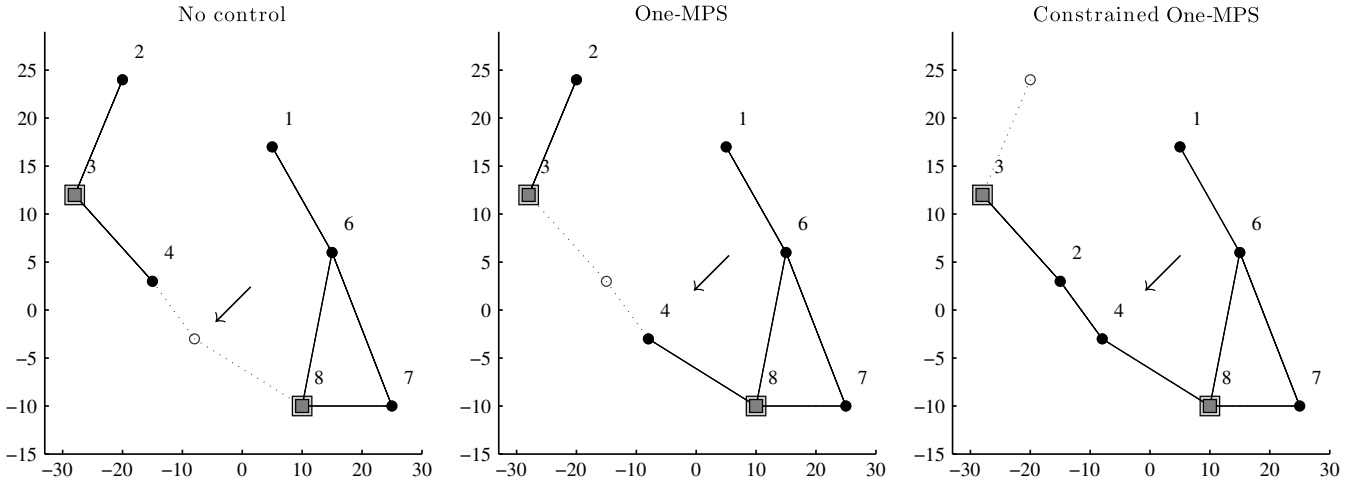


Fig. 15 UAV 5, for which the location at $t = 0$ is shown by the arrow, is removed at $t = 1$.

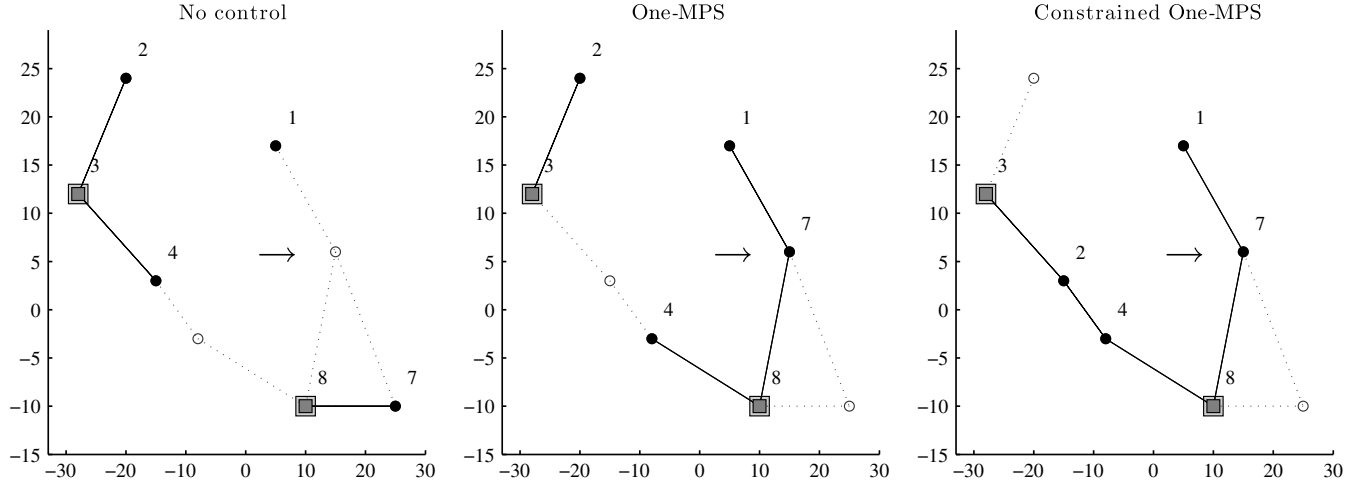


Fig. 16 UAV 6, for which the location at $t = 1$ is shown by the arrow, being removed at $t = 2$.

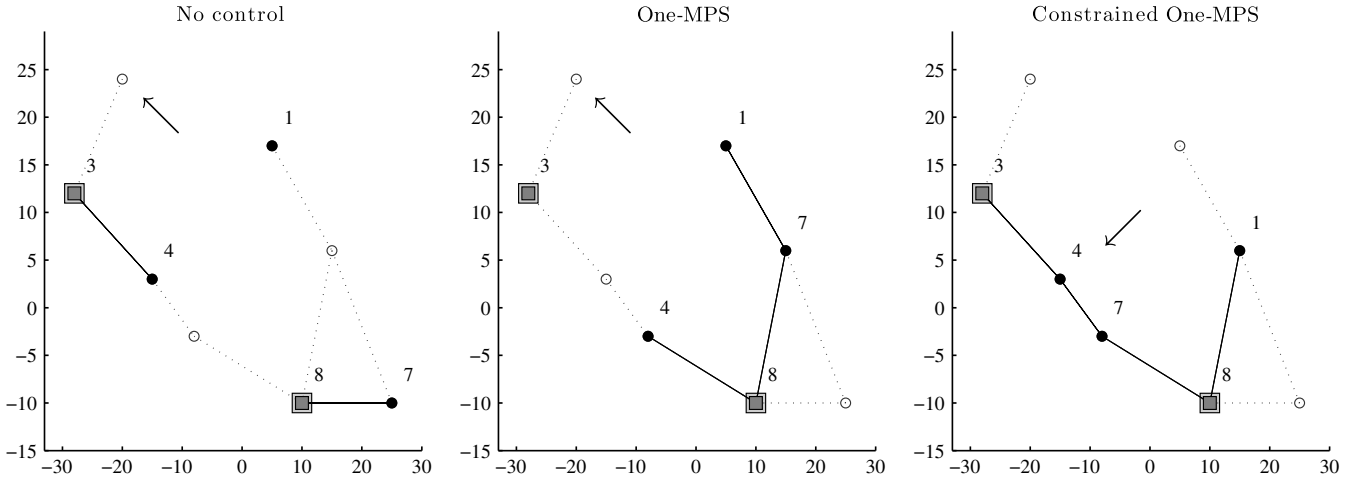


Fig. 17 UAV 2, for which the location at $t = 2$ is shown by the arrow, being removed at $t = 3$.

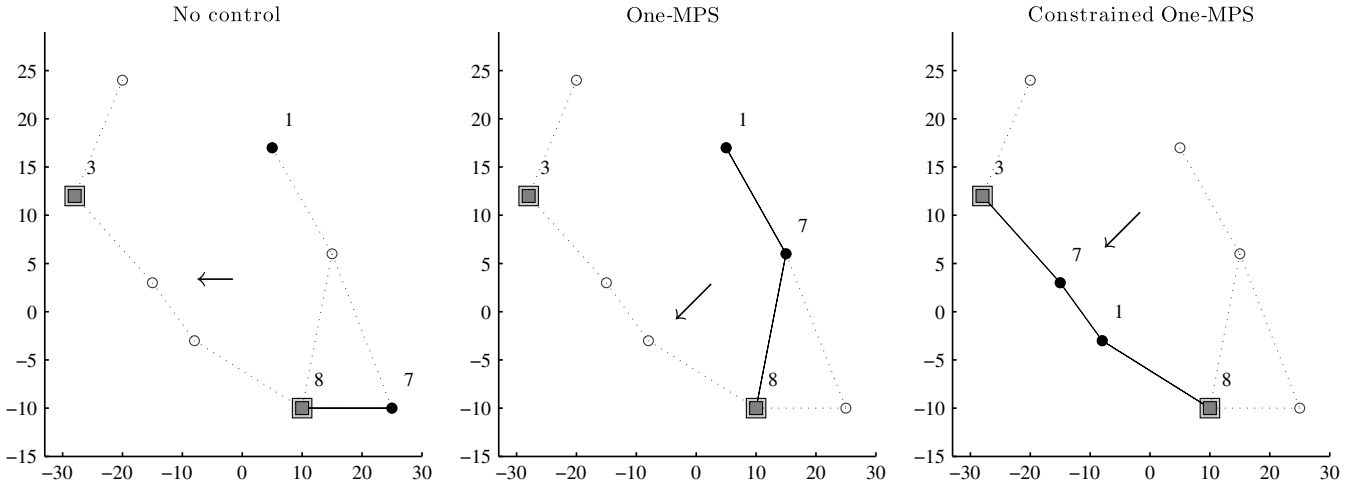


Fig. 18 UAV 4, for which the location at $t = 3$ is shown by the arrow, being removed at $t = 4$.

constrained δ-MPS ensures a connected network among the bases and the agents after any agent removal.

VI. Conclusions

In this paper, the connectivity of networked systems in the face of arbitrary agent removals was addressed, and a decentralized connectivity recovery strategy was proposed. It was shown that the message-passing strategy (MPS) recovered the graph connectivity through a sequence of replacements for any initially connected network if a single agent was removed at a time and the consecutive removals were sufficiently spaced in time, i.e., all the replacements were completed before the next agent removal. The replacements were initiated by the removed agent, and they ended with an agent for which the relocation did not cause a disconnection. The optimality gap of the MPS was discussed through some Monte Carlo simulations by comparing it to the optimal (the minimum number of replacements) centralized solution. Although the MPS recovered connectivity even in the case of agents using a minimum amount of information (i.e., only neighbor IDs.), it was observed that incorporating a local criticality notion (i.e., δ-criticality) in the decision mechanism significantly improved the resulting performance. As such, the δ-MPS was proposed as a variant of the MPS. An underlying assumption of the MPS and δ-MPS was that an agent executed a replacement once a request message was received. However, in some scenarios, some of the agents might reject a replacement request (e.g., due to occupying a special location or reaching a critical state). Therefore, the constrained δ-MPS was proposed to deal with such scenarios, where the constrained agents were not involved in the replacement sequence for the recovery of connectivity. Finally, the constrained δ-MPS was implemented in a multi-UAV surveillance scenario, and the recovery of connectivity among a set of UAVs and some bases in the face of arbitrary UAV removals was demonstrated.

References

[1] Zavlanos, M. M., Egerstedt, M. B., and Pappas, G. J., "Graph-Theoretic Connectivity Control of Mobile Robot Networks," *Proceedings of the IEEE*, Vol. 38, No. 11, 2011, pp. 1525–1540. doi:10.1109/JPROC.2011.2157884

[2] Mesbahi, M., and Hadaegh, F. Y., "Formation Flying Control of Multiple Spacecraft via Graphs, Matrix Inequalities, and Switching," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 2, 2001, pp. 369–377. doi:10.2514/2.4721

[3] Intanagonwiwat, C., Govindan, R., and Estrin, D., "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom '00)*, ACM, New York, 2000, pp. 56–67. doi:10.1145/345910.345920

[4] Olfati-Saber, R., "Distributed Tracking for Mobile Sensor Networks with Information-Driven Mobility," *American Control Conference (ACC '07)*, IEEE Publ., Piscataway, NJ, 2007, pp. 4606–4612. doi:10.1109/ACC.2007.4282261

[5] White, B., Tsourdos, A., Ashokaraj, I., Subchan, S., and Zbikowski, R., "Contaminant Cloud Boundary Monitoring Using Network of UAV Sensors," *IEEE Sensors Journal*, Vol. 8, No. 10, 2008, pp. 1681–1692. doi:10.1109/JSEN.2008.2004298

[6] Beard, R. W., McLain, T. W., Nelson, D. B., Kingston, D., and Johanson, D., "Decentralized Cooperative Aerial Surveillance Using Fixed Wing Miniature UAVs," *Proceedings of the IEEE*, Vol. 94, No. 7, 2006, pp. 1306–1324. doi:10.1109/JPROC.2006.876930

[7] Li, X., and Xi, Y., "Distributed Connected Coverage Control for Groups of Mobile Agents," *International Journal of Control*, Vol. 83, No. 7, 2010, pp. 1347–1363. doi:10.1080/00207171003736279

[8] Dai, R., Maximo, J., and Mesbahi, M., "Formation of Connected Networks for Fractionated Spacecraft," *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2012-5047, 2012. doi:10.2514/6.2012-5047

[9] Zhou, J., Hu, Q., and Friswell, M. I., "Decentralized Finite Time Attitude Synchronization Control of Satellite Formation Flying," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 1, 2013, pp. 185–195. doi:10.2514/1.56740

[10] Ponda, S. S., Johnson, L. B., Choi, H.-L., and How, J. P., "Ensuring Network Connectivity for Decentralized Planning in Dynamic Environments," *AIAA Infotech@ Aerospace Conference*, AIAA Paper 2011-1462, 2011. doi:10.2514/6.2011-1462

[11] Casbeer, D. W., Swindlehurst, A. L., and Beard, R., "Connectivity in a UAV Multi-Static Radar Network," *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2006-6209, 2006. doi:10.2514/6.2006-6209

[12] Valenti, M., Dale, D., and How, J., "Mission Health Management for 24/7 Persistent Surveillance Operations," *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2007-6508, 2007. doi:10.2514/6.2007-6508

[13] Motevallian, S. A., Yu, C. B., and Anderson, B., "Robustness to the Loss of Multiple Nodes in the Localizability of Sensor Networks," *Proceedings of the IFAC World Congress*, Vol. 18, No. 1, 2011, pp. 7836–7841. doi:10.3182/20110828-6-IT-1002.03415

[14] Summers, T. H., Yu, C., and Anderson, B., "Addressing Agent Loss in Vehicle Formations and Sensor Networks," *International Journal of Robust and Nonlinear Control*, Vol. 19, No. 15, 2009, pp. 1673–1696. doi:10.1002/rnc.v19:15

[15] Yazicioglu, A., Egerstedt, M., and Shamma, J., "Decentralized Formation of Random Regular Graphs for Robust Multi-Agent Networks," *53rd IEEE Conference on Decision and Control (CDC)*, IEEE Publ., Piscataway, NJ, 2014, pp. 595–600. doi:10.1109/CDC.2014.7039446

[16] Akkaya, K., Thimmapuram, A., Senel, F., and Uludag, S., "Distributed Recovery of Actor Failures in Wireless Sensor and Actor Networks," *IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE Publ., Piscataway, NJ, 2008, pp. 2480–2485. doi:10.1109/WCNC.2008.436

[17] Abbasi, A. A., Younis, M., and Baroudi, U., "Recovering from a Node Failure in Wireless Sensor-Actor Networks with Minimal Topology Changes," *IEEE Transactions on Vehicular Technology*, Vol. 62, No. 1, 2013, pp. 256–271. doi:10.1109/TVT.2012.2212734

[18] Aksaray, D., and Mavris, D., "Maintaining Connectivity for Networked Mobile Systems in the Presence of Agent Loss," *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2013-4886, 2013. doi:10.2514/6.2013-4886

[19] Corke, P., Hrabar, S., Peterson, R., Rus, D., Saripalli, S., and Sukhatme, G., "Deployment and Connectivity Repair of a Sensor Net with a Flying Robot," *Experimental Robotics IX*, Vol. 21, Springer Tracts in Advanced Robotics, Springer, Berlin, 2006, pp. 333–343. doi:10.1007/11552246_32

[20] Ji, M., and Egerstedt, M., "Distributed Coordination Control of Multiagent Systems While Preserving Connectedness," *IEEE Transactions on Robotics*, Vol. 23, No. 4, 2007, pp. 693–703. doi:10.1109/TRO.2007.900638

[21] Zavlanos, M. M., Jadbabaie, A., and Pappas, G. J., "Flocking While Preserving Network Connectivity," *46th IEEE Conference on Decision and Control*, IEEE Publ., Piscataway, NJ, 2007, pp. 2919–2924. doi:10.1109/CDC.2007.4434530

[22] Sabattini, L., Chopra, N., and Secchi, C., "On Decentralized Connectivity Maintenance for Mobile Robotic Systems," *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, IEEE Publ., Piscataway, NJ, 2011, pp. 988–993. doi:10.1109/CDC.2011.6161067

[23] Sabattini, L., Secchi, C., and Chopra, N., "Decentralized Connectivity Maintenance for Networked Lagrangian Dynamical Systems," *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE Publ., Piscataway, NJ, 2012, pp. 2433–2438. doi:10.1109/ICRA.2012.6224857

[24] Bauso, D., Giarre, L., and Pesenti, R., "Quantized Dissensus in Networks of Agents Subject to Death and Duplication," *IEEE Transactions on Automatic Control*, Vol. 57, No. 3, 2012, pp. 783–788. doi:10.1109/TAC.2011.2167810

[25] Abbasi, A. A., Younis, M., and Akkaya, K., "Movement-Assisted Connectivity Restoration in Wireless Sensor and Actor Networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 20, No. 11, 2009, pp. 1366–1379. doi:10.1109/TPDS.2008.246

[26] Mi, Z., Yang, Y., and Liu, G., "HERO: A Hybrid Connectivity Restoration Framework for Mobile Multi-Agent Networks," *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE Publ., Piscataway, NJ, 2011, pp. 1702–1707. doi:10.1109/ICRA.2011.5979682

[27] Zamanifar, A., Kashefi, O., and Sharifi, M., "AOM: An Efficient Approach to Restore Actor-Actor Connectivity in Wireless Sensor and Actor Networks," *International Journal of Computer Networks and Communications*, Vol. 1, No. 1, 2009, pp. 61–72.

[28] Valenti, M., Bethke, B., How, J. P., de Farias, D. P., and Vian, J., "Embedding Health Management into Mission Tasking for UAV Teams," *American Control Conference (ACC)*, IEEE Publ., Piscataway, NJ, 2007, pp. 5777–5783. doi:10.1109/ACC.2007.4282719

[29] Sampigethaya, K., Poovendran, R., and Bushnell, L., "Security of Future eEnabled Aircraft Ad hoc Networks," *AIAA Aviation Technology, Integration and Operations (ATIO)*, AIAA Paper 2008-8894, 2008. doi:10.2514/6.2008-8894

[30] Savchenko, S. V., "On the Number of Noncritical Vertices in Strongly Connected Digraphs," *Mathematical Notes*, Vol. 79, No. 5, 2006, pp. 687–696. doi:10.1007/s11006-006-0078-7