Received: 6 September 2022 | Revised: 29 March 2023 | Accepted: 13 May 2023

DOI: 10.1002/asjc.3149

SPECIAL ISSUE

WILEY

Cluster assignment in multi-agent systems: Sparsity bounds and fault tolerance

Miel Sharf[1] | **Daniel Zelazo**[2]

[1]Jether Energy Research, Tel Aviv, Israel

[2]Faculty of Aerospace Engineering, Technion—Israel Institute of Technology, Haifa, Israel

Correspondence
Miel Sharf, Jether Energy Research, Tel Aviv, Israel.
Email: mielsharf@gmail.com

Funding information
Israel Science Foundation, Grant/Award Number: 2017658 and 2285/20

Abstract

We study cluster assignment in homogeneous diffusive multi-agent networks. Given the number of clusters and agents within each cluster, we design the network graph ensuring the system will converge to the prescribed cluster configuration. Using recent results linking clustering and symmetries, we show that it is possible to design an oriented graph for which the action of the automorphism group of the graph has orbits of predetermined sizes, guaranteeing the network will converge to the prescribed cluster configuration. We provide bounds on the number of edges needed to construct these graphs along with a constructive approach for their generation. We also consider the robustness of the clustering process under agent malfunction.

KEYWORDS

clustering, diffusive coupling, fault tolerance, graph theory, multi-agent networks, sparsity

1 | INTRODUCTION

One of the most important tasks in the field of multi-agent systems (MASs) is reaching agreement. Distributed protocols guaranteeing the agents reach agreement appear in many different fields, including robotics [1], sensor networks [2], and distributed computation [3]. A natural generalization is the *cluster agreement* problem, which seeks to drive agents into groups, so that agents within the same group reach an agreement. The clustering problem appears in social networks [4], neuroscience [5], and biomimetics [6]. Clustering is also vital in the operation of autonomous decentralized MASs in order to perform on-the-fly efficient task allocation. For example, a group of fire-fighting drones handling multiple active fires of varying severities must allocate which agents will attend which fires in a way that avoids deadlock situations [7]. This should happen even though all agents are identical and without a designated leader. Once the severity of each fire is estimated, and the number of agents allocated to handle each fire can be designated. The agents can be dispatched using a clustering algorithm. Clustering has been studied using different approaches, for example, network optimization [8], pinning control [9], inter-cluster nonidentical inputs [10], and exploiting the structural balance of the underlying graph [11]. We tackle the clustering problem by using *symmetries* within the MAS. Recent works on MAS apply graph symmetries to study various problems, for example, controllability and observability of MAS [12–15].

Our recent work [16] introduced the notion of the *weak automorphism group of a MAS*, combining the two concepts of the automorphism group for graphs and weak equivalence of dynamical systems. The former summarizes all symmetries for a given graph, while the latter characterizes similarities between achievable steady states of heterogeneous (dynamical) agents. Therefore, the weak automorphisms of a MAS can be understood as permu-

tations of the nodes in the underlying graph that preserve both graph symmetries and certain input–output properties of the corresponding agents. More specifically, the paper [16] focused on clustering for *diffusively coupled networks* and showed that under appropriate passivity assumptions, these diffusively coupled networks converge to a clustered steady-state solution, where two agents are in the same cluster if and only if there exists a weak automorphism mapping one to the other. Thus, the clustering of the MAS can be understood by studying the action of the weak automorphism group on the underlying interaction graph.

We focus in this paper on homogeneous networks, that is, networks where the agent dynamics are all identical, noting that the weak automorphism group is identical to the automorphism group of the underlying graph in this case. We wish to design graphs ensuring the MAS will converge to a prescribed cluster configuration, that is, specifying the number of clusters and the number of agents within each cluster. Our previous work [17] applied tools from group theory to prescribe an algorithm for constructing an oriented graph such that the action of the automorphism group on the graph has orbits of prescribed sizes. It also provided implicit upper and lower bounds on the number of edges needed to construct such graph. This work extends the previous work [17] in two ways. First, we further explore the bounds on the number of edges needed to build such graphs by understanding the reason for the difference between the upper and the lower bounds (the example in Remark 1). We also study the scaling properties of the upper bound with the number of agents n and the number of clusters k (Theorem 5 and Remark 3), showing that a graph with the desired clustering configuration can be built with roughly $O(n^2/k)$ edges, and providing a construction to build it. Furthermore, we study the robustness of such graphs to agent malfunctions. We show that the existing graph synthesis method does not provide any guarantees on the behavior of the closed-loop network when agent malfunctions occur, and explain how to alter the synthesis procedure to guarantee the most extensive possible robustness of the clustering possible (Section 3.2). This results in a network topology which guarantees the agents cluster in the provided formation when no malfunctions occur, and as close as possible to the provided clustering configuration for any number of faults. The general results on the number of edges needed to construct graphs with the desired cluster configuration (Theorem 5 and Remark 3) are naturally extended to the case of these robust graphs.

The rest of paper is organized as follows. Section 2 reviews basic concepts related to network systems and group theory required to define a notion of symmetry for MAS. Section 3 presents the main results about cluster assignment, as well as a numerical study to demonstrate the theory. Finally, some concluding remarks are offered in Section 5.

1.1 | Notations

This work employs basic notions from graph theory [18]. An undirected graph $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ consists of finite sets of vertices $\mathbb{V}$ and edges $\mathbb{E} \subset \mathbb{V} \times \mathbb{V}$. We denote the edge with ends $i, j \in \mathbb{V}$ as $e = \{i, j\}$. For each edge e, we pick an arbitrary orientation and denote $e = (i, j)$ when $i \in \mathbb{V}$ is the *head* of edge and $j \in \mathbb{V}$ is its *tail*. A path is a sequence of distinct nodes $v_1, v_2, \dots, v_n$ such that $\{v_i, v_{i+1}\} \in \mathbb{E}$ for all i. A cycle is path $v_1, \dots, v_n, v_1$. A simple cycle is a cycle whose vertices are all distinct. A graph is connected if there is a path between any two vertices, and a tree if it is connected but contains no simple cycles. The incidence matrix of $\mathcal{G}$, denoted $\mathcal{E} \in \mathbb{R}^{|\mathbb{E}|\times|\mathbb{V}|}$, is defined such that for any edge $e = (i, j) \in \mathbb{E}$, $[\mathcal{E}]_{ie} = +1$, $[\mathcal{E}]_{je} = -1$, and $[\mathcal{E}]_{\ell e} = 0$ for $\ell \neq i, j$. Moreover, the greatest common divisor of two positive integers m, n is denoted by $\gcd(m, n)$, and their least common multiple is denoted by $\operatorname{lcm}(m, n)$. Note that $\operatorname{lcm}(m, n) = \frac{mn}{\gcd(m,n)}$ always holds. Two integers are relatively prime (or coprime) if there is no integer greater than one that divides them both. The cardinality of a finite set A is denoted by $|A|$.

2 | SYMMETRIES IN NETWORKED SYSTEMS

In this section, we provide background on the notion of symmetries for MASs originally proposed in [16].

2.1 | Diffusively coupled networks

This section describes the structure of the MAS studied in the papers [16] and [19]. Consider a set of agents interacting over a network $\mathcal{G} = (\mathbb{V}, \mathbb{E})$. Each node $i \in \mathbb{V}$ is assigned a dynamical system Σ_i, and the edges $e \in \mathbb{E}$ are assigned controllers Π_e, having the following form:

$$\begin{aligned} \Sigma_i &: \begin{cases} \dot{x}_i = f_i(x_i, u_i), \; y_i = h_i(x_i, u_i), \end{cases} \\ \Pi_e &: \begin{cases} \dot{\eta}_e = \phi_e(\eta_e, \zeta_e), \; \mu_e = \chi_e(\eta_e, \zeta_e) \; . \end{cases} \end{aligned} \tag{1}$$

We consider the stacked vectors $u = [u_1^T, \dots, u_{|\mathbb{V}|}^T]^T$ and similarly for y, ζ, and μ. The MAS is diffusively coupled with the controller input described by $\zeta = \mathcal{E}^T y$, and the control input is $u = -\mathcal{E}\mu$, where $\mathcal{E}$ is the incidence matrix of the graph $\mathcal{G}$. This structure is denoted by the

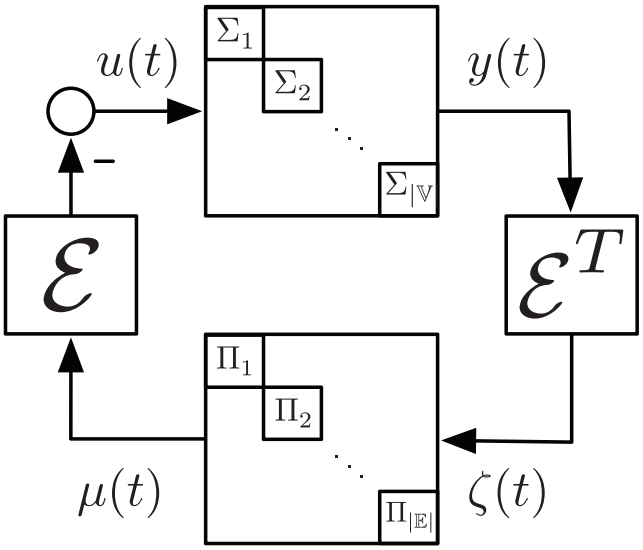


FIGURE 1 A diffusively coupled network.

triplet $(\mathcal{G}, \Sigma, \Pi)$ and is illustrated in Figure 1. In Bürger et al. [19], it was shown that the network converges to a steady state satisfying the interconnection constraints if the agents and controllers are (output-strictly) maximum equilibrium independent passive (MEIP). The details of this definition and related definitions are not essential for the development of this work, and the interested reader is referred to the papers [19] and [20] for more details. In this work, we focus on homogeneous networks, that is, where all the agent dynamics and control dynamics are identical. Moreover, we assume one of the following two alternatives (Assumption 1). If this is not the case, see previous studies [21–23] for plant augmentation techniques.

Assumption 1. The agents Σ_i are output-strictly MEIP and the controllers Π_e are MEIP, or vice versa.

A final technical definition is needed to characterize the steady states of the network. Indeed, we implicitly assume that each agent and controller converges to a steady-state output given a constant input. This allows us to define a relation between constant inputs and constant outputs called the *steady-state relation* of a system; see Bürger et al. [19]. We denote the steady-state relations of node i and edge e by k_i and γ_e, respectively. For example, for an agent i, we say that $(\mathrm{u}_i, \mathrm{y}_i)$ is a steady-state input/output pair if $\mathrm{y}_i \in k_i(\mathrm{u})$. We now introduce the notion of weak equivalence for dynamical systems.

Definition 1 (Sharf and Zelazo [16])**.** Two systems Σ_1 and Σ_j are weakly equivalent if their steady-state relations are identical.

We refer the reader to the paper [16] for a more thorough study and examples of weakly equivalent systems.

2.2 | Group theory, graph automorphisms, and symmetric MAS

Our approach for clustering will hinge on symmetry, which is modeled by the mathematical theory of groups [24]. The notion of a group can be defined in various ways, but we opt for the most concrete one.

Definition 2. Let X be a set, and let $\mathbb{G}$ be a collection of invertible functions $X \to X$. The collection $\mathbb{G}$ is called a *group* if for any $\mathbb{G} \ni f, g : X \to X$, both the composite function $f \circ g$ and the inverse function f^{-1} belong to $\mathbb{G}$.

Colloquially, the group $\mathbb{G}$ defines a collection of symmetries of the set X, and its action on X allows us to identify certain elements of X which are symmetric. Of interest in this work is the automorphism group of a (oriented) graph, which encodes structural symmetries of a graph.

Definition 3. An automorphism of a (directed or undirected) graph $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ is a permutation $\psi : \mathbb{V} \to \mathbb{V}$ such that $i \in \mathbb{V}$ is connected to $j \in \mathbb{V}$ if and only if $\psi(i)$ is connected to $\psi(j)$. We denote the automorphism group of $\mathcal{G}$ by $\mathrm{Aut}(\mathcal{G})$.

We slightly abuse notation and say that $\mathrm{Aut}(\mathcal{G})$ acts on $\mathcal{G}$ (rather than on $\mathbb{V}$).

Definition 4. Let $\mathbb{G}$ be a group of functions $X \to X$. We say $x, y \in X$ are *exchangeable* (under the action of $\mathbb{G}$) if there is some $f \in \mathbb{G}$ such that $f(x) = y$. The *orbit* of $x \in X$ is the set of elements which are exchangeable with X.

Exchangeability was considered in the paper [16] to describe the clustering behavior of a MAS. Namely, the different clusters corresponded to the different orbits of the weak automorphism group of the MAS. The following result ensures that we can consider orbits, and consequently different clusters in an MAS, as disjoint sets.

Proposition 1 (Dummit and Foote [24])**.** *Let $\mathbb{G}$ be a group of functions $X \to X$. Then, X can be written as the union of disjoint orbits. In particular, any element of X belongs to exactly one orbit.*

Finally, we combine the notions of graph automorphisms and diffusively coupled MAS comprised of weakly equivalent agents.

Definition 5 (Sharf and Zelazo [16, 17])**.** Let $(\mathcal{G}, \Sigma, \Pi)$ be a diffusively coupled MAS. A weak automorphism of a MAS is a map $\psi : \mathbb{V} \to \mathbb{V}$ with the following properties: (1) ψ is an automorphism of the graph $\mathcal{G}$ and preserves edge orientations; (2) for any $i \in \mathbb{V}$, Σ_i and $\Sigma_{\psi(i)}$ are weakly equivalent; and (3) for any $e \in$

$\mathbb{E}$, Π_e and $\Pi_{\psi(e)}$ are weakly equivalent. We denote the collection of all weak automorphisms of $(\mathcal{G}, \Sigma, \Pi)$ by $\mathrm{Aut}(\mathcal{G}, \Sigma, \Pi)$.

Naturally, the weak automorphism of a MAS is a subgroup of the group of automorphisms $\mathrm{Aut}(\mathcal{G})$ of the graph $\mathcal{G}$.

3 | CLUSTER ASSIGNMENT IN MAS

We now consider the clustering problem for MAS. Specifically, we focus on the case where the agents are homogeneous, that is, they have the exact same model, and restrict ourselves by requiring the edge controllers (1) are also homogeneous. The paper [16] established a link between the clustering behavior of a MAS and certain symmetries it has, using Definition 5. The main result from [16] is summarized below.

Theorem 1 (Sharf and Zelazo [16])**.** *Take a diffusively coupled MAS $(\mathcal{G}, \Sigma, \Pi)$ where Assumption 1 holds. Then, for any steady-state $\mathrm{y} = \begin{bmatrix} \mathrm{y}_1 & \dots & \mathrm{y}_{|\mathbb{V}|} \end{bmatrix}^T$ of the closed-loop and any weak automorphism $\psi \in Aut(\mathcal{G}, \Sigma, \Pi)$, we have $P_\psi \mathrm{y} = \mathrm{y}$, where P_ψ is the permutation matrix representation of ψ.*

This result can in fact be used to show that the network converges to a clustering configuration, where the clusters are given by the orbits of the weak automorphism group. Namely, one considers diffusively coupled MAS $(\mathcal{G}, \Sigma, \Pi)$ satisfying Assumption 1, for which the closed loop is known to converge, and the invariance properties of the steady-state limit are studied. Focusing on homogeneous networks, the paper [16] identified the value of $\gamma_e(0)$, the steady-state relation for the controller on the eth edge, as indicative of clustering. Namely, it shows that if $0 \in \gamma_e(0)$ for all $e \in \mathbb{E}$, then the MAS $(\mathcal{G}, \Sigma, \Pi)$ converges to consensus, and otherwise, it displays a clustering behavior. Namely, for homogeneous MAS, two nodes are in the same cluster if they are exchangeable under the action of $\mathrm{Aut}(\mathcal{G})$, and the converse is almost surely true.

Although the paper [16] presented a strong link between symmetry and clustering in MAS, it did not consider a synthesis procedure for solving the clustering problem:

Problem 1. Consider a collection of n homogeneous agents $\{\Sigma_i\}_{i \in \mathbb{V}}$, and let $r_1, \dots, r_k$ be positive integers which sum to n. Find a directed graph $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ and homogeneous edge controllers $\{\Pi_e\}_{e \in \mathbb{E}}$ such that the closed-loop MAS converges to a clustered steady state, with a total of k clusters of sizes $r_1, \dots, r_k$.

The goal of this section is use the tools of [16] to solve Problem 1. As described above, this can be achieved in two steps. We first make the following assumption about the controllers:

Assumption 2. The homogeneous MEIP controllers are chosen so that $0 \notin \gamma_e(0)$ for any edge $e \in \mathbb{E}$.

Second, given the desired cluster sizes $r_1, \dots, r_k$, we seek an oriented graph $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ such that the action of $\mathrm{Aut}(\mathcal{G})$ on $\mathcal{G}$ has orbits of sizes $r_1, \dots, r_k$. Moreover, we require $\mathcal{G}$ to be weakly connected[1] to assure a flow of information throughout the corresponding diffusively coupled network. If we find such a graph and Assumption 2 holds, the results of [16] guarantee that the desired clustering behavior is achieved almost surely. We make the following definition for the sake of brevity and define the corresponding problem:

Definition 6. The oriented graph $\mathcal{G}$ is said to be of type $\mathrm{OS}(r_1, \dots, r_k)$ if it is weakly connected and the action of $\mathrm{Aut}(\mathcal{G})$ on $\mathcal{G}$ has orbits of sizes $r_1, \dots, r_k$.[2]

Problem 2. Given positive integers $r_1, \dots, r_k$, determine if an oriented graph of type $\mathrm{OS}(r_1, \dots, r_k)$ exists, and if so, construct it.

Before moving on to the solving this problem, we present a tool we apply later in the proofs, called the graph quotient.

Definition 7. Let $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ be a (directed or undirected) graph, and let $V_1, V_2, \dots, V_k$ be a partition of $\mathbb{V}$ to disjoint sets. The *quotient of* $\mathcal{G}$, according to the partition $V_1, V_2, \dots, V_k$, is a graph $\mathcal{C}$ with the following properties:

(i) The nodes of $\mathcal{C}$ are denoted by $1, 2, \dots, k$.
(ii) For any $l_1, l_2 \in \{1, 2, \dots, k\}$, there is an edge $l_1 \to l_2$ in the quotient graph $\mathcal{C}$ if and only if there is at least one edge between elements of V_{l_1} and V_{l_2}.

In other words, the quotient graph is achieved by grouping the nodes of $\mathcal{G}$ by the sets $V_1, V_2, \dots, V_k$, removing edges within the same set, and identifying all edges going between the same two groups V_i and V_j. An illustration can be seen in Figure 2. It is easy to see that if $\mathcal{G}$ is connected, then so is its quotient. Indeed, if i, j are two arbitrary nodes in the quotient graphs, we take nodes $v_i \in V_i$ and $v_j \in V_j$. As $\mathcal{G}$ is connected, we can find a path from v_i to v_j in $\mathcal{G}$, which in turn defines a path in the quotient graph from i to j. This fact will play a vital role later.

[1] Recall that a directed graph is weakly connected if its unoriented counterpart is connected.
[2] OS stands for "orbit structure."

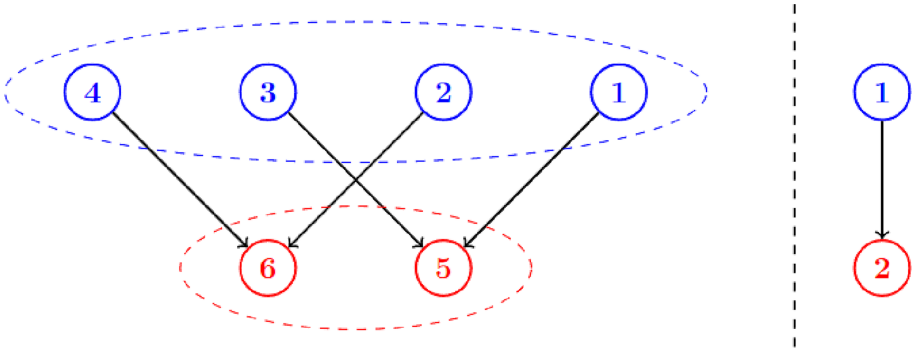


FIGURE 2 An example of graph quotient. The original graph $\mathcal{G}$ is depicted on left, and the sets V_1, V_2 of the partition are marked in blue and red, respectively. The corresponding quotient graph can be seen on the right. [Color figure can be viewed at wileyonlinelibrary.com]

3.1 | Construction and sparsity bounds on OS-type graphs

In this subsection, we exhibit a construction for OS-type graphs, as well as bounds on the sparsity of such graphs. Running the system requires means to implement the corresponding interconnections, ergo graphs with fewer edges are desirable. The following theorem provides a lower bound on the number of edges required to construct OS-type graphs.

Theorem 2. *Let $r_1, \dots r_k$ be any positive integers. Any directed graph of type OS $(r_1, \dots, r_k)$ has at least m edges, where*

$$m = \min_{\mathcal{T}\ \text{tree on}\ k\ \text{vertices}} \sum_{\{i,j\}\in\mathcal{T}} lcm(r_i, r_j). \tag{2}$$

Proof. Let $\mathcal{G}$ be a graph of type OS$(r_1, \dots, r_k)$, and $V_1, \dots, V_k$ be the orbits of Aut$(\mathcal{G})$ in $\mathcal{G}$, corresponding to the different clusters. The proof will consist of two steps. First, we show that if there exists at least one edge between V_i and V_j, then there are at least lcm(r_i, r_j) edges between V_i and V_j. This will follow from the fact that the automorphism group of $\mathcal{G}$ can map any two nodes in each V_i to one another. Second, we consider the quotient graph by the partition $V_1, \dots, V_k$, which must be connected as $\mathcal{G}$ is connected. Hence, it must have a spanning tree, which can be used to determine which pairs $i, j \in \{1, \dots, k\}$ are connected by an edge in the quotient graph. These two facts together will allow us to establish the lower bound (2).

We start with the former claim. Let $i, j \in \{1, \dots, k\}$ such that there is an edge between a node in V_i and a node in V_j. Let $\mathcal{G}_{ij}$ be the following subgraph of $\mathcal{G}$—it consists of nodes in $V_i \cup V_j$ and of edges touching one node of V_i and one node of V_j. We recall that V_i and V_j are both invariant under any automorphism of $\mathcal{G}$, meaning that the restriction of any automorphism $\psi \in$ Aut$(\mathcal{G})$ to $\mathcal{G}_{ij}$ is an automorphism of $\mathcal{G}_{ij}$. In particular, by assumption, any two nodes in V_i can be mapped to one another using automorpshims, and the same goes for V_j. Now, recall that automorphisms preserve the degree of nodes, that is, that if ψ is an automorphism of some graph and x is a node in that graph, then x and $\psi(x)$ have the same degree. For the graph $\mathcal{G}_{ij}$, which only contains nodes between V_i and V_j, the degree of a node in V_i is equal to the number of nodes in V_j it is linked to. Thus, as all the nodes in V_i can be mapped to one another using automorphisms, we conclude they are all linked to the same number of nodes in V_j. We denote this number of nodes by d_i. A similar argument with opposite roles for i and j shows that each node in V_j is linked to the same number of nodes in V_i, and this number is denoted as d_j. Recall now that each edge of $\mathcal{G}_{ij}$ touches one node from V_i and one node from V_j. In particular, the number of edges in $\mathcal{G}_{ij}$ can be found by summing over the degrees of nodes in V_i. As there are r_i nodes in V_i, and each has a degree of d_i, we conclude that $\mathcal{G}_{ij}$ has a total of $r_i d_i$ edges. However, if we sum on the nodes in V_j instead, we similarly find that $\mathcal{G}_{ij}$ has $r_j d_j$ edges. In particular, the equality $r_i d_i = r_j d_j$ holds, where both sides are positive integers.

The rest of the proof of this part of the claim follows from basic number theory. We divide both sides of the equation $r_i d_i = r_j d_j$ by the greatest common divisor $\gcd(r_i, r_j)$ and get the equation $\frac{r_i}{\gcd(r_i,r_j)} d_i = \frac{r_j}{\gcd(r_i,r_j)} d_j$, where all four numbers $\frac{r_i}{\gcd(r_i,r_j)}$, $\frac{r_j}{\gcd(r_i,r_j)}$, d_i, and d_j are positive integers. As $\frac{r_i}{\gcd(r_i,r_j)}$ and $\frac{r_j}{\gcd(r_i,r_j)}$ are coprime by the definition of the greatest common divisor, we conclude that d_i must be divided by $\frac{r_j}{\gcd(r_i,r_j)}$. In particular, $d_i \geq \frac{r_j}{\gcd(r_i,r_j)}$, and in turn, the graph $\mathcal{G}_{ij}$ has at least $r_i d_i \geq \frac{r_i r_j}{\gcd(r_i,r_j)} = \text{lcm}(r_i, r_j)$ edges, as claimed

We now move to the second part of the proof. We consider the quotient C of $\mathcal{G}$ by the partition $V_1, \dots, V_k$. As $\mathcal{G}$ is (weakly) connected, so is the quotient C. In particular, there exists a spanning tree $\mathcal{T}$ for C. By the definition of the quotient, for any two nodes i, j connected in $\mathcal{T}$ (and hence in C), there is at least one edge between elements of V_i and V_j in $\mathcal{G}$. However, by the

first part of the proof, we conclude that there are at least $\text{lcm}(r_i, r_j)$ edges between them. By summing over all connected pairs of nodes in $\mathcal{T}$, we conclude that the graph $\mathcal{G}$ has at least $\sum_{\{i,j\}\in\mathcal{T}} \text{lcm}(r_i, r_j) \geq m$ edges. This concludes the proof. □

Beside giving a lower bound on the number of edges in an OS-type graph, Theorem 2 also highlights the role of the quotient base graph $\mathcal{T}$ in the construction of such graphs. Namely, $\mathcal{T}$, which was taken as a tree, determines which clusters are connected in $\mathcal{G}$. The following algorithm, Algorithm 1, uses this idea to construct OS-type graphs when $\mathcal{T}$ is taken as a path graph.

Algorithm 1 Building sparse OS-type graphs

Input: A collection $r_1, \dots, r_k$ of positive integers summing to n, and a path $\mathcal{T}$ on k vertices.

Output: A graph $\mathcal{G}$ of type $\text{OS}(r_1, \dots, r_k)$.

1: Let $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ be an empty graph.
2: For any $j = 1, \dots, k$ and $p = 1, \dots, r_j$, add a node with label v_p^j to $\mathbb{V}$.
3: For any edge $\{i, j\}$ in $\mathcal{T}$ and any $p = 1, \dots, \text{lcm}(r_i, r_j)$, add the edge $v^i_{p \bmod r_i} \to v^j_{p \bmod r_j}$ to $\mathbb{E}$
4: Compute $i^\star = \arg\min\{r_i\}$. If $r_{i^\star} = 1$, go to Step 6.
5: For any $p = 1, \dots, r_{i^\star}$, add the edge $v_p^{i^\star} \to v^{i^\star}_{(p+1) \bmod r_{i^\star}}$ to $\mathbb{E}$.
6: **Return** $\mathcal{G} = (\mathbb{V}, \mathbb{E})$.

Algorithm 1 essentially tries to reverse the quotient process described in the proof of Theorem 2. It starts with the quotient graph, given by the tree $\mathcal{T}$, and it constructs the original graph $\mathcal{G}$ be assigning nodes to each element of the partition. More precisely, the algorithm assigns to each node j in the tree a set of r_j nodes in $\mathcal{G}$, denoted by $V_j = \{v_p^j\}_{p=1}^{r_j}$. In Step 3, the algorithm populates the graph $\mathcal{G}$ with edges corresponding to those found in the quotient $\mathcal{T}$, and it does so with the minimal possible amount of edges guaranteeing symmetry (as seen in the proof of Theorem 2). An illustration of this step can be seen in Figure 3. In Step 5, the algorithm adds a few more edges to guarantee the constructed graph $\mathcal{G}$ is connected, but in a way that does not effect the quotient process, as all new edges are between nodes in the same set $V_{i^\star}$ of the partition. As the following theorem shows, choosing any path $\mathcal{T}$ will result in a graph of type $\text{OS}(r_1, \dots, r_k)$. However, the number of edges in the graph depends on the path $\mathcal{T}$. We note that $\mathcal{T}$ must be a path rather than a general tree, see Remark 1 for further discussion.

Theorem 3. *Let $r_1, \dots r_k$ be positive integers summing to n. For any path $\mathcal{T}$, Algorithm 1 outputs a graph $\mathcal{G}$ of type $OS(r_1, \dots, r_k)$ having $\sum_{\{i,j\}\in\mathcal{T}} lcm(r_i, r_j) + \min_i r_i$ edges. Thus, there is a graph of type $OS(r_1, \dots, r_k)$ having M edges, where*

$$M = \min_{\mathcal{T}\ path\ on\ k\ vertices} \sum_{\{i,j\}\in\mathcal{T}} lcm(r_i, r_j) + \min_i r_i. \quad (3)$$

Proof. We assume, without loss of generality and for the benefit of neater notation, that the path $\mathcal{T}$ is of the form $1 \to 2 \to \dots \to k$, and let $i^\star$ be the vertex chosen in Step 5, that is, the node at which r_i is minimized. Step 3 adds a total of $\sum_{\{i,j\}\in\mathcal{T}} \text{lcm}(r_i, r_j)$ edges to the graph, and Step 5 adds $r_{i^\star}$ more edges to the graph if $r_{i^\star} \geq 2$ (and no edges if $r_{i^\star} = 1$). In particular, the number of edges in $\mathcal{G}$ is equal to $\sum_{\{i,j\}\in\mathcal{T}} \text{lcm}(r_i, r_j)$, plus $r_{i^\star}$ if $r_{i^\star} \geq 2$. Thus, it suffices to show that $\mathcal{G}$ is a graph of type $\text{OS}(r_1, \dots, r_k)$, that is, that the orbits of the action of $\text{Aut}(\mathcal{G})$ on $\mathcal{G}$ are given by $V_1, \dots, V_k$, where $V_j = \{v_j^p\}_{p=1}^{r_j}$, and that $\mathcal{G}$ is weakly connected. We start with the former.

Regarding the orbits, we must show that all nodes in V_j are exchangeable and that V_j are invariant under $\text{Aut}(\mathcal{G})$, for any $j = 1, \dots, k$. For the first claim, we consider the map $\psi : \mathbb{V} \to \mathbb{V}$ defined by sending each vertex v_p^j to $v^j_{(p+1) \bmod r_j}$, illustrated by the dashed green edges in Figure 3. One can check that this map is an automorphism of $\mathcal{G}$. Moreover, if we repeatedly apply ψ on v_p^j, we can get any vertex in V_j; hence, any two nodes in V_j are exchangeable.

Second, we show that each V_j is invariant under $\text{Aut}(\mathcal{G})$, which would imply that the orbits of $\text{Aut}(\mathcal{G})$ in $\mathcal{G}$ are exactly $V_1, \dots V_k$. This is obvious if $k = 1$, as $V_1 = \mathbb{V}$. If $k \geq 2$, we first consider the case for which $i^\star \neq k$. Graph automorphisms preserve all graph properties, including the out-degree of vertices. As all edges are oriented from V_j to V_{j+1} or from $V_i^\star$ to itself, the out-degree of all nodes in V_k is zero, and the out-degree of all other nodes in the graph is at least 1. In particular, for any automorphism of $\mathcal{G}$, any node of V_k must be mapped to a node with out-degree zero, meaning its image must lie in V_k. Therefore, we conclude that V_k is invariant under $\text{Aut}(\mathcal{G})$. Similarly, the set of all vertices in $\mathcal{G}$ that have an edge toward V_k is exactly V_{k-1}, so a similar argument shows that V_{k-1} must also be $\text{Aut}(\mathcal{G})$-invariant. Repeating this argument shows that $V_1, \dots, V_k$ must all be $\text{Aut}(\mathcal{G})$-invariant, as desired. If instead we have $i^\star = k$, then we must have $i^\star \neq 1$ as $k \geq 2$. One can now reach the same result using a similar argument, starts from V_1 and moving forward by looking at the in-degree (instead of starting from V_k and moving backwards using the out-degree).

Now, we show that $\mathcal{G}$ is weakly connected. First, note the induced subgraph on $V_{i^\star}$ is weakly connected (by Step 5). Indeed, this is clear if $r_{i^\star} = 1$, and in the case

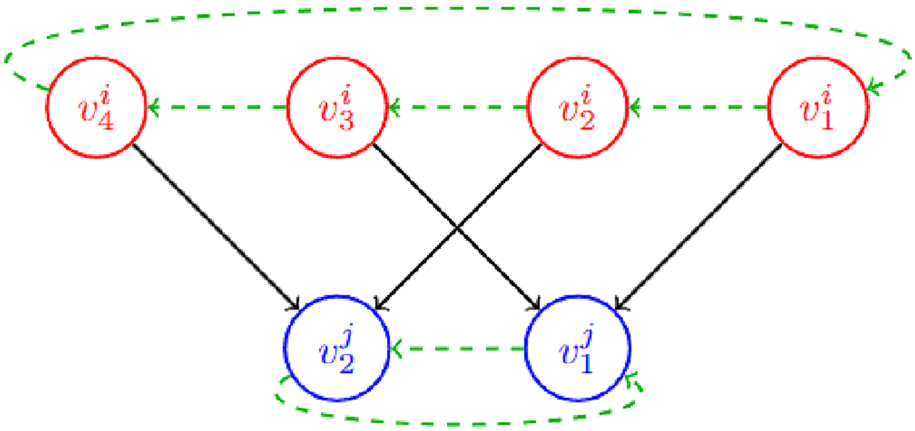


FIGURE 3 An illustration of Step 3 in Algorithm 1 with $r_i = 4$ (in red) and $r_j = 2$ (in blue). The algorithm starts with the edge between v_1^i and v_1^j. It then moves along the green dashed lines, adding an edge after each one step. This step of the algorithm terminates when the algorithm tries to add an already existing edge, resulting in the black edges depicted in the figure. [Color figure can be viewed at wileyonlinelibrary.com]

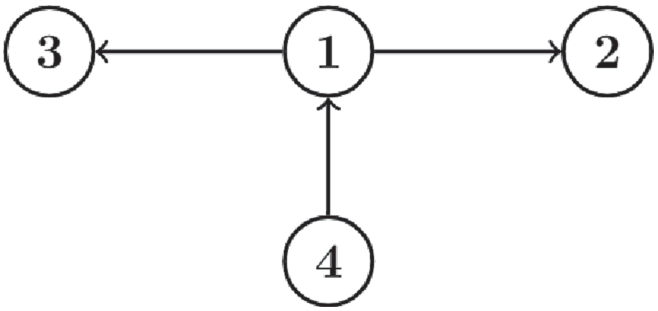


FIGURE 4 The tree graph discussed in Remark 1.

$r_{i^\star} \geq 2$, the cycle $v_1^{i^\star} \to v_2^{i^\star} \to v_3^{i^\star} \to \cdots$ eventually passes through all the nodes in $V_{i^\star}$. However, by using the edges added to the graph in Step 3 of the algorithm, we can build a path between any two nodes v_p^i and v_p^j. Namely, this is the path $v_p^i \to v_{p \bmod r_{i+1}}^{i+1} \to \cdots \to v_{p \bmod r_{j-1}}^{j-1} \to v_p^j$. Thus, any two arbitrary vertices $v_{p_1}^{j_1}$ and $v_{p_2}^{j_2}$ can be linked—first, go from $v_{p_1}^{j_1}$ to $v_{p_1 \bmod r_i}^i$ using the edges added in Step 3; then, move to $v_{p_2 \bmod r_i}^i$ using the edges added in Step 5; lastly, continue from $v_{p_2 \bmod r_i}^i$ to $v_{p_2}^{j_2}$ using the edges added to the graph in Step 3. Hence, $\mathcal{G}$ is weakly connected, and the proof is complete. □

Remark 1. The lower bound considers all possible trees, but the upper bounds only considers path graphs. It can be seen in the proof of Theorem 3 that the fact that $\mathcal{T}$ is a path is only used to prove that each V_j is invariant under the action of $\mathrm{Aut}(\mathcal{G})$. This might be false if $\mathcal{T}$ is any tree, as the following example shows. Consider Algorithm 1 with four clusters of size 1 and a tree $\mathcal{T}$ as depicted in Figure 4. In this case, the graph $\mathcal{G}$ is equal to the tree $\mathcal{T}$. However, the permutation switching the Nodes 2 and 3 is a graph automorphism, so there is a cluster of size at least 2; hence, $\mathcal{G}$ is not $\mathrm{OS}(1,1,1,1)$. Nevertheless, one should notice that the upper and lower bounds coincide when there are at most three clusters, as any tree on at most three nodes must be a path.

Remark 2. The lower bound (2) can be found using Kruskal's algorithm, which runs in $O(k^2 \log(k))$ time in our case [25]. Contrarily, the upper bound (3) requires one to solve a variant of the traveling salesman problem, which is **NP**-hard.

Algorithm 1 solves the general cluster assignment problem, as it constructs graphs of type $\mathrm{OS}(r_1, \ldots, r_k)$ for any cluster sizes $r_1, \ldots, r_k$. Unfortunately, the bound (3) is implicit in terms of the number of nodes and clusters. We elucidate it by applying it to more specific cases, resulting in concrete bounds on the number of edges needed for clustering in these cases.

Corollary 1. *Suppose all cluster sizes $r_1, \ldots, r_k > 1$ are equal. Then, there exists a graph of type $\mathrm{OS}(r_1, \ldots, r_k)$ with at most $n = r_1 + \cdots + r_k$ edges, and this number of edges is minimal.*

Proof. Let r be the size of all clusters, so that $\mathrm{lcm}(r, r) = r$ and the number of clusters is $k = n/r$. Thus, the graph $\mathcal{G}$ outputted by Algorithm 1 (for an arbitrary $\mathcal{T}$) has exactly n edges, as the summation over the edges has $k - 1$ elements. It remains to show that no graph of type $\mathrm{OS}(r_1, \ldots, r_k)$ can have fewer than n edges. As any graph with fewer than $n - 1$ edges is not weakly connected [18], it suffices to prove that such a graph cannot have exactly $n - 1$ edges.

First, note that the out-degree is preserved by the action of $\mathrm{Aut}(\mathcal{G})$, meaning that vertices in the

same cluster have the same out-degree. Denoting the out-degree of nodes in the ith cluster by d_i, the total number of edges is equal to the sum of the out-degree over all nodes, that is, to $r(d_1 + \cdots + d_k)$. In particular, the number of edges, $n - 1$ is divisible by r. As $n = kr$, n is also divisible by r, which together implies that r divides 1, which is absurd. Thus, no such graph on $n - 1$ edges can exist. □

Corollary 2. *Let $r_1, \dots, r_k$ be positive integers such that $k \geq 2$ and that for every j, l, either r_j divides r_l or vice versa. Then, there exists a graph of type OS($r_1, \dots, r_k$) with $n = r_1 + \cdots + r_k$ edges.*

Proof. We reorder the numbers $r_1, \cdots, r_k$ so that r_l divides r_j for $l \leq j$. We note that if r_l divides r_j, then $\operatorname{lcm}(r_l, r_j) = r_l$. Thus, running Algorithm 1 with $\mathcal{T} = 1 \to 2 \to \dots \to k$ gives a graph type OS($r_1, \dots, r_k$) with the following number of edges:

$$\sum_{j=1}^{k-1} \operatorname{lcm}(r_j, r_{j+1}) + r_1 = \sum_{j=1}^{k-1} r_{j+1} + r_1 = \sum_{j=1}^{k} r_j = n.$$

□

Corollary 3. *Let $r_1, \dots, r_k$ be positive integers such that $r_j \leq q$ for all j, and let $n = r_1 + \cdots + r_k$. Then, there exists a graph of type OS($r_1, \dots, r_k$) with at most $n + O(q^3)$ edges.*

Proof. We assume without loss of generality that $r_1 \leq r_2 \leq \cdots \leq r_k$. Let m_l be the number of clusters of size l for $l = 1, 2, \dots, q$, and let $\mathcal{G}$ be the graph constructed by Algorithm 1 for the path $\mathcal{T} = 1 \to 2 \to \cdots \to k$. If $r_j = r_{j+1}$, then $\operatorname{lcm}(r_j, r_{j+1}) = r_j$, and $\operatorname{lcm}(r_j, r_{j+1}) \leq r_j r_{j+1}$ otherwise. Thus, the number of edges in $\mathcal{G}$ is given by

$$\sum_{j=1}^{k-1} \operatorname{lcm}(r_j, r_{j+1}) + r_1 \leq \sum_{\substack{l \in \{1, \dots, q\}, \\ m_l \neq 0}} (m_l - 1)l + \sum_{l=1}^{q-1} l(l-1) + r_1.$$

Indeed, for each $l \in \{1, \dots, q\}$, if there's at least one cluster of size l, then there are $m_l - 1$ edges in the path $\mathcal{T}$ that touch two clusters of size l. The second term bounds the number of edges between clusters of different sizes. We note that $n = \sum_{l=1}^{q} l m_l$, so the first term is bounded by n. As for the second term, we write $l(l-1) \leq l^2$ and use the formula $\sum_{l=1}^{q-1} l^2 = \frac{(q-1)q(2q-1)}{6}$. Lastly, the last term r_1 is bounded by q. This completes the proof. □

Theorem 5 will show that the upper bound (3) is bounded by $\frac{(k-1)n^2}{k^2}$ for any cluster sizes, and it will also give a heuristic for choosing the path $\mathcal{T}$ for Algorithm 1 guaranteeing the number of edges does not exceed this upper bound.

3.2 | Robust OS-type graphs

The previous section presented a solution to the problem of cluster assignment. Namely, given a collection of homogeneous agents and the desired cluster sizes, the designer constructs the interconnection graph using Algorithm 1, and then chooses a controller following Assumption 2. The analysis depicted in the paper [16] shows that the closed-loop network would then converge to the desired clustered steady state. However, there are no guarantees on what happens if the network changes abruptly, either due to hardware or software malfunction, a cyber attack, or both. In these cases, one (or more) of the agents effectively become disconnected from the rest of the network and are effectively removed from the dynamical system and the interaction graph. For this reason, we wish to explore the robustness of OS-type graphs to changes. Ideally, when one (or more) agent is removed from the system due to a malfunction, all other agents should still cluster as before. More specifically, two non-compromised agents that were previously in the same cluster, should still belong to the same cluster. We thus make the following definition:

Definition 8. The oriented graph $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ with n nodes is said to be s-robustly OS($r_1, \dots, r_k$) for a positive integer s (called the clustering robustness parameter) if the following conditions hold: (i) $\mathcal{G}$ is weakly connected; (ii) the orbits $V_1, \dots, V_k$ of the action of $\operatorname{Aut}(\mathcal{G})$ on $\mathcal{G}$ have sizes $r_1, \dots, r_k$, respectively; and (iii) for any set $A \subseteq \mathbb{V}$ such that $|A| \geq n - s$ (comprised of the non-compromised agents), denoting the induced subgraph with node set A as $\tilde{\mathcal{G}}$, the action of $\operatorname{Aut}(\tilde{\mathcal{G}})$ on $\tilde{\mathcal{G}}$ has orbits $V_1 \cap A, \dots, V_k \cap A$. Moreover, we say that the oriented graph $\mathcal{G}$ is *totally robustly OS*($r_1, \dots, r_k$) if it is s-robustly OS($r_1, \dots, r_k$) for $s = n$.

Proposition 2. *Let $r_1, \dots, r_k$ be positive integers, and let $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ be a 1-robustly OS($r_1, \dots, r_k$) with clusters $V_1, \dots, V_k$. Then, for any $i \neq j \in \{1, \dots, k\}$, either there are no edges between V_i and V_j or that any node from V_i is linked with any node from V_j.*

Proof. Let $i, j \in \{1, \dots, k\}$, and suppose there is an edge between two nodes $v_i^\star \in V_i$ and $v_j^\star \in V_j$. We first show that any node in $v_j \in V_j$ are connected to $v_i^\star$. Indeed, consider the graph $\tilde{\mathcal{G}} \backslash \{v_i^\star\}$ corresponding to the case in which $v_i^\star$ malfunctions. As $\mathcal{G}$ is 1-robustly OS($r_1, \dots, r_k$) with clusters $V_1, \dots, V_k$, we conclude that v_j and $v_j^\star$ are in the same cluster for both $\mathcal{G}$ and $\tilde{\mathcal{G}}$. Thus, there exist automorphisms $\psi \in \operatorname{Aut}(\mathcal{G})$ and

$\tilde{\psi} \in \mathrm{Aut}(\tilde{\mathcal{G}})$ such that $\psi(v_j) = v_j^\star$ and $\tilde{\psi}(v_j) = v_j^\star$. As automorphisms preserve the degree of nodes, we conclude that v_j and $v_j^\star$ have the same degree both in $\mathcal{G}$ and in $\tilde{\mathcal{G}}$. However, the degree of $v_j^\star$ in $\tilde{\mathcal{G}}$ is smaller than its degree in $\mathcal{G}$ by one, as we removed the edge between $v_i^\star$ and $v_j^\star$ when we constructed $\tilde{\mathcal{G}}$. This means that the degree of v_j in $\tilde{\mathcal{G}}$ is smaller than its degree in $\mathcal{G}$ by one, which implies that v_j is connected to $v_i^\star$ in $\mathcal{G}$, as $\tilde{\mathcal{G}} = \mathcal{G}\backslash\{v_i^\star\}$. If we repeat this argument while replacing $v_j^\star$ by v_j and swapping i and j, we conclude that any node in V_i is connected to v_j. As v_j was an arbitrary node in V_j, we conclude that any node in V_i is connected to any node in V_j, as claimed. □

Proposition 2 suggests a necessary update to Algorithm 1 to get robust OS-type graphs. Indeed, the modulo-based construction of edges between different clusters is replaced by taking all possible edges. The adapted Algorithm 2 is given below.

Algorithm 2 Building totally robust OS-type graphs

Input: A collection $r_1, \dots, r_k$ of positive integers summing to n, and a path $\mathcal{T}$ on k vertices.

Output: A graph $\mathcal{G}$ of type $\mathrm{OS}(r_1, \dots, r_k)$.

1: If $k = 1$, return the complete graph on n nodes. Otherwise, continue.
2: Let $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ be an empty graph.
3: For any $j = 1, \dots, k$ and $p = 1, \dots, r_j$, add a node with label v_p^j to $\mathbb{V}$.
4: For any edge $\{i, j\}$ in $\mathcal{T}$, $p_i = 1, \dots, r_i$ and $p_j = 1 \dots, r_j$, add the edge $v_p^i \to v_q^j$ to $\mathbb{E}$.
5: **Return** $\mathcal{G} = (\mathbb{V}, \mathbb{E})$.

Theorem 4. *Let $r_1, \dots r_k$ be positive integers, and let $n = r_1 + \cdots + r_k$. For any path $\mathcal{T}$, Algorithm 2 outputs a graph $\mathcal{G}$ which is totally robustly OS$(r_1, \dots, r_k)$, having $\sum_{\{i,j\}\in\mathcal{T}} r_i r_j$ edges.*

Proof. Let us denote the graph constructed by Algorithm 2 by $\mathcal{G} = \mathcal{G}_{\mathrm{Alg},2}(r_1, r_2, \dots, r_k|\mathcal{T})$, and consider the following set of nodes $V_i = \{v_p^i\}_{p=1}^{r_i}$. We first prove that this graph is of type $\mathrm{OS}(r_1, \dots, r_k)$, that is, that $\mathcal{G}$ is weakly connected, that all nodes inside each V_i are exchangeable, and that each set V_i is invariant under $\mathrm{Aut}(\mathcal{G})$. The graph is obviously weakly connected, as any node in V_i is connected to any node in V_{i+1}. Moreover, one can show that each set V_i is invariant using a similar argument to the proof of Theorem 3. As for the claim that all nodes inside each V_i are exchangeable, we note that the nodes inside each set V_i can be permuted arbitrarily without changing the structure of the graph. More precisely, if σ_i is an arbitrary permutation on the set $\{1, 2, \dots, r_i\}$, then the map $\psi : \mathbb{V} \to \mathbb{V}$ defined by $\psi(v_p^i) = v_{\sigma_i(p)}^i$ is an automorphism of $\mathcal{G}$. In particular, any to nodes inside each V_i can be mapped to one another by an automorphism, as desired. We therefore conclude that $\mathcal{G} = \mathcal{G}_{\mathrm{Alg},2}(r_1, r_2, \dots, r_k|\mathcal{T})$ is of type of type $\mathrm{OS}(r_1, \dots, r_k)$.

Now, let A be a set of non-compromised agents, and let $c_i = |V_i\backslash A|$ be the number of compromised agents in cluster i. We observe that the induced subgraph $\tilde{\mathcal{G}}$ on A is equal to $\mathcal{G}_{\mathrm{Alg},2}(r_1 - c_1, r_2 - c_2, \dots, r_k - c_k|\mathcal{T})$, where the clusters are given by $V_1 \cap A, V_2 \cap A, \dots, V_k \cap A$. Thus, $\mathcal{G}$ is a completely robust OS-type graph. Lastly, counting the number of edges in $\mathcal{G}$ added in Step 3 of Algorithm 2, we conclude that $\mathcal{G}$ has $\sum_{\{i,j\}\in\mathcal{T}} r_i r_j$ edges, as claim. This completes the proof of the theorem. □

Regrading sparsity, Proposition 2 and Algorithm 2 show that the number of edges in s-robust (or totally robust) OS-type graphs can be bounded from above by M' and from below by m', where

$$\begin{aligned} m' &= \min_{\mathcal{T} \text{ tree on } k \text{ vertices}} \sum_{\{i,j\}\in\mathcal{T}} r_i r_j, \\ M' &= \min_{\mathcal{T} \text{ path on } k \text{ vertices}} \sum_{\{i,j\}\in\mathcal{T}} r_i r_j. \end{aligned} \tag{4}$$

Furthermore, the relation $ab = \gcd(a, b)\mathrm{lcm}(a, b)$ gives a connection between (4), (2), and (3). Namely, if $\rho = \max_{i,j} \gcd(r_i, r_j)$, then $m \le m' \le \rho m$ and $M \le M' \le \rho M$. We note that $\rho \le \max_i r_i$, meaning the number of edges required to get totally robust OS-type graphs isn't much larger than the number of edges required to get OS-type graph, at least when there are no large clusters.

Before moving on to a numerical example, we wish to provide one more result relating the upper bound M' (and hence M) to the number of nodes and the number of clusters.

Theorem 5. *Let $r_1, \dots, r_k$ be positive integers summing to n. Suppose without loss of generality that $r_1 \ge r_2 \ge \cdots \ge r_k$, and let $\mathcal{T}$ be the path $1 \to k \to 2 \to (k-1) \to \dots$ on k nodes. The graph $\mathcal{G}$ outputted by Algorithm 2 has no more than $\frac{(k-1)n^2}{k^2}$ edges.*

Proof. We first consider the case in which k is odd, that is, $k = 2\ell + 1$ for some integer ℓ. There are two types of edges in the path $\mathcal{T}$—edges of the form $i \to (k+1) - i$ for $i = 1, 2, \dots, \ell$, and edges of the form $(k+1) - i \to i + 1$ for $i = 1, 2, \dots, \ell$. Thus, by Theorem 4, the number of edges in $\mathcal{G}$ is no larger than the value of the following (continuous) optimization problem:

$$\nu = \max \left\{ \sum_{\{i,j\}\in\mathcal{T}} x_i x_j = \sum_{i=1}^{\ell} x_i x_{(k+1-i)} + \sum_{i=1}^{\ell-1} x_{i+1} x_{(k+1-i)} : \sum_{i=1}^{k} x_i = n, x_1 \geq x_2 \geq \cdots \geq x_k \right\}.$$

We let $x_1^\star, \ldots, x_k^\star$ be the optimal solution to the problem above. If we show that all $x_i^\star$ are equal to each other (and hence to n/k), this would imply that the number of edges in $\mathcal{G}$ is bounded by the value of the cost function at $x_1^\star, \ldots, x_k^\star$, which is equal to $\nu = \frac{(k-1)n^2}{k^2}$, as claimed. Suppose, for example, and heading toward contradiction, that $x_2^\star < x_3^\star$ and that $\ell \geq 3$. The derivative of the cost function F satisfies the following inequality:

$$\frac{\partial F}{\partial x_2}|_{x^\star} = x_k^\star + x_{k-1}^\star \leq x_{k-1}^\star + x_{k-2}^\star = \frac{\partial F}{\partial x_3}|_{x^\star},$$

where the inequality follows from the constraints of the optimization problem. Thus, slightly reducing $x_2^\star$ and increasing $x_3^\star$ by the same amount results in a feasible solution with at least the same value of the cost function. This is contradictory to the manner in which $x^\star$ was chosen, meaning that $x_2^\star = x_3^\star$. A similar argument shows that in fact, $x_1^\star = x_2^\star = \cdots = x_\ell^\star$, and that $x_{\ell+1}^\star = \cdots = x_k^\star$. Finally, if the value of the former is different from the value of the latter, one similarly shows that reducing all $x_1^\star, \ldots, x_\ell^\star$ by some small $\epsilon > 0$ and simultaneously increasing all $x_{\ell+1}^\star, \ldots, x_k^\star$ by the same ϵ gives a feasible solution with at least the same value of the cost function. Therefore, all the $x_i^\star$s must be equal, and the bound is achieved. The proof for an even number of clusters k is analogous and is omitted for the sake of brevity and due to lack of space. □

Remark 3. The proof of Theorem 5 also gives the worst case scenario for Algorithm 2, namely, the maximal number of edges is achieved when all clusters are equally sized. The upper bound in Theorem 5 also applies to the non-robust graphs outputted by Algorithm 1. In fact, this upper bound is tight, at least in order of magnitude. Indeed, a graph with $O\left(\frac{(k-1)n^2}{k^2}\right)$ edges can be found by taking $r_1, \ldots, r_k$ as the $(L+1), \ldots, (L+k)$th prime numbers, where the L is chosen as an integer satisfying $L \log L \approx \frac{n}{k}$.

4 | NUMERICAL EXAMPLE

We consider a team of identical drones trying to monitor multiple fires of different sizes in a forested region [7, 26,27]. As in Sujit et al. [28], we assume the number of agents required to monitor a given site increases with the severity of the fire and can be between two and four agents. The agents are assumed to be governed by the following two-input two-output simplified first-order dynamics that includes a saturation function on the velocity,

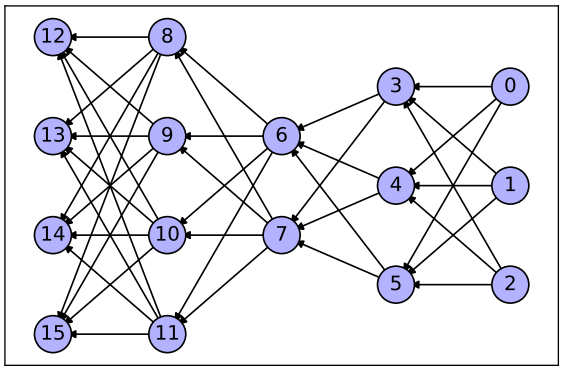
$$\dot{p}_i = -\text{sat}(0.1p_i + u_i), \;\; y_i = p_i.$$

Here, $p_i \in \mathbb{R}^2$ is the position of the agent, $u_i \in \mathbb{R}^2$ is the control input, and the saturation function defined as $\text{sat}(x) = \frac{20 \cdot x}{\max\{20, \|x\|\}}$ scales the actuation signal to guarantee that velocity never exceeds 20 m/s without affecting its direction. The forested region is of size 2 km × 2 km, and there are a total of five fires to be monitored—one mild fire requiring two drones, two medium-severity ones requiring three drones each, and two severe ones requiring four drones each. The 16 required drones begin their task from a warehouse stationed in the middle of the forest, where their initial position is a 4 × 4 grid with distances of 25 m from each other.

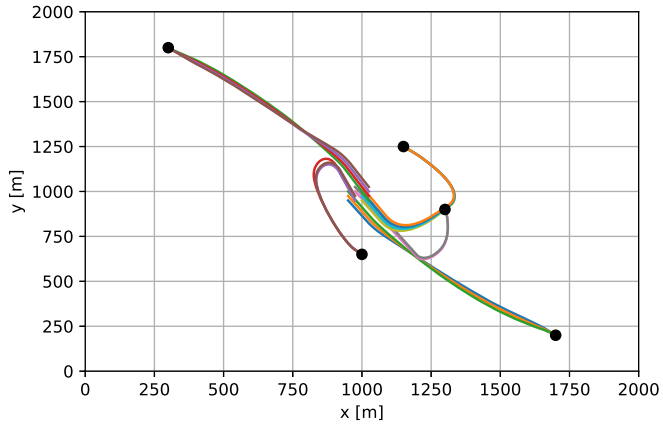
The drones use Algorithm 2 to construct a totally robustly OS-type graph $\mathcal{G}$ guaranteeing the desired clustering behavior, corresponding to one cluster of size 2, two clusters of size 3, and two clusters of size 4, shown in Figure 5a. While Theorem 4 guarantees that the desired clusters form, we now force these clusters to be at the sites of fires. This is done by considering the dependence of the steady states on the agents, controllers, and underlying graph, as expressed in [16]. Namely, if we let k be the steady-state relation of the agents, let γ be the steady-state relation of the controllers, and let y be the steady-state output of the closed-loop system, then the following relation holds:

$$0 = k^{-1}(\text{y}) + \mathcal{E}_\mathcal{G}\gamma(\mathcal{E}_\mathcal{G}^\top \text{y}), \tag{5}$$

where k^{-1} describes all constant inputs that can produce the steady-state y [19]. If we fix the steady-state output y as the locations of the fire sites and treat the controller input/output relation γ (which we choose to be identical between all edges) as the variable, then (5) turns into a linear equation connecting the values of γ at different points $\zeta^{(1)}, \ldots, \zeta^{(N)} \in \mathbb{R}^2$. For example, focusing on the first cluster gives the equation $0 = k^{-1}(\text{y}_1) - r_2\gamma(\text{y}_2 - \text{y}_1)$, where $\text{y}_i \in \mathbb{R}^2$ is the desired position of the ith cluster and $\zeta^{(1)} = \text{y}_2 - \text{y}_1 \in \mathbb{R}^2$. Solving these linear equations, we get conditions of the form $\gamma(\zeta^{(i)}) = \mu^{(i)}$ for vectors $\zeta^{(1)}, \ldots, \zeta^{(N)} \in \mathbb{R}^2$ and $\mu^{(1)}, \ldots, \mu^{(N)} \in \mathbb{R}^2$. So long that these conditions hold, and the closed-loop network is known to converge (e.g., due to passivity), the results of the papers [16, 20] guarantee that the closed-loop network will converge to the desired clustered steady state. In particular, choosing the controller as a piecewise-linear function satisfying $\gamma(\zeta^{(i)}) = \mu^{(i)}$ will work.

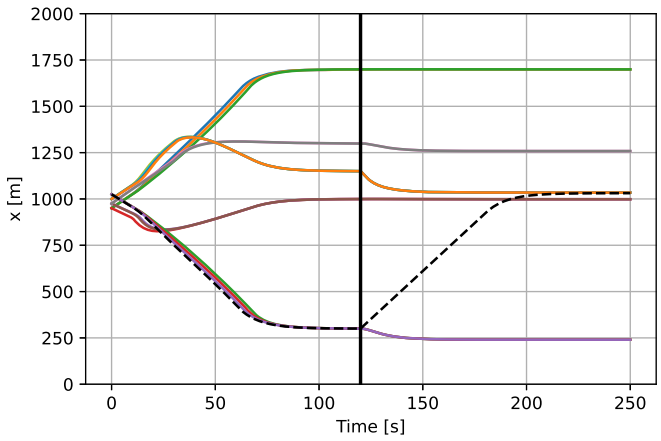


(a) Graph built by Algorithm 2 for cluster sizes $r_1 = r_2 = 3, r_3 = 2$, and $r_4 = r_5 = 4$. This is a totally robust OS-type graph, as guaranteed by Theorem 4.

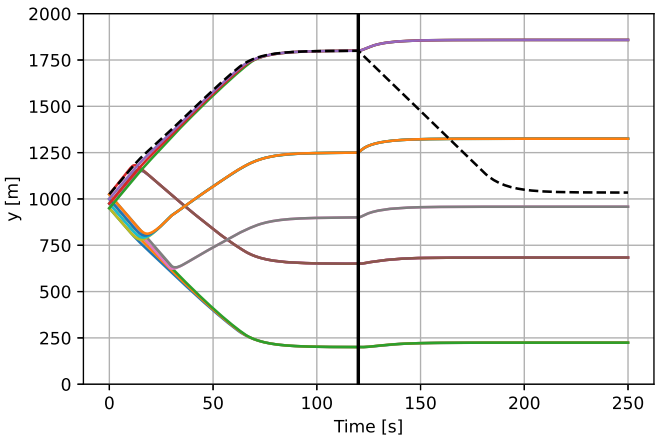


(b) Agent's trajectories in the x-y plane on the closed-loop system. The black dots represent the fire sites.

FIGURE 5 First run for forest fire numerical example. [Color figure can be viewed at wileyonlinelibrary.com]



(a) The x coordinate of the agents.



(b) The y coordinate of the agents.

FIGURE 6 Second run for forest fire numerical example. Agent #15, which was originally allocated to a large fire, malfunctions at $t = 120$ s. The moment of malfunction can be seen by the vertical black line, and the path of the agent is portrayed by the dashed black line. It can be seen that all still-functioning agents remain in their original clusters, as promised by Theorem 4. [Color figure can be viewed at wileyonlinelibrary.com]

In our case, the desired steady-state output is composed of the locations of the fire sites. The site of the mild fire is at y = (1300,900), the sites of the medium-severity fires are at (1700,200) and (1000,650), and the two severe fires are at (1150,1250) and (300,1800). Solving (5) and applying the process above yields the following piecewise-linear controller, which is applied to all edges in the graph $\mathcal{G}$:

$$\gamma\left(\begin{bmatrix}\zeta_1\\ \zeta_2\end{bmatrix}\right) = \left(\begin{bmatrix}\text{interp}([-850,-700,-150,300],[18.28,22.29,30.63,31.88])\\ \text{interp}([250,350,450,550],[-60.94,-49.06,-27.81,-19.14])\end{bmatrix}\right),$$

where interp(…) is the (1-D) linear interpolation function. To conclude, Theorem 4 guarantees we achieve the desired clustering structure by using the graph $\mathcal{G}$, and the constructed controller guarantees the clusters are at the fire sites. We run a closed-loop simulation for this system, the results of which can be seen in Figure 5b. One can see that the agents indeed converge to the desired clustering structure at the sites of the fires.

We demonstrate the robustness of clustering to malfunctioning agents. Recall that a malfunctioning agent does not communicate with any of its neighbors, and is effectively disconnected from the rest of the network. Theorem 4 implies that the constructed graph $\mathcal{G}$ is totally robust,

meaning that any number of agent malfunctions should have no effect on the clustering structure of the other, still-functioning agents. To verify this claim, we rerun the simulation, but cause one of the agents (#15) assigned to a large fire to malfunction after two minutes. Figure 6 show the x and y coordinates of each of the agents, and one can see that the clustering structure remains the same, as promised by Theorem 4, although the clusters change their location a bit. As agent #15 looses interaction with its neighbors, its control becomes 0 and the agent returns to its initial position (dashed line in Figure 6).

5 | CONCLUSIONS

This work explored the problem of cluster assignment for homogeneous diffusively coupled MASs. We relied upon the clustering analysis results of Sharf and Zelazo [16] to exhibit synthesis procedures that guarantee a clustering behavior, no matter the desired number of clusters nor their size. This was done by prescribing graph synthesis algorithms which have certain symmetry properties that reflect the desired clustering assignment. When such graphs are used in a network comprised of weakly equivalent agent and controller dynamics, the network converges to a cluster configuration. We further studied the robustness of such MAS to node malfunctions and presented a graph synthesis procedure which guarantees the clustering structure is robust to any number of agent malfunctions. The results were demonstrated in a numerical example.

AUTHOR CONTRIBUTIONS

Miel Sharf: Conceptualization; formal analysis; investigation; methodology; validation; visualization; writing—original draft; writing—review and editing. **Daniel Zelazo:** Conceptualization; investigation; project administration; resources; software; supervision; writing—review and editing.

CONFLICT OF INTEREST STATEMENT

The authors declare no potential conflict of interests.

ORCID

Daniel Zelazo https://orcid.org/0000-0002-2931-245X

REFERENCES

1. N. Chopra and M. W. Spong, *Advances in robot control: from everyday physics to human-like movements*, Springer, Berlin, Heidelberg, 2006, pp. 107–134.
2. R. Olfati-Saber, *Distributed Kalman filtering for sensor networks*, Proc. IEEE Conf. Decis. Control, IEEE, New Orleans, LA, 2007, pp. 5492–5498.
3. L. Xiao and S. Boyd, *Fast linear iterations for distributed averaging*, Syst. Control Lett. **53** (2004), no. 1, 65–78.
4. A. Lancichinetti and S. Fortunato, *Consensus clustering in complex networks*, Sci. Rep. **2** (2012), no. 1, 336.
5. A. Schnitzler and J. Gross, *Normal and pathological oscillatory communication in the brain*, Nature Rev. Neurosci. **6** (2005), 285–96.
6. K. M. Passino, *Biomimicry of bacterial foraging for distributed optimization and control*, IEEE Control Syst. Mag. **22** (2002), no. 3, 52–67.
7. C. Yuan, Y. Zhang, and Z. Liu, *A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques*, Canad. J. Forest Res. **45**, no. 7, 783–792.
8. M. Bürger, D. Zelazo, and F. Allgöwer, *Hierarchical clustering of dynamical networks using a saddle-point analysis*, IEEE Trans. Autom. Control **58** (2013), no. 1, 113–124.
9. J. Qin and C. Yu, *Cluster consensus control of generic linear multi-agent systems under directed topology with acyclic partition*, Automatica **49** (2013), no. 9, 2898–2905.
10. Y. Han, W. Lu, and T. Chen, *Cluster consensus in discrete-time networks of multiagents with inter-cluster nonidentical inputs*, IEEE Trans. Neural Netw. Learn. Syst. **24** (2013), no. 4, 566–578.
11. C. Altafini, *Consensus problems on networks with antagonistic interactions*, IEEE Trans. Autom. Control **58** (2013), no. 4, 935–946.
12. A. Chapman and M. Mesbahi, *On symmetry and controllability of multi-agent systems*, Proc. IEEE Conf. Decis. Control, IEEE, Los Angeles, CA, 2014, pp. 625–630.
13. A. Chapman and M. Mesbahi, *State controllability, output controllability and stabilizability of networks: a symmetry perspective*, Proc. IEEE Conf. Decis. Control, IEEE, Osaka, 2015, pp. 4776–4781.
14. G. Notarstefano and G. Parlangeli, *Controllability and observability of grid graphs via reduction and symmetries*, IEEE Trans. Autom. Control **58** (2013), no. 7, 1719–1731.
15. A. Rahmani and M. Mesbahi, *Pulling the strings on agreement: anchoring, controllability, and graph automorphisms*, Proc. Am. Control Conf., IEEE, New York, 2007, pp. 2738–2743.
16. M. Sharf and D. Zelazo, *Symmetry-induced clustering in multi-agent systems using network optimization and passivity*, Proc. Mediterr. Conf. Control. Autom., 2019, pp. 19–24.
17. M. Sharf and D. Zelazo, *Cluster assignment in multi-agent systems*, Asian Control Confer. **2022** (2022), 947–952.
18. C. Godsil and G. F. Royle, *Algebraic graph theory*, Graduate Texts in Mathematics, Springer, New York, 2001.
19. M. Bürger, D. Zelazo, and F. Allgöwer, *Duality and network theory in passivity-based cooperative control*, Automatica **50** (2014), no. 8, 2051–2061.
20. M. Sharf and D. Zelazo, *A network optimization approach to cooperative control synthesis*, IEEE Control Syst. Lett. **1** (2017), no. 1, 86–91.
21. A. Jain, M. Sharf, and D. Zelazo, *Regulatization and feedback passivation in cooperative control of passivity-short systems: a network optimization perspective*, IEEE Control Syst. Lett. **2** (2018), 731–736.
22. M. Sharf, A. Jain, and D. Zelazo, *A geometric method for passivation and cooperative control of equilibrium-independent passivity-short systems*, IEEE Trans. Autom. Control **66** (2021), no. 12, 5877–5892.

23. M. Sharf and D. Zelazo, *Network feedback passivation of passivity-short multi-agent systems*, IEEE Control Syst. Lett. **3** (2019), no. 3, 607–612.
24. D. S. Dummit and R. M. Foote, *Abstract algebra*, 3rd ed., Wiley, Hoboken, NJ, 2004.
25. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 3rd ed., The MIT Press, Cambridge, MA, 2009.
26. M. S. Allauddin, G. S. Kiran, G. S. S. R. Kiran, G. Srinivas, G. U. R. Mouli, and P. V. Prasad, *Development of a surveillance system for forest fire detection and monitoring using drones*, IGARSS 2019—2019 IEEE Int. Geosci. Remote Sens. Symp., IEEE, Yokohama, 2019, pp. 9361–9363.
27. D. Kinaneva, G. Hristov, J. Raychev, and P. Zahariev, *Early forest fire detection using drones and artificial intelligence*, 2019 42nd Int. Convention Inf. Commun. Technol. Electron. Microelectron. (MIPRO), IEEE, Opatija, Croatia, 2019, pp. 1060–1065.
28. P. B. Sujit, D. Kingston, and R. Beard, *Cooperative forest fire monitoring using multiple UAVs*, 2007 46th IEEE Conf. Decis. Control, IEEE, New Orleans, LA, 2007, pp. 4875–4880.

AUTHOR BIOGRAPHIES

Miel Sharf is a researcher at Jether Energy Research, Tel Aviv, Israel. He received his B.Sc. (2013) and M.Sc. (2016) degrees in Mathematics and his Ph.D. (2020) in Aerospace Engineering at the Technion—Israel Institute of Technology. From 2020 to 2022, he was a postdoctral researcher at the Division of Decision and Control Systems, KTH Royal Institute of Technology, Stockholm, Sweden. He is a recipient of the Springer Thesis Award and was selected as a part of the 2021 class of Forbes Israel "30 under 30." His research interests include electrical networks, graph theory, multi-agent systems, nonlinear control and optimization, data-driven control, and formal verification of control systems.

Daniel Zelazo received the B.Sc. and M.Eng. degrees in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1999 and 2001, respectively, and the Ph.D. degree in Aeronautics and Astronautics from the University of Washington, Seattle, WA, USA, in 2009. He is an associate professor of Aerospace Engineering and the director of the Philadelphia Flight Control Laboratory, Technion—Israel Institute of Technology, Haifa, Israel. From 2010 to 2012, he was a postdoctoral research associate and a lecturer with the Institute for Systems Theory and Automatic Control, University of Stuttgart, Stuttgart, Germany. His research interests include topics related to multi-agent systems. Dr. Zelazo is currently a subject editor of the International Journal of Robust and Nonlinear Control.

How to cite this article: M. Sharf and D. Zelazo, *Cluster assignment in multi-agent systems: Sparsity bounds and fault tolerance*, Asian J. Control (2023), 1–13, DOI 10.1002/asjc.3149.