A Decentralized Learning Strategy to Restore Connectivity during Multi-agent Formation Control

Rajdeep Dutta[a,*], Harikumar Kandath[b], Senthilnath Jayavelu[a,*], Li Xiaoli[a], Suresh Sundaram[c], Daniel Pack[d,*]

[a]*Department of Machine Intellection, Institute for Infocomm Research, A*STAR, Singapore-138632*
[b]*Robotics Research Center, IIIT Hyderabad-500032, India*
[c]*Department of Aerospace Engineering, Indian Institute of Science, Bangalore, 560012, India*
[d]*College of Engineering and Computer Science, University of Tennessee at Chattanooga, Chattanooga, TN 37403-2598, USA*

Abstract

In this paper, we propose a decentralized learning algorithm to restore communication connectivity in multi-agent formation control. The time-varying connectivity profile of a mobile multi-agent system represents the dynamic information exchange capabilities among agents. While connected to the neighbors, each mobile agent in the proposed scheme learns to raise the team connectivity. When the inter-agent communication is lost, the associated trained neural network generates appropriate control actions to restore connectivity. The proposed learning technique leverages an adaptive control formalism, wherein a neural network tries to mimic the negative gradient of a value that relies on the agent-to-neighbor distances. All agents use the conventional consensus protocol during the connected multi-agent dynamics, and under communication loss, only the lost agent executes the neural network predicted actions to come back to the fleet. Simulation results demonstrate the effectiveness of our proposed approach for single/multiple agent loss even in the presence of velocity disturbances.

Keywords: Multi-agent system, Formation control, Connectivity restoration, Decentralized learning

1. INTRODUCTION

In the field of automatic control and robotics, multi-agent system (MAS) has been one of the most active research areas with a wide range of applications including distributed control of robots, resource allocation and management, search and object retrieval, surveillance and rescue operations, and target tracking and formation [1, 2, 3]. A dynamic fleet of multiple cooperative agents works most efficiently upon reaching a consensus [1, 4]. Essentially, the information sharing capability among all team members relies on its communication connectivity. A cooperative mission mandates a positive connectivity enabling information flow among multiple agents to take appropriate actions. A strong connectivity enhances the robustness of a multi-agent network while improving its dynamic stability [4, 5]. On the contrary, loss of agent(s) deteriorates the team performance and might even lead to a mission failure. Earlier works have contributed in designing control laws for maximizing connectivity [5, 6], maintaining and improving connectivity during formation control [7], and devising potential fields to preserve and increase connectivity [8, 9]. However, a little attention has been paid on restoring it after there is a loss of inter-agent communication. This work presents a decentralized strategy to bring back the lost/disconnected agent(s) during a formation control task.

A cooperative task like formation control demands connected multi-agent configurations throughout the dynamics [4, 2, 7, 10]. A major concern is: what happens if agents get disconnected during a mission due to various reasons including faults and attacks [11, 12]? There exist ways to drive the disconnected agents to a desired formation using the global localization [13], though the same problem becomes intractable when agent movements rely on the relative

*Corresponding author
Email addresses: Rajdeep_Dutta@i2r.a-star.edu.sg (Rajdeep Dutta), harikumar.k@iiit.ac.in (Harikumar Kandath), J_Senthilnath@i2r.a-star.edu.sg (Senthilnath Jayavelu), xlli@i2r.a-star.edu.sg (Li Xiaoli), vssuresh@iisc.ac.in (Suresh Sundaram), daniel-pack@utc.edu (Daniel Pack)

localization. Previous research has established resilient connectivity during multi-agent flocking [11] and target tracking [14], without addressing agent attrition and recovery related concerns. A distributed and online framework was developed in [15] for temporal logic path planning and intermittent communication control, where agents can only communicate at their meeting locations. To solve the distributed synchronization problem of a networked system, previous research [16] leveraged stochastic modeling to capture intermittent communications among agents under random occurring update laws in their adaptive controllers. In [17], a distributed coordination strategy is proposed to establish intermittent communication among agents while monitoring an environment. In search and rescue applications, Bayesian frameworks [18, 19] proved to be effective in finding the lost target(s) by maximizing the cumulative probability of detection. Nevertheless, the Bayesian approach does not suggest any direction for the lost agent to come back to (reconnect with) other teammates. A neural network-based learning mechanism was exploited in [20] to recover lost agent(s) using global localization, although this study was limited to navigation along a circle.

Current Limitation: The existing research attempted to solve the consensus recovery problem for a multi-agent system under node (agent) failure [12, 21, 22], without paying attention to recovering the lost agent(s). In [12], the concept of connecting neighbor set was proposed to reconfigure agents for merging the sub-graphs resulting from an external attack, while keeping a balance between improving the communication robustness against external attacks and enhancing the convergence speed for achieving consensus. Previous researchers [21] designed a monotonically non-decreasing weight to reconnect the neighbors of a failed agent (placed at cut-node), which belong to different sub-graphs. To further increase connectivity and achieve fast consensus post recovery, a recent approach tried to reconnect the disconnected set of neighbors with highest degrees [22]. Nevertheless, the achieved maximum connectivity was less than the perfect case (no failure), since their research did not account for getting back the failed agent. Recently, a history following algorithm is proposed in [23] to recover disconnected agents during multi-agent dynamics, which moves memory enabled agents towards the ideal (predefined) consensus point until their communication gets restored. However, it might not be feasible to have prior knowledge of an ideal consensus in real-world nonlinear and stochastic dynamics. To execute the history following algorithm [23], agents not only need memory of an ideal consensus but also they need to communicate when there is a communication loss and who is lost; such information are difficult to pass among agents in a decentralized control framework. Clearly, there exists ample scope of research in recovering the lost agents during multi-agent formation control.

Motivation: In decentralized control of cooperative multi-agent systems, it is not easy to teach the agents since each one has only access to its neighbor information (data) as long as the entire team is connected via at least one spanning tree [24]. Severe problem arises when there is a loss of inter-agent communication. A lost agent has no neighbors and can not receive information from any other teammates. Hence, there emerges the need for a memory-based mapping technique [25, 26, 27]. Also, in adaptive control formalism [28, 29, 30], neural networks (NNs) have been utilized to approximate complex mappings involved in solving nonlinear Hamilton-Jacobi-Bellman equations. Previous studies have shown that an NN-based Adaptive Dynamic Programming facilitates in designing the optimal tracking controller to drive an unknown nonlinear dynamics towards the desired direction [31, 32]. Such approximations motivate us to call for a neural network to predict the appropriate control actions for connectivity restoration. Learning an absolute position-based mapping requires a large amount of training data, so we teach the agents a relative displacement-based mapping. A learning approach using predefined trajectories lacks generality to offer adaptive solutions. It is cumbersome to generate various reference trajectories for different initial conditions; especially in the current problem, agent movements rely on their relative locations where any sudden change causes them to deviate from the intended paths. Therefore, we prefer an approach that learns relative displacement mapping from the data collected on the fly.

No previous works, as far as the authors are aware of, addressed the problem of connectivity restoration during multi-agent formation control, which we attempt to solve here. The contributions of our work are highlighted in the following.

- The present work adopts a realistic communication model to account for time-varying information exchange among agents.
- We propose a decentralized learning technique to enable agents taking appropriate actions continually in real-time, by utilizing only neighborhood based information during the connected dynamics.
- Our proposed approach trains agents to cope with the future risk of connectivity loss along with their coordi-

nated movements according to a consensus dynamics.

- In our formulation, the lost agent predicts the learnt direction to come back to a mobile fleet, while other teammates are still engaged in pursuing a formation without looking out for the lost.

The above contributions are supported by simulations while considering single/multiple agent loss(es) along with the effect of disturbances. The remainder of the paper is organized as follows. In Section 2, we explain the related multi-agent communications along with concise graph theory basics and relevant information on state-dependent connectivity. The underlying multi-agent dynamics is discussed and the effect of inter-agent communication loss is investigated in Section 3. In Section 4, we explain our contributions elaborately and include the current problem statement, a motivating example, and the proposed decentralized learning algorithm. Section 5 presents the obtained numerical results, and finally, Section 6 concludes the current work.

2. Multi-agent Communication

2.1. Graph Representation

In this section, we explain a realistic mathematical model to analyze a dynamic multi-agent system (MAS) comprised of point-mass agents, which leverages a time-varying graph representation, $G(t)$, with a fixed number of nodes or agents and variable number of edges or communication links. Let, $\mathbf{x}_i \in \Re^{m\times 1}$ and $\mathbf{x}_j \in \Re^{m\times 1}$ denote the position state vectors of the i^{th} and the j^{th} agents, respectively; $\mathbf{r}_{ij} \in \Re^{m\times 1}$ denotes the relative displacement vector between agents $(i,\ j)$; R_u and R_l are the upper and lower bounds on the communication range of each agent, respectively.
There are total n agents with identical communication ranges and the inter-agent communication links are bidirectional, so the multi-agent graph of order n, i.e. $G_n(t)$, is an undirected time-varying graph. An agent i gets connected to a neighbor j when their separation, $r_{ij} = \|\mathbf{r}_{ij}\|$, lies within the communication range, R_u, where the neighborhood of the i^{th} agent is defined as: $\mathcal{N}_i(t) = \{(j \neq i) \in V : r_{ij} = \|\mathbf{r}_{ij}\| = \| \mathbf{x}_i - \mathbf{x}_j \| \leq R_u\}$.
The time-varying connections among agents are captured using a time-varying Laplacian matrix [7, 33], whose elements depend on the separations between the corresponding agents. A state dependent Laplacian matrix of order n is given as: $\mathbf{L}_n(\mathbf{X}(t)) = \mathbf{D}_n(\mathbf{X}(t)) - \mathbf{A}_n(\mathbf{X}(t))$, where $\mathbf{X}(t)$ denotes the vector comprised of all agent states, $\mathbf{D}_n$ is the Degree matrix, and $\mathbf{A}_n$ is the Adjacency matrix. In accordance with an exponential communication model [5, 6, 7], the elements of a weighted adjacency matrix (a_{ij}) are given as

$$a_{ij}(t) = \begin{cases} 0 & \text{if } \ i = j\ ; \\ 1 & \text{if } \ j \in \mathcal{N}_i \ \ \& \ \ r_{ij} < R_l\ ; \\ e^{-\tau(\frac{r_{ij}-R_l}{R_u-R_l})} & \text{if } \ j \in \mathcal{N}_i \ \ \& \ \ R_l \leq r_{ij} \leq R_u\ ; \\ 0 & \text{if } \ i \neq j \ \ \& \ \ r_{ij} > R_u\ , \end{cases} \tag{1}$$

where τ is a positive constant denoting the rate of decay of the communication strength over distance [7]. The connection between two agents $(i,\ j)$ is of maximum strength when their separation r_{ij} is below a lower bound on the range R_l, it is lost beyond an upper bound on the range R_u, and it varies exponentially within the range from R_l to R_u.
The Degree matrix is a diagonal matrix with elements given as: $d_i(t) = \sum_{j(\neq i)=1}^{n} a_{ij}(t)$. Consequently, the Laplacian matrix $(\mathbf{L}_n = \mathbf{D}_n - \mathbf{A}_n)$ elements can be expressed as

$$l_{ij}(t) = \begin{cases} -a_{ij}(t) & \text{for } i \neq j\ ; \\ \sum_{j(\neq i)=1}^{n} a_{ij}(t) & \text{for } i = j\ . \end{cases} \tag{2}$$

The Laplacian $(\mathbf{L}_n(t))$ for a weighted undirected graph $(G_n(t))$ is a positive semidefinite symmetric matrix. The smallest eigenvalue of the Laplacian matrix, λ_1 is zero, and the corresponding eigenvector is $\mathbf{1}$.

2.2. State-dependent Connectivity

The second smallest eigenvalue of $L(t)$, i.e. $\lambda_2(t)$, is called the *algebraic connectivity* of the graph, and the corresponding eigenvector is known as the *Fiedler vector* [24]. $\lambda_2(t)$ provides a quantitative measure of the entire network's connectivity representing the level of information sharing capability among agents at time t.
The communication connectivity measure, $\lambda_2(t)$ of a dynamic MAS, changes with the varying connections among agents. The algebraic connectivity lies in $0 < \lambda_2(t) \leq n$ as long as there exists at least one spanning tree in the multi-agent graph, indicating any two agents can communicate through direct and/or indirect links. $\lambda_2(t)$ increases or decreases as the number and/or the strength of inter-agent connections increases or decreases. Intuitively, the state functional connectivity changes in a concave manner as described below.

2.2.1. Optimal Change in Connectivity

Consider a state-dependent Laplacian matrix $\mathbf{L}_n(\mathbf{X}(t))$. $\lambda_2(\mathbf{L}_n)$ in fact [5], is a concave function of $\mathbf{L}_n$ in the space $\mathbf{1}^\perp$ and it is the infimum of a set of linear functions in $\mathbf{L}_n$ given by

$$\begin{aligned} &\lambda_2(\mathbf{L}_n)\mathbf{v}^T\mathbf{v} \leq \mathbf{v}^T\mathbf{L}_n\mathbf{v},\ \forall \mathbf{v} \in \mathbf{1}^\perp \\ &\text{or, } \lambda_2(\mathbf{L}_n) = \inf_{\mathbf{v}\in\mathbf{1}^\perp}\{\tfrac{\mathbf{v}^T\mathbf{L}_n\mathbf{v}}{\mathbf{v}^T\mathbf{v}}\}\ . \end{aligned} \tag{3}$$

The unit eigenvector of $\mathbf{L}_n$ corresponding to the eigenvalue $\lambda_2(\mathbf{L}_n)$ is denoted by $\mathbf{v}_F$, so $\mathbf{v}_F^T\mathbf{L}_n\mathbf{v}_F = \mathbf{v}_F^T\lambda_2(\mathbf{L}_n)\mathbf{v}_F = \lambda_2$ since $\|\mathbf{v}_F\| = 1$. Note that $\mathbf{v}_F$ can be computed in a decentralized way by sharing Laplacian row information among neighbors, according to the Decentralized Orthogonal Iteration Algorithm [5, 34]. After applying a perturbation, the base Laplacian $\mathbf{L}_n$ turns into $\tilde{\mathbf{L}}_n$.

$$\begin{aligned} \mathbf{v}_F^T(\tilde{\mathbf{L}}_n)\mathbf{v}_F &= \mathbf{v}_F^T\mathbf{L}_n\mathbf{v}_F + \mathbf{v}_F^T(\tilde{\mathbf{L}}_n - \mathbf{L}_n)\mathbf{v}_F \\ &= \mathbf{v}_F^T\lambda_2(\mathbf{L}_n)\mathbf{v}_F + \mathbf{v}_F^T(\tilde{\mathbf{L}}_n - \mathbf{L}_n)\mathbf{v}_F \\ &= \lambda_2(\mathbf{L}_n) + < \mathbf{v}_F\mathbf{v}_F^T, (\tilde{\mathbf{L}}_n - \mathbf{L}_n) >\ , \end{aligned} \tag{4}$$

where $<>$ denotes the inner product of two matrices. Also, (3) boils down to the inequality: $\lambda_2(\tilde{\mathbf{L}}_n)\mathbf{v}_F^T\mathbf{v}_F \leq \mathbf{v}_F^T\tilde{\mathbf{L}}_n\mathbf{v}_F$. Moving forward, (4) leads to the following relation.

$$\lambda_2(\tilde{\mathbf{L}}_n) \leq \lambda_2(\mathbf{L}_n) + < \mathbf{v}_F\mathbf{v}_F^T, (\tilde{\mathbf{L}}_n - \mathbf{L}_n) > \tag{5}$$

In (5), the outer product $\mathbf{v}_F\mathbf{v}_F^T = \mathbf{G}_s$ represents the supergradient of $\lambda_2(\mathbf{L}_n)$, which unveils the optimal direction of connectivity increase. Therefore, in order to maximize the connectivity, the Laplacian matrix update must abide by

$$\mathbf{L}_n(t+1) = \mathbf{L}_n(t) + \beta(t)\mathbf{G}_s(t)\ . \tag{6}$$

The above update (6) converges to the optimal solution if the step-size, $\beta(t) > 0$, follows the coefficient of a square-summable series [6].

2.2.2. Optimal Change in Laplacian Elements

Let us now inspect how the optimal change in the connectivity reflects in the Laplacian matrix elements. The Laplacian matrix $\mathbf{L}_n$ can be decomposed as [5]:

$$\mathbf{L}_n = \sum_{i=1,2,\ldots,n\ ;\ j>i} \mathbf{C_{ij}} p_{ij}\ , \tag{7}$$

where $C_{ij} \in \Re^{n\times n}$ is defined for each pair of agents (i, j): $\mathbf{C_{ij}}_{ij} = \mathbf{C_{ij}}_{ji} = 1$, $\mathbf{C_{ij}}_{ii} = \mathbf{C_{ij}}_{jj} = -1$ if i and j are neighbors, and 0 otherwise for rest of the entries; $p_{ij} = \mathbf{L}_{n_{ij}}$ are the elements of a vector, $\mathbf{p} \in \Re^{\frac{n(n-1)}{2}\times 1}$, comprised of all the off-diagonal entries of Laplacian $\mathbf{L}_n$.
At this point, the connectivity maximization problem can be viewed as:

$$\max_{\mathbf{X}} \lambda_2(\mathbf{L}_n(\mathbf{X})) \equiv \max_{\mathbf{p}} \lambda_2(\sum_{i=1,2,\ldots,n\ ;\ j>i} \mathbf{C_{ij}} p_{ij})\ . \tag{8}$$

Using the decomposition (7) of $\lambda_2(\mathbf{p})$ in (5), we obtain

$$\begin{aligned}\lambda_2(\tilde{\mathbf{p}}) &\leq \lambda_2(\mathbf{p}) + \sum_{i=1,2,...,n\ ;\ j>i} < \mathbf{v}_F \mathbf{v}_F^T, \mathbf{C_{ij}}(\tilde{p}_{ij} - p_{ij}) > \\ &= \lambda_2(\mathbf{p}) + \sum_{i=1,2,...,n\ ;\ j>i} (\mathbf{v}_F^T {\mathbf{C_{ij}}}^T \mathbf{v}_F)(\tilde{p}_{ij} - p_{ij}) \ . \end{aligned} \tag{9}$$

According to (9), the supergradient vector $\mathbf{g}$ for $\mathbf{p}$ is:

$$\mathbf{g}^T = [\mathbf{v}_F^T {\mathbf{C_{12}}}^T \mathbf{v}_F, ..., \mathbf{v}_F^T {\mathbf{C_{ij}}}^T \mathbf{v}_F, ..., \mathbf{v}_F^T {\mathbf{C_{n-1,n}}}^T \mathbf{v}_F] \ .$$

For each element $p_m = p_{ij}$, the relative supergradient takes shape as: $g_{ij} = \mathbf{v}_F^T {\mathbf{C_{ij}}}^T \mathbf{v}_F = -(v_{F_i} - v_{F_j})^2$. The update rule for each element of the vector $\mathbf{p}$ is deduced below.

$$p_{ij}(t+1) = p_{ij}(t) + \gamma(t) g_{ij}(t)\ ;\ g_{ij}(t) = -(v_{F_i} - v_{F_j})^2, \tag{10}$$

where $\gamma(t)$ is the coefficient of the supergradient $g_{ij}(t)$ such that $\gamma(t) g_{ij}(t) < 0$ ensures $p_{ij}(t+1) < p_{ij}(t)$ [5]. Equation (10) reveals that the optimal change in the $(i,j)^{th}$ element of $\mathbf{L}_n$ depends on the i^{th} and j^{th} entries of the corresponding Fiedler vector.
To understand how the supergradient affects the evolution of states, (10) can further be reshaped using the definition of p_{ij}:

$$\begin{aligned} a_{ij}(t+1) &= a_{ij}(t) - \gamma(t) g_{ij}(t) \ , \\ \text{or, } e^{-\tau \tilde{r}_{ij}(t+1)} &= e^{-\tau \tilde{r}_{ij}(t)} - \gamma(t) g_{ij}(t)\ ;\ 0 \leq \tilde{r}_{ij} = (\tfrac{r_{ij} - R_l}{R_u - R_l}) \leq 1 \ . \end{aligned} \tag{11}$$

3. Multi-agent Dynamics

In this section, we explain the main concepts associated with multi-agent motion and control, and investigate the influence of connectivity in a multi-agent position consensus. We consider a swarm of n agents with the dynamics of each agent being governed by

$$\dot{\mathbf{x}}_i(t) = \begin{bmatrix} \dot{x}_i(t) \\ \dot{y}_i(t) \end{bmatrix} = \begin{bmatrix} v_{x_i}(t) \\ v_{y_i}(t) \end{bmatrix} = \mathbf{u}_i(t) \ , \tag{12}$$

where $\mathbf{x}_i(t) = [x_i(t),\ y_i(t)]^T$ denotes the two-dimensional position of an agent i at time t, $\mathbf{v}_i(t) = [v_{x_i}(t),\ v_{y_i}(t)]^T$ is its velocity coordinates and $\theta_i(t) = tan^{-1}(\frac{v_{y_i}(t)}{v_{x_i}(t)})$ is the heading angle, and $\mathbf{u}_i(t)$ is the control input. Note that the i^{th} agent's position coordinates represent its state in (12), and the associated heading angle changes implicitly with the changes in states.

3.1. Dynamics of Connected MAS

The following describes a traditional formation controller that works properly for connected multi-agent configurations.
Traditional Consensus Control: Let $\tilde{\mathbf{x}}_i(t) = \mathbf{x}_i(t) - \mathbf{x}_{fi}$ represent the deviation of the current state $\mathbf{x}_i(t)$ of agent i from the desired final state $\mathbf{x}_{fi}$. In order to drive a MAS to a position consensus when inter-agent communication is present, the traditional control objective is to design a control law $\mathbf{u}_i(t)\ \forall\ i$ such that $\tilde{\mathbf{x}}_i(t) - \tilde{\mathbf{x}}_j(t) \to \mathbf{0}$ or $\mathbf{x}_i(t) - \mathbf{x}_j(t) \to \mathbf{x}_{fi} - \mathbf{x}_{fj}$ as $t \to \infty$ for $j \in \mathcal{N}_i$. The feedback control action of each agent is given by

$$\mathbf{u}_i = \dot{\mathbf{x}}_i(t) = \ -\alpha \sum_{j \in \mathcal{N}_i(t)} a_{ij}(t)(\tilde{\mathbf{x}}_i(t) - \tilde{\mathbf{x}}_j(t)) \ , \tag{13}$$

where $\mathbf{u}_i = \dot{\tilde{\mathbf{x}}}_i(t) = \dot{\mathbf{x}}_i(t) \in \Re^{m \times 1}$ is the velocity control action applied to agent i; $j \neq i$ is a neighbor of the i^{th} agent and $\mathcal{N}_i(t)$ represents its neighborhood at time t, $a_{ij}(t)$ is the time-varying weighting factor between the neighbors

(i, j), and $\alpha > 0$ is the control gain. Equation (13) governs the dynamics of the i^{th} agent based on its neighborhood information, which can be extended for all agents using a matrix algebraic form as follows

$$\mathbf{U} = \dot{\mathbf{X}}(t) = -\alpha(\mathbf{L}_n \otimes I_m)\tilde{\mathbf{X}}(t) \ , \tag{14}$$

where $\mathbf{X},\ \tilde{\mathbf{X}},\ \mathbf{U} \in \Re^{mn\times 1}$; $\tilde{\mathbf{X}}(t) = \mathbf{X}(t) - \mathbf{X}_f$, where $\mathbf{X}(t) = [\mathbf{x}_1(t), \mathbf{x}_2(t), ..., \mathbf{x}_n(t)]^T$ is the total agent state vector containing all member states at time t and $\mathbf{X}_f = [\mathbf{x}_{f1}, \mathbf{x}_{f2}, ..., \mathbf{x}_{fn}]^T$ is the total final desired state; $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n]^T$ is the total control input vector; $\otimes$ symbol stands for the Kronecker product; and $I_m \in \Re^{m\times m}$ denotes an identity matrix. The decentralized controller (14) drives the evolving relative displacement vectors, $\mathbf{x}_i(t) - \mathbf{x}_j(t)$, to the desired displacements, $\mathbf{x}_{fi} - \mathbf{x}_{fj}$, without requiring any global localization (reference) for the final formation. Note that each agent shares only its state information with the neighbor(s). The closed-loop stability analysis of a connected MAS is available in the Appendix section.

3.2. Change in Dynamics due to Communication Loss

Here, we investigate the change in the multi-agent dynamics due to an inter-agent communication loss. The effect of a mobile agent in the entire MAS dynamics can be analyzed by segregating the corresponding row and column from the whole Laplacian matrix. To explain further, the associated Laplacian matrix is decomposed to reshape (14) as

$$\begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \\ \cdot \\ \cdot \\ \dot{\mathbf{x}}_{n-1} \\ \dot{\mathbf{x}}_n \end{bmatrix} = -\alpha \left(\left[\begin{array}{c|c} \mathbf{L}_{n-1} + B_{n-1} & -\mathbf{b}_{n-1} \\ \hline -\mathbf{b}_{n-1}^T & c_n \end{array}\right] \otimes I_m \right) \begin{bmatrix} \tilde{\mathbf{x}}_1 \\ \tilde{\mathbf{x}}_2 \\ \cdot \\ \cdot \\ \tilde{\mathbf{x}}_{n-1} \\ \tilde{\mathbf{x}}_n \end{bmatrix}, \tag{15}$$

where $\mathbf{L}_{n-1} \in \Re^{n-1\times n-1}$ represents the Laplacian matrix consisting of $(n-1)$ agents, $B_{n-1} = \text{diag}(\mathbf{b}_{n-1}) \in \Re^{n-1\times n-1}$ with $\mathbf{b}_{n-1} = [a_{n1}, a_{n2}, ..., a_{n,n-1}]^T \in \Re^{n-1\times 1}$, and $c_n = \sum_{j=1}^{n-1} a_{nj}$. Note that altering the sequence of nodes in a multi-agent graph preserves its eigenvalues [33]. Hence, the nodes in a multi-agent graph can be reordered to treat an agent of concern as the end node. Suppose the lost agent is not placed at the cut-node of a multi-agent graph, and losing it does not result in two disjoint sub-graphs. In other words, the rest of the team members can still communicate among themselves even if the concerned agent gets disconnected.

All the agents move according to the governing dynamics (15) as long as the multi-agent graph remains connected, i.e. $\lambda_2(\mathbf{L}_n(t)) > 0$ for $0 < t < t_{dc}$ sec. When an agent loses connection at $t = t_{dc}$ sec, then $\lambda_2(\mathbf{L}_n(t)) = 0$ for $t \geq t_{dc}$ sec. So, the governing equation of motion takes shape as follows.

$$\begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \\ \cdot \\ \cdot \\ \dot{\mathbf{x}}_{n-1} \\ \dot{\mathbf{x}}_n \end{bmatrix} = -\alpha \left(\left[\begin{array}{c|c} \mathbf{L}_{n-1} & \mathbf{0} \\ \hline \mathbf{0}^T & 0 \end{array}\right] \otimes I_m \right) \begin{bmatrix} \tilde{\mathbf{x}}_1 \\ \tilde{\mathbf{x}}_2 \\ \cdot \\ \cdot \\ \tilde{\mathbf{x}}_{n-1} \\ \tilde{\mathbf{x}}_n \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \cdot \\ \cdot \\ \mathbf{0} \\ \mathbf{v}_c \end{bmatrix}. \tag{16}$$

In (16), $\mathbf{v}_c$ denotes the critical velocity of the n'th agent at $t = t_{dc}$ sec. Without any interventions, the disconnected agent continues moving at $\mathbf{v}_c$ and gradually diverges from the rest of the teammates.

4. Formulation and Methodology

4.1. Problem Statement and Approach

The present work aims to restore the communication connectivity during a multi-agent dynamics in case of link failure(s). In accordance with the adopted decentralized consensus controller, agents pursue a position consensus using only their relative localizations. The relative displacement states evolve with time and finally converge to the desired separation vectors. However, each agent is aware of its position with respect to a global coordinate frame. So, the challenge is: how to bring back a lost agent into the fleet by utilizing its own and last known nearest neighbor's positions?

The involved processes are: (i) During the connected phase of the dynamics (14), each agent $i \in [1,n]$ learns to increase connectivity and cope with the future risk of communication loss by training a function approximator NN_i: $\hat{\mathbf{u}}_i = g(\mathbf{x}_i)$; NN_i training objective needs information on agent i's own state $\mathbf{x}_i$ and its neighbor state(s) $\mathbf{x}_j$ for $j \in \mathcal{N}_i$. (ii) When an agent κ is disconnected, then it uses the learnt function approximation to predict the required control action $\hat{\mathbf{u}}_\kappa$ to get back to the team and restore connectivity. This prediction does not require any neighbor's state information. (iii) Once the connectivity is restored, then all the agents follow the consensus protocol (14) to reach the desired symmetric formation. A schematic of the proposed methodology is captured in Fig. 1.

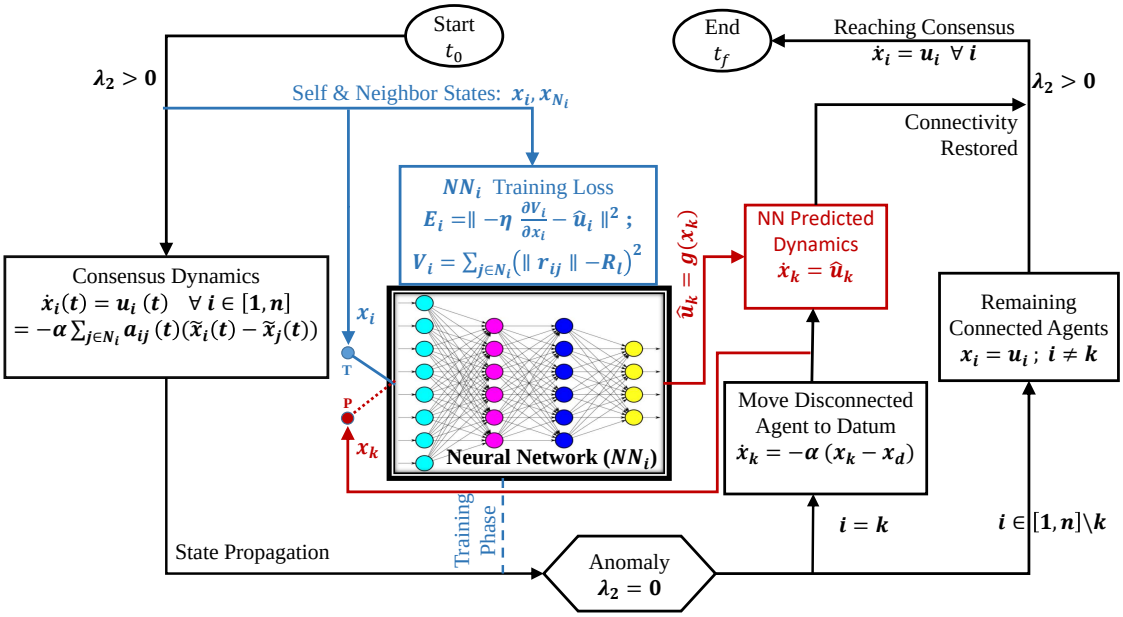


Figure 1: Schematic for restoring connectivity during multi-agent formation control. Here, the blue colored items (lines and boxes) are encountered during NN training and the red colored items are encountered during NN prediction. Note: the inputs $\mathbf{x}_i$ (in blue) and $\mathbf{x}_k$ (in red) are not fed to the NN, concurrently; $\mathbf{x}_i$ of each agent i is passed during training (T) and $\mathbf{x}_k$ of the lost agent k is passed during prediction (P).

4.2. Increasing Connectivity by Function Approximator

Recall that the connectivity, $\lambda_2(\mathbf{L}_n(\mathbf{X}(t)))$, is a state functional. The update rule in (11) states how the relative distance between $(i,\ j)$ decreases along the supergradient direction, although it involves the Fiedler vector elements that are difficult to estimate. This estimation problem can be mitigated by adopting a potential-based control strategy [5]. Without any communication loss, previous research [5] has proved that the negative gradient of a decentralized value can imitate the connectivity supergradient direction (10). However, their potential-based control can not maximize connectivity of a dynamic MAS under inter-agent communication loss, to overcome which, we call for a function approximator like Neural Network (NN) in our research.

The existing concept of potential (value) is adopted in the objective to train the currently employed NN. An underlying assumption is that the training is over prior to losing connectivity. According to the universal approximation theory [25], NN can approximate unknown dynamics in finite time. When agent(s) gets disconnected, the learnt NN predicts the control actions required to restore connectivity. For instance, thanks to the mapping approximated by a trained NN, a reasonable velocity prediction, $\hat{\mathbf{v}}_c = g(\mathbf{x}_n)$ in (16), can still be achieved when $t > t_{dc}$. At the time of reconnection $t = t_{rc} > t_{dc}$, agent n links with at least one of the rest (o), which reflects as a non-zero element l_{no} of $\mathbf{L}_n$, hence $\dot{\mathbf{x}}_n = g(\mathbf{x}_n) = \alpha\ a_{on}(\mathbf{x}_o - \mathbf{x}_n)$ satisfies at $t = t_{rc}$.

4.2.1. A Motivating Example

Here, we analyze how the relative displacements between different agents evolve during the MAS dynamics. Let k is the concerned agent that gets disconnected from its only neighbor i after t_{dc} secs. Now, consider a candidate value function: $V_k(t) = \frac{1}{2}\|\mathbf{r}_{ki}\|^2 = \frac{1}{2}(\mathbf{x}_k - \mathbf{x}_i)^T(\mathbf{x}_k - \mathbf{x}_i)$, which is always positive except where the separation between (k, j) agents is zero. The time-derivative of V_k is given by

$$\dot{V}_k = (\mathbf{x}_k - \mathbf{x}_i)^T(\dot{\mathbf{x}}_k - \dot{\mathbf{x}}_i) \ . \tag{17}$$

In current formalism, agent k learned to minimize its separation $\|\mathbf{r}_{ki}\|$ from agent i when they were connected. During a mission, the value V_k can be computed only for $t < t_{dc}$, when agent k learns the negative gradient of V_k with respect to its state, which is proportional to: $-\frac{\partial V_k}{\partial \mathbf{x}_k} = -(\mathbf{x}_k - \mathbf{x}_i)$.
After agent k gets disconnected, then it moves along the learnt direction. Therefore

$$\dot{\mathbf{x}}_k = g(\mathbf{x}_k) = -\eta(\mathbf{x}_k - \mathbf{x}_i) + \varepsilon \ ; \ \eta > 0 \ , \tag{18}$$

where $\varepsilon = 0$ is the prediction error. Note that the last known neighbor (i) of the disconnected agent (k) is dynamic though its state evolution is governed by the consensus protocol. Consequently, agent i's states do not change abruptly and the evolution pattern is captured within the trained NN. Thus, the disconnected agent is expected to move along the approximated direction (18).
Here, agent i is still connected to the rest of the team. Suppose, it has a neighbor agent j. Hence, in accordance with the consensus dynamics

$$\dot{\mathbf{x}}_i = -\alpha a_{ij}(\mathbf{x}_i - \mathbf{x}_j) \ . \tag{19}$$

By substituting (18) and (19) into (17), we obtain

$$\begin{aligned} &\dot{V}_k = (\mathbf{x}_k - \mathbf{x}_i)^T(-\eta(\mathbf{x}_k - \mathbf{x}_i) + \varepsilon + \alpha a_{ij}(\mathbf{x}_i - \mathbf{x}_j)) \\ &\text{or, } \dot{V}_k = -(\eta + \alpha a_{ij})\|\mathbf{r}_{ki}\|^2 + \alpha a_{ij}(\mathbf{x}_k - \mathbf{x}_i)^T(\mathbf{x}_k - \mathbf{x}_j) + (\mathbf{x}_k - \mathbf{x}_i)^T\varepsilon \ . \end{aligned} \tag{20}$$

The first term on the right side of (20) is always negative since $\|\mathbf{r}_{ki}\|^2 > 0$ and $(\eta + \alpha a_{ij}) > 0$, and the last term is negligible considering an accurate prediction. However, the second term on the right side of (20) is concerning towards establishing the negative definiteness of $\dot{V}_k$, which implies that the separation between agents k and i decreases. According to (20), $\dot{V}_k < 0$ if $(\mathbf{x}_k - \mathbf{x}_i)^T(\mathbf{x}_k - \mathbf{x}_j) \leq 0$, which indicates a specific geometric configuration where the angle between the vectors $(\mathbf{x}_k - \mathbf{x}_i)$ and $(\mathbf{x}_k - \mathbf{x}_j)$ lies in $90 \leq \angle(\mathbf{x}_k - \mathbf{x}_i), (\mathbf{x}_k - \mathbf{x}_j) \leq 270$ degrees. This condition is visually explained in Fig. 2.

The location of the last known neighbor of agent k is called as the datum. Recall that agents i and j are neighbors and their coordinated movements are bound to the consensus dynamics, therefore, they are likely to be on the same side with reference to a location (datum) already visited by agent i. Thus, it is possible to maintain the angle between $(\mathbf{x}_k - \mathbf{x}_i)$ and $(\mathbf{x}_k - \mathbf{x}_i)$ within a range of $90 - 270$ degrees towards ensuring $\dot{V}_k < 0$. To tackle more challenging scenarios due to multiple agent losses, we adopt a generic learning approach as explained in the following. Also, it is worth noting that the NN-predicted velocity commands are applied with respect to the datum as it is contained within the spatial domain used to train NN.

4.3. Decentralized Learning Method

This section illustrates the value-action based decentralized learning algorithm to restore connectivity during multi-agent formation control.

Consider a function approximator, NN_κ, assigned to the concerned agent κ, which takes its states $\mathbf{x}_\kappa$ as input and outputs the required control actions $\hat{\mathbf{u}}_\kappa$. NN_κ consists of only one hidden layer with n_h number of neurons. The NN forward propagation is governed by

$$\hat{\mathbf{u}}_\kappa = \sigma_2(W_2^T\mathbf{z} + \mathbf{b}_2), \ \mathbf{z} = \sigma_1(W_1^T\mathbf{x}_\kappa + \mathbf{b}_1),$$

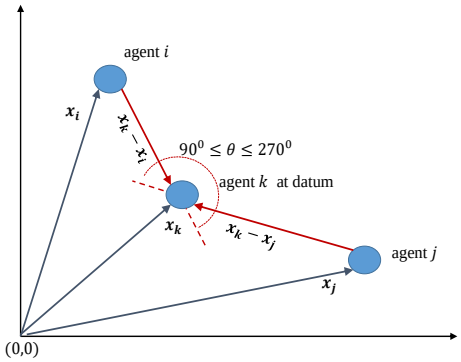


Figure 2: An explanatory vector diagram: the angle between the relative displacement vectors $(\mathbf{x}_k - \mathbf{x}_i)$ and $(\mathbf{x}_k - \mathbf{x}_j)$ with reference to the disconnected agent k located at datum.

where $W_1 \in \Re^{m \times n_h}$, $\mathbf{b}_1 \in \Re^{n_h \times 1}$ denote input to hidden layer weight matrix and bias vector, and $W_2 \in \Re^{n_h \times m}$, $\mathbf{b}_2 \in \Re^{m \times 1}$ are hidden layer to output weight matrix and bias vector, σ_1 and σ_2 stand for input and output activation functions, respectively.
The value function V_κ is computed in a decentralized way during the connected multi-agent dynamics, as follows.

$$\begin{aligned} V_\kappa(t) &= \textstyle\sum_{j \in N_\kappa(t)} \{\|\mathbf{r}_{\kappa j}(t)\| - R_l\}^2 \\ &= \textstyle\sum_{j \in \mathcal{N}_\kappa(t)} \{\|\mathbf{x}_\kappa(t) - \mathbf{x}_j(t)\| - R_l\}^2 \, . \end{aligned} \tag{21}$$

In (21), the value V_κ for agent κ depends on its own and neighbors state information. The associated training error is defined as

$$E_\kappa = \|\hat{\mathbf{u}}_\kappa - \mathbf{u}_\kappa^*\|^2 \; ; \; \mathbf{u}_\kappa^* = -\eta \frac{\partial V_\kappa}{\partial \mathbf{x}_\kappa} \, . \tag{22}$$

Here, $\mathbf{u}_\kappa^*$ represents the desired action along the negative gradient of a decentralized value V_κ, and $\eta > 0$ is a constant. By minimizing the error E_κ iteratively, NN_κ learns a mapping from 'the states' to 'the negative gradient of a decentralized value'.
The weight matrices and bias vectors are updated iteratively according to the back propagation law given as

$$\begin{aligned} W_1^{it+1} &= W_1^{it} - \beta_{NN_\kappa} * dW_1, \; \mathbf{b}_1^{it+1} = \mathbf{b}_1^{it} - \beta_{NN_\kappa} * d\mathbf{b}_1, \\ W_2^{it+1} &= W_2^{it} - \beta_{NN_\kappa} * dW_2, \; \mathbf{b}_2^{it+1} = \mathbf{b}_2^{it} - \beta_{NN_\kappa} * d\mathbf{b}_2, \end{aligned}$$

where $dW_1, dW_2, d\mathbf{b}_1, d\mathbf{b}_2$ are vectorized forms of weight matrix and bias vector differentials; $dW_1 = \frac{\partial E_\kappa}{\partial W_1}$, $d\mathbf{b}_1 = \frac{\partial E_\kappa}{\partial \mathbf{b}_1}$, $dW_2 = \frac{\partial E_\kappa}{\partial W_2}$, $d\mathbf{b}_2 = \frac{\partial E_\kappa}{\partial \mathbf{b}_2}$, and β_{NN_κ} is NN's learning rate. A simple gradient descent method is adopted in the back-propagation to minimize the training error. E_κ decreases over iterations as NN_κ learns, and $\hat{\mathbf{u}}_\kappa \to \mathbf{u}_\kappa^*$ when $W_1 \to W_1^*$, $W_2 \to W_2^*$, $\mathbf{b}_1 \to \mathbf{b}_1^*$, $\mathbf{b}_2 \to \mathbf{b}_2^*$. This trained NN with steady state weights and biases, is used for prediction purpose. The lost agent κ triggers NN_κ action prediction after it reaches its last known nearest neighbor's location (datum). NN_κ generates actions, $\hat{\mathbf{u}}_\kappa = g(\mathbf{x}_\kappa) = \sigma_2(W_2^T.\sigma_1(W_1^T\mathbf{x}_\kappa + \mathbf{b}_1) + \mathbf{b}_2)$, to reconnect agent κ with the rest of the teammates.

Note that every agent tied to a distinct NN learns to cope with the future risk of connectivity loss; however, only the lost agent(s) executes NN-predicted actions to reconnect with others. The proposed method is summarized in *Algorithm 1*.

Algorithm 1: Learn to Restore Connectivity

- Initialize all agent positions in 2D space
- Set parameters: motion, communication and neural network (NN)
- NN training time: t_{tr}, time of disconnection: $t_{dc} > t_{tr}$, time of reconnection: $t_{rc} > t_{dc}$, final time: t_f .

- **while** $t < t_f$
- Determine state-dependent Laplacian matrix: $\mathbf{L}_n(t)$ (2)
- Evaluate current connectivity measure: $\lambda_2(\mathbf{L}_n(t))$
- *Directional Guidance:*
- **if** $0 \leq t < t_{dc}$ and $\lambda_2(t) > 0$
- Update $\forall\ i$ agent states using Consensus Protocol (13)
- *if* $t < t^*_{tr}$
- Compute value function V_i (21)
- Train NN$_i$: $\hat{\mathbf{u}}_i = g(\mathbf{x}_i)$, by minimizing error E_i (22)
- *end if*
- **elseif** $t \geq t_{dc}$ and $0 \leq \lambda_2(t) < \lambda_b$
- *if* $\|\mathbf{x}_\kappa - \mathbf{x}_{datum}\| > \delta$
- Send lost agent $i = \kappa$ to the datum by proportional guidance
- *else*
- Update $i = \kappa$ lost agent state using action prediction $\hat{\mathbf{u}}_\kappa(t)$
- *end if*
- Update $i \neq \kappa$ other agent states using Consensus Protocol (13)
- **elseif** $t \geq t_{rc}$ and $\lambda_2(t) \geq \lambda_b$
- Update all agent states using Consensus Protocol (13)
- **end if**
- Update time axis: $t \rightarrow t + \Delta t$
- **end while**

* NN$_\kappa$ training time t_{tr} lies in $0 < t_{tr} < t_{dc}$ for the first encountered scenario and it equals to 0 in other scenarios.

5. Numerical Results

In simulations, we consider a swarm of five agents flying at a constant altitude to attain a symmetric formation. The total simulation time is $t_f = 37$ secs, with an increment of $\Delta t = 0.001$ sec. Simulation parameters are itemized as follows.

- Motion: $\alpha = 1e-3$, velocity bounds from -7 to 7 m/s.
- Communication: $R_l = 5$ m, $R_u = 40$ m, $\sigma = -5$, and $\delta = 0.2$.
- Action-NNs: $\beta_{NN_2} = 1e-7$, $\beta_{NN_4} = 8e-6$, $\beta_{NN_5} = 8e-7$ $n_h = 10$, $\eta = 1e-3$, $t_{tr} = 1.5$ sec, and activation '*tanh*'.
- Desired formation: Regular pentagon with a separation of 5 m and a relative angle of $\frac{2\pi}{5}$ with respect to the center.

With the same initial condition shown in Table 1, we considered five different scenarios. *Scenario 0* in Fig. 3 represents the ideal case without any communication loss throughout a mission; *Scenario 1* and *Scenario 2* represent two different multi-agent dynamics with single agent loss and recovery; *Scenario 3* and *Scenario 4* represent two different multi-agent dynamics with double agents loss and recovery, as shown in Fig. 4. The time-varying connectivity profiles for all the scenarios are captured in Fig. 5.

Table 1: Initial positions of five agents.

Agent	1	2	3	4	5
$x(t_0)$(m)	-68.35	-53.98	-74.01	-83.19	-83.65
$y(t_0)$(m)	101.57	123.48	138.32	131.30	115.55

Single Agent Loss & Recovery: In Scenario 1, agent 2 gets disconnected after 1.7 sec has passed while the rest four agents $\{1,3,4,5\}$ are pursuing a position consensus. Agent 2 being disconnected at $[-100,\ 107]^T$, can not access any other's information, and so it first goes back to its last known nearest neighbor's location (datum) and then starts searching for other team members using NN_2's action prediction. Fig. 4a shows that the predicted action elegantly

maneuvers agent 2 from the datum towards restoring connectivity. In Fig. 4a, agent 2's trajectory has four intermediate triangle markers; the path between 1st and 2nd ▲s is due to the drift caused by an anomaly and the path between 3rd and 4th ▲s is due to the action-NN_2 prediction. In Scenario 2, agent 2 gets disconnected at the same time but drifts to another location $[0,\ 220]^T$; however, the same action-NN_2 prediction enables it to take an appropriate turn (right from datum) to restore connectivity, as shown in Fig. 4b.

Double Agent Loss & Recovery: In Scenario 3, agent 2 gets disconnected at 1.7 s and agent 4 gets disconnected at 3.0 s. The associated NN_2 and NN_4 predict the required control actions after the lost agents reach their corresponding datum locations, as shown in Fig. 4c. In Scenario 4, agent 2 and agent 5 get disconnected at time instants 1.7 and 2.5 s, respectively. NN_2 starts predicting control actions for agent 2 after it reaches its last known nearest neighbor's (agent 5) location. NN_5 predicts control actions for agent 5 after it reaches its last known nearest neighbor's (agent 1) location. Fig. 4d shows that the lost agent 5 is able to maneuver smoothly from the datum towards restoring connectivity. Note that Scenario 4 is very challenging to solve as the last known nearest neighbor of a lost agent also gets disconnected after a while. All the five agents' velocity profiles/control inputs are shown in Fig. 6a. Fig. 6b presents the evolution of training errors involved in NN_2 and NN_5 for corresponding agents 2 and 5, respectively. Agent legends are coherent in Fig. 3, Fig. 4 and Fig. 6.

The lost agent uses NN predictions to move in the direction of connectivity increase until $\lambda_2 \geq \lambda_b = 0.5$, when the connectivity is restored. After this, all the agents follow the traditional consensus protocol to pursue the desired formation. Fig. 5 shows that the team connectivity measures in Scenarios 1, 2, 3, and 4 are restored around $22,\ 30,\ 27$ and 25 s, respectively. As expected, the desired formations in Scenarios 1-4 are achieved later than that of Scenario 0.

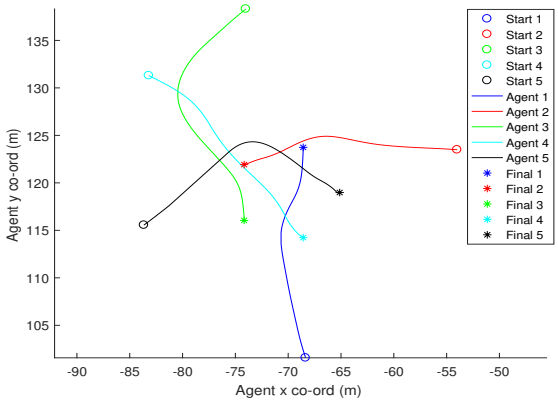


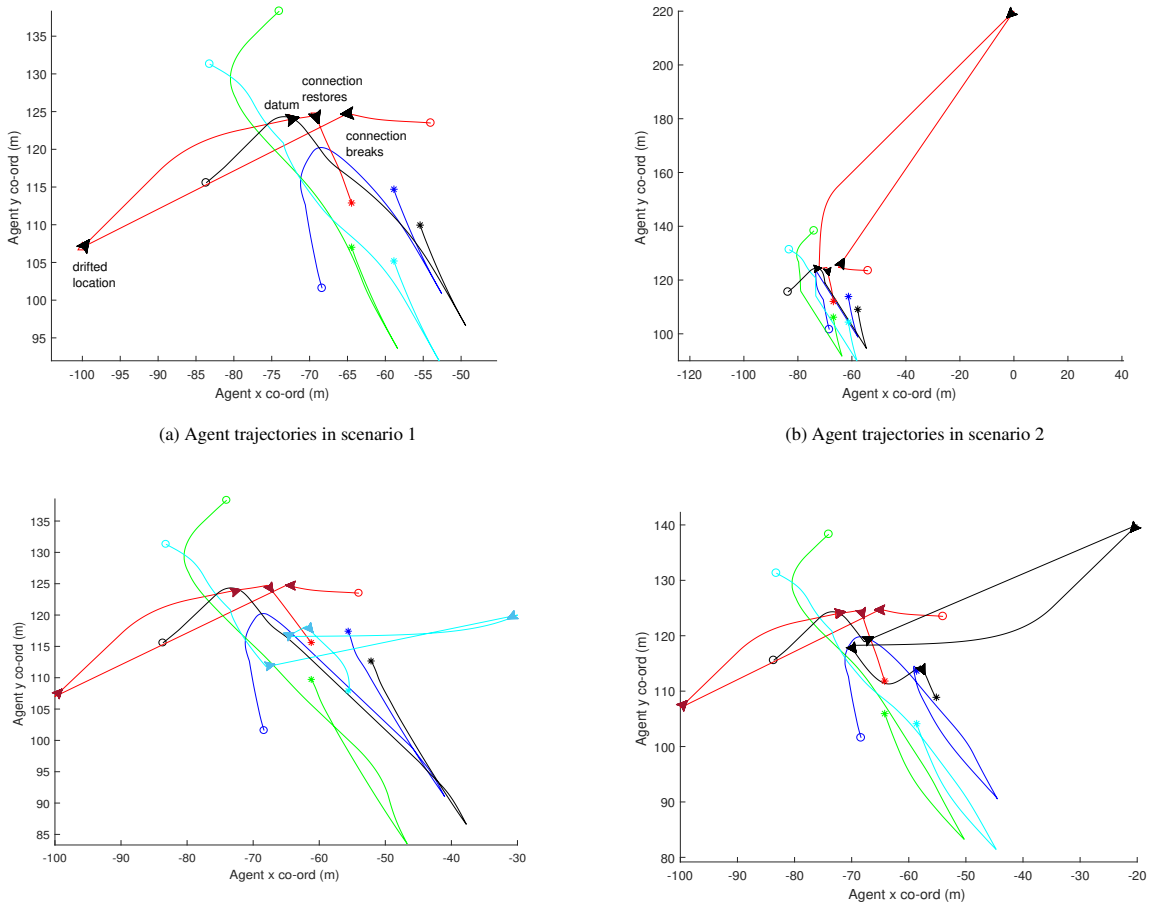
Figure 3: Agent trajectories in scenario 0

5.1. Validating Robustness

To verify the robustness of the proposed method, we vary the time of agent loss, t_{dc}, in different scenarios and record the successful occasions. Precisely, we draw 50 random t_{dc} instances within a range of 1.5 to 10 s, without changing the lost agents' dispersed locations. The success rates reported in Table 2, reveal that *Algorithm 1* performs quite well in Scenarios (1 & 3) with single and double agent loss. For Scenario 3, the success rate is lower and the mean convergence time is higher than Scenario 1, due to the complexity in recovering more agents. In support of the effectiveness of our proposed approach, various ablation studies are presented below.

5.1.1. Performance Comparison

For Scenario 1, we compare the performance of *Algorithm 1* with two naive approaches: (N1) After communication loss, all the agents come back to their initial (t_0) configuration by individual proportional navigation and then use the consensus controller, (N2) The lost agent comes back to the datum and applies its past velocity command just before disconnection. The N2 approach fails to restore the MAS connectivity, as depicted in Fig. 7a. Fig. 7a



(a) Agent trajectories in scenario 1

(b) Agent trajectories in scenario 2

(c) Agent trajectories in scenario 3

(d) Agent trajectories in scenario 4

Figure 4: Agent trajectories in different scenarios; ○ - initial positions, ∗ - final positions, and ▲ - intermediate positions.

Table 2: *Algorithm 1* performance in different scenarios.

Scenario	Single-agent Loss	Double-agent loss
Success rate (%)	100	86
t_f mean (s)	27.71	33.54
variance	1.78	2.67

exhibits that the achieved connectivity profiles with *Algorithm 1* have less transient behavior than those achieved with the naive approaches, also, our algorithm can produce faster response with proper parameter tuning ($\lambda_b = 0.1$ and $\alpha = 0.003$). Note that *Algorithm 1* verifies the resilience of NN by restricting any possible communication on the way (from lost location) to datum, and there onward, it applies the NN-predicted commands to the lost agent for restoring connectivity.

Further, we compare the proposed algorithm's performance with the existing history-following approach [23], where the lost agent moves towards a predefined consensus point until it gains communication with others. In this case, we consider that agent 2 loses communication at 1.7 sec due to a drift at location $[-200, -55]$. The proportional gain in each agent dynamics is selected as $\alpha = 0.002$, and a disturbance of $+1$ m/s is injected into all agent velocities. In Fig. (7b), the connectivity profile achieved with Algorithm 1 ($\delta = 1$) increases faster than that achieved with the history-following approach. Thus, Algorithm 1 is able to cope with the disturbance better than the history-following

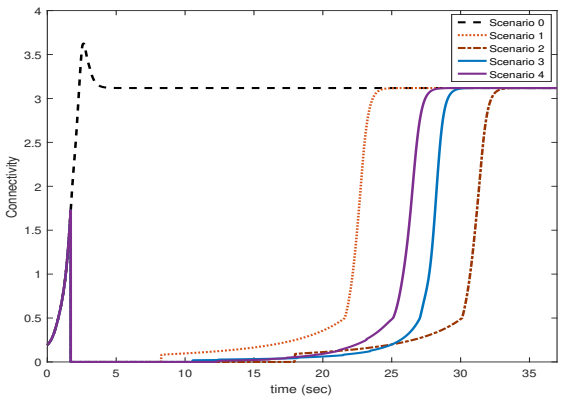
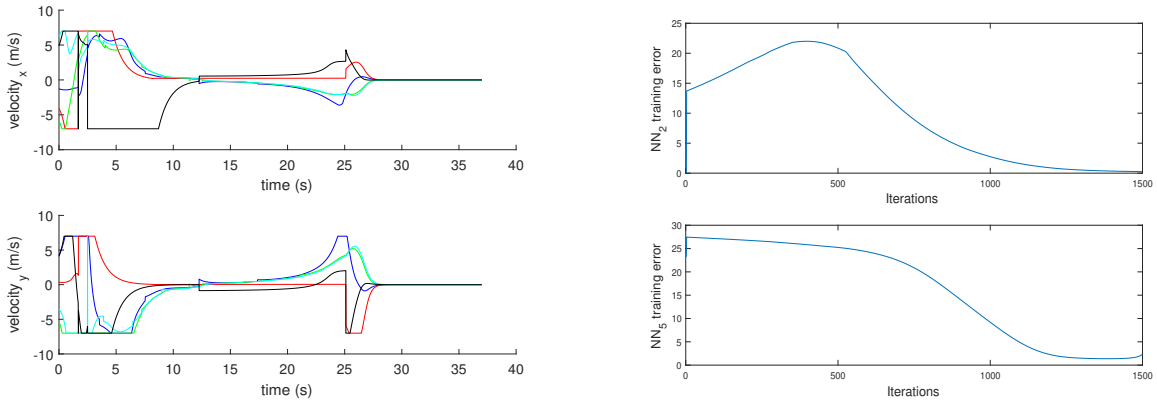


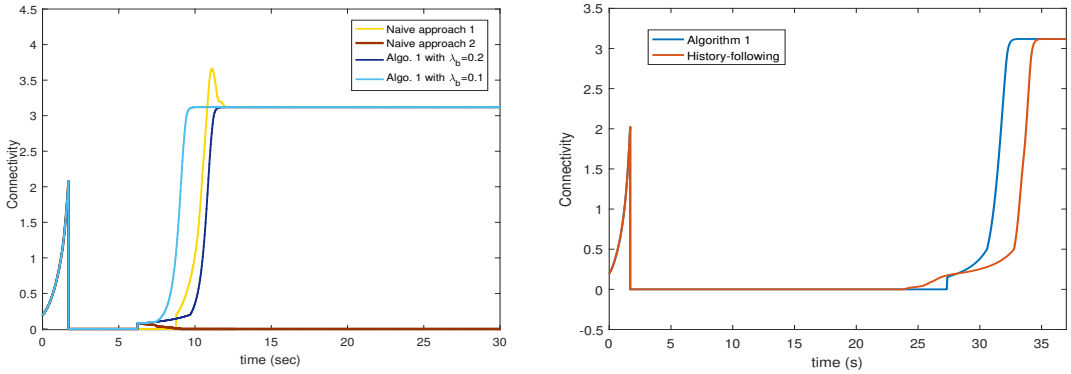
Figure 5: MAS Connectivity profiles in all four scenarios.



(a) Control inputs for all the agents

(b) Evolution of associated NN training errors

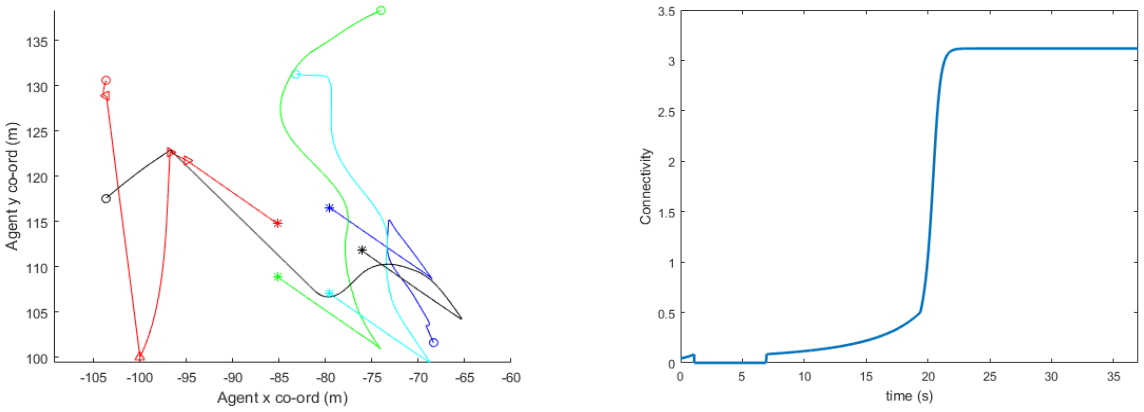
Figure 6: Simulation details of Scenario 4



(a) Algorithm 1 vs naive approaches

(b) Algorithm 1 vs history-following

Figure 7: Different connectivity profiles achieved with Algorithm 1, naive approaches, and history-following.



(a) Agent trajectories

(b) Connectivity profile

Figure 8: Learning performance with a different initial configuration.

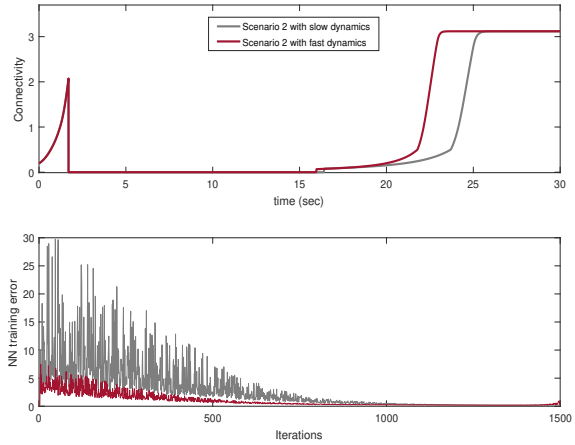
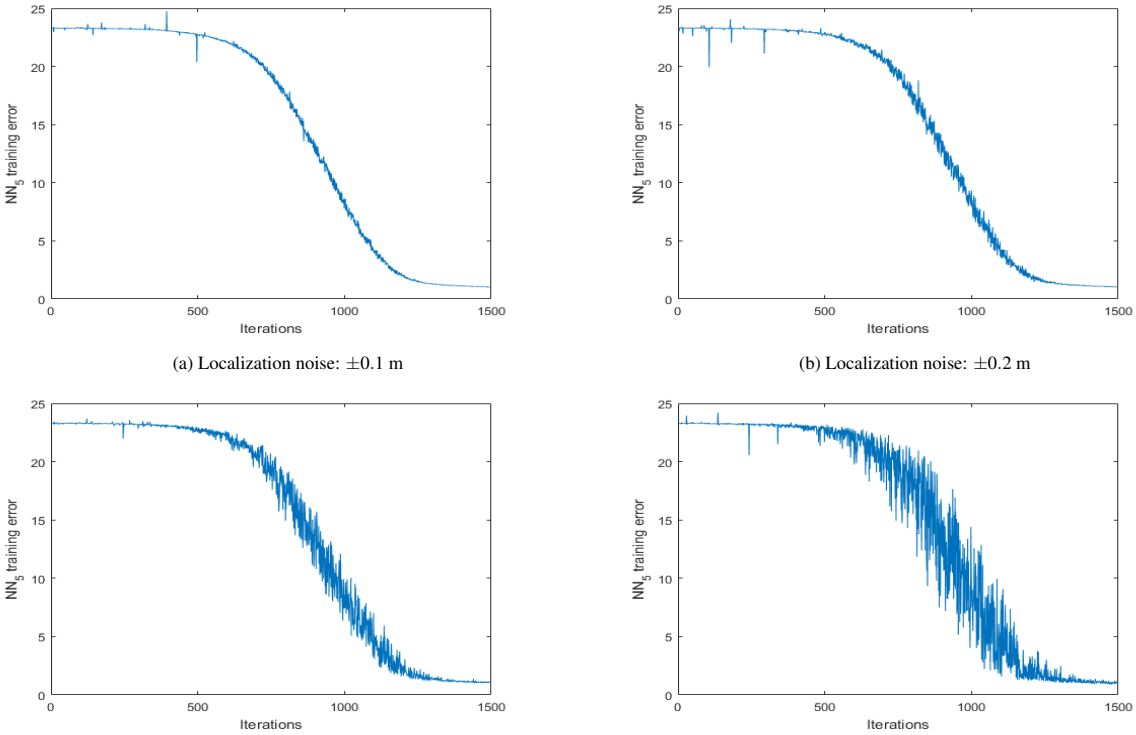


Figure 9: Effect of velocity disturbance in NN_2 training and connectivity for Scenario 2.

and it offers relatively faster consensus convergence. Also, in practice, Algorithm 1 is more viable than the history-following approach because: (i) a predefined consensus point might not be available in an unknown or unexplored area, whereas, the last known nearest neighbor's location (datum) can easily be available to a lost agent; (ii) it is cumbersome to pass information on who is lost and when it is lost to an entire mobile fleet in a decentralized control framework.

5.1.2. Learning to Adapt

Here, we investigate the adaptability of the trained neural networks and the generality of the associated function approximations. While considering a practical situation, we train the associated NNs by collecting data on the fly when connected agents are engaged in pursuing a position consensus. In other words, data is collected in a sample-by-sample manner at every time instant during the connected phase of the MAS dynamics. However, the trained NN for an agent can be applied to different scenarios. In other words, the learnt direction for an agent to restore connectivity, can be utilized for different conditions like initial configuration and time of loss. In this scenario, we use the NN_2 trained in Scenario 1, and exploit its predictions for a different initial condition with a different time of loss. Fig. 8 reveals that the learnt direction is able to adapt to new scenario for connectivity restoration.



(a) Localization noise: ± 0.1 m

(b) Localization noise: ± 0.2 m

(c) Localization noise: ± 0.5 m

(d) Localization noise: ± 1 m

Figure 10: Evolution training errors different noise levels.

5.1.3. Effect of Noise

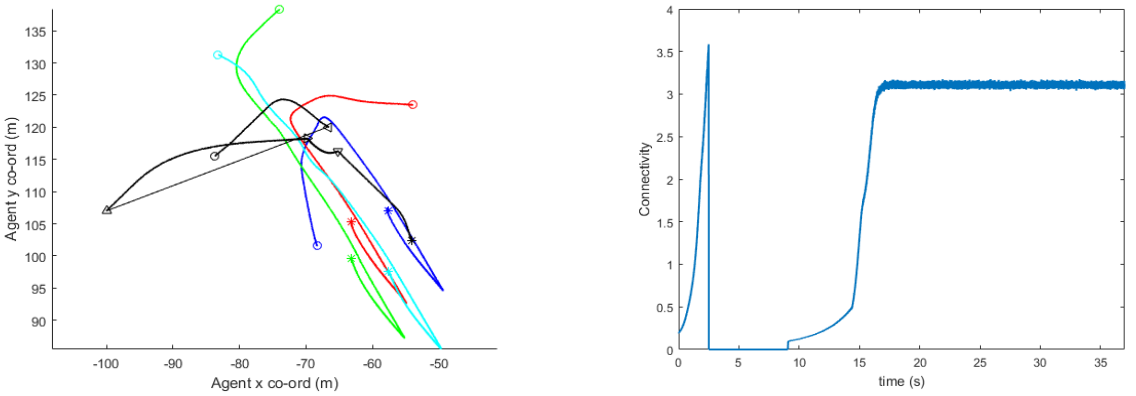
Further, to validate the performance of the proposed learning approach in the presence of disturbance, we inject Gaussian randomness $\pm 5\%$ $v_{\max}$ into agent velocities (2D) for slow ($\alpha = 0.002$) and fast ($\alpha = 0.003$) agent dynamics in Scenario 2. Fig. 9 justifies that the associated NN is able to learn even if agent velocities contain disturbances to some extent. The effect of disturbance reduces when the multi-agent dynamics is comparatively faster.

To examine the effect of localization noise in the proposed decentralized learning, we add different magnitudes of Gaussian noise in agent states. The initial configuration remains the same as mentioned in Table I. In the presence of localization noise, agent 5 gets disconnected at 2.5 sec. The associated NN_5 is trained for the initial 1.5 sec of the dynamics. Interestingly, NN_5 predicted actions are able to steer agent 5 effectively to restore connectivity. In this case, we reduced the reference magnitude with $\eta = 0.001/200$ to train NN_5. Fig. 10 shows the learning performance with different amounts of noise ranging from ± 0.1 to ± 1 m. The agent trajectories along with the connectivity profile obtained for a localization noise of ± 0.1 m, are presented in Fig 11.

5.1.4. Scalability

Moreover, we consider the initial configuration as mentioned in Table I. In this scenario, three agents get disconnected consecutively. Agent 2, 5 and 4 are lost at 1.7, 2.5 and 3.1 sec, respectively. The associated NN_2, NN_5 and NN_4 are trained for the initial 1.3 sec of the dynamics, and the predictions are executed after the lost agents reach their corresponding datum locations. Fig. 12 shows that the NN-predicted actions are able to steer the lost agents towards restoring connectivity.

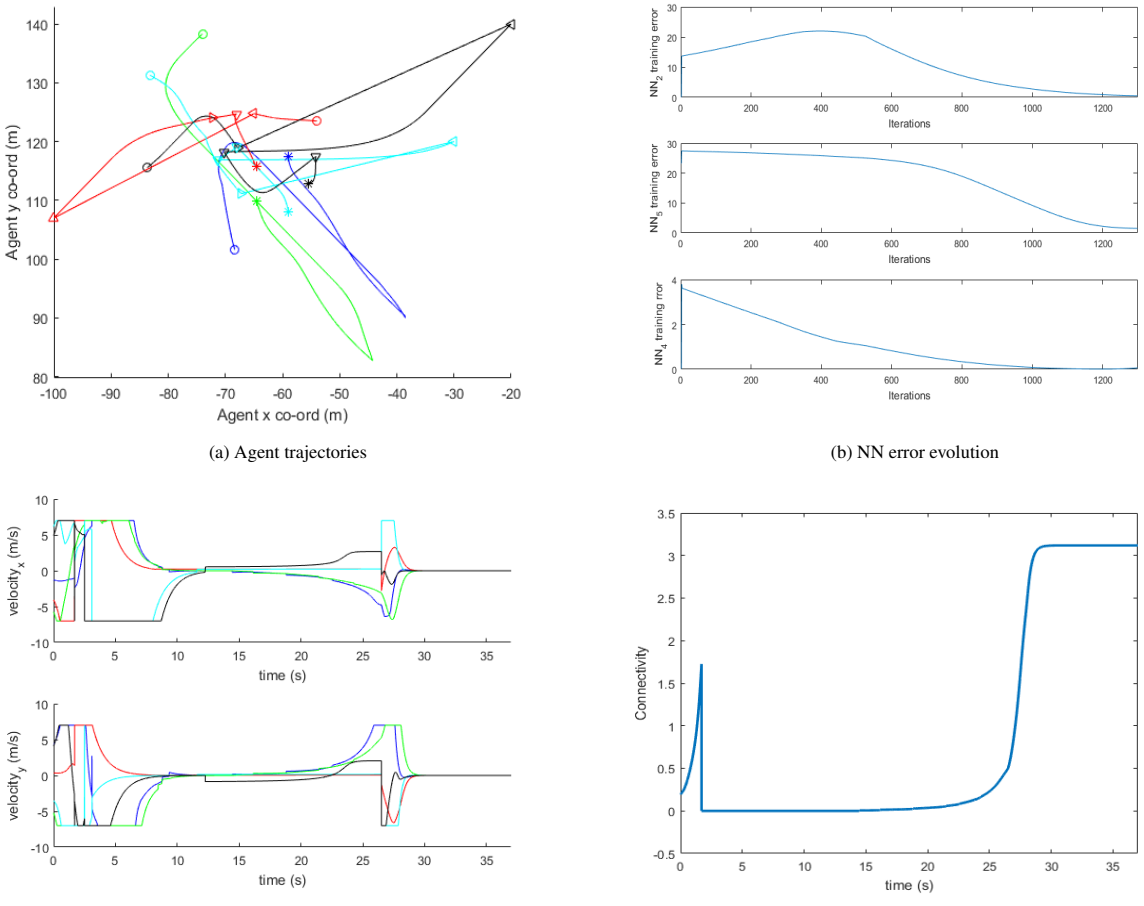
Discussion: Simulation results justify that the proposed neural network based decentralized learning strategy effectively drives the lost agents to the locations of connectivity restoration. Interestingly, the associated neural networks are able to quickly learn the required mapping to raise connectivity, even in the presence of disturbances. Nevertheless, it is tough for one agent to raise the entire team's connectivity up to a desired level without accessing any neighbor's information. During a mission, a function approximator associated with each agent is trained with the data collected



(a) Agent trajectories

(b) NN error evolution

Figure 11: Agent trajectories and connectivity profile with a localization noise of ± 0.1 m.



(a) Agent trajectories

(b) NN error evolution

(c) Agent velocities

(d) Connectivity profile

Figure 12: Decentralized learning performance in connectivity restoration with three agent loss and recovery; $\circ$ - initial positions, $*$ - final positions, and $\triangle$ - intermediate positions.

on the fly; however, the learnt mapping that steers an agent along the optimal direction to restore connectivity, is preserved and can adapt to different scenarios. Therefore, the same NN_2 trained online during Scenario 1 works for Scenario 2 as well. Such a learning strategy is not restricted to robots with known kinematics only.

6. CONCLUSION

This work presents a proof-of-concept to show the impact of a decentralized learning method in tackling sudden changes in communication during multi-agent formation control. Simulation results demonstrate that our proposed strategy offers appropriate maneuvers to recover single and double disconnected agents while others are engaged in pursuing a position consensus. Neural network generated coordinated movements are effective even in the presence of disturbance, without accessing any neighborhood-based information. To this end, we have established a unique idea to restore connectivity during formation control by leveraging the power of function approximation when information sharing is impossible in a cooperative mission.

In the future, we plan to extend the study to a large team with many link failures, which demands finer approximations that can be achieved by incorporating deep neural architecture and employing advanced optimization techniques in back-propagation.

7. Appendix

Closed-loop Stability of Connected MAS: The dynamic stability of the closed-loop system governed by (14) can be established by using Lyapunov theory. Consider a Lyapunov candidate function, $V(t) > 0$ for all $\tilde{\mathbf{X}}(t) \neq 0$, as follows.

$$V(t) = \frac{1}{2}\tilde{\mathbf{X}}^T(t)\,\tilde{\mathbf{X}}(t) \tag{23}$$

Now, we take time-derivatives on both sides of (23) and substitute the underlying dynamics (14), which leads to

$$\begin{aligned} \dot{V}(t) = \tilde{\mathbf{X}}^T(t)\,\dot{\tilde{\mathbf{X}}}(t) &= \tilde{\mathbf{X}}^T(t)\{-\alpha(\mathbf{L}_n \otimes I_m)\tilde{\mathbf{X}}(t)\} \\ \text{or, } \dot{V}(t) &= -\alpha\,\tilde{\mathbf{X}}^T(t)(\mathbf{L}_n \otimes I_m)\tilde{\mathbf{X}}(t)\,. \end{aligned} \tag{24}$$

$\dot{V}(t)$ in (24) is negative for all $\tilde{\mathbf{X}}(t)$, except $V(t) = 0$ at $\tilde{\mathbf{X}}(t) = c \times \mathbf{1}_{nm}$ for $c \in \Re^1$. Thus, the asymptotic stability of the closed-loop system is ensured, i.e. $\tilde{\mathbf{X}}(t) \to c \times \mathbf{1}_{nm}$ as $t \to 0$, which implies $(\mathbf{L}_n(t) \otimes I_m)\tilde{\mathbf{X}}(t) \to 0 \implies \tilde{\mathbf{x}}_i(t) - \tilde{\mathbf{x}}_j(t) \to \mathbf{0}$ or, $\mathbf{x}_i(t) - \mathbf{x}_j(t) \to \mathbf{x}_{fi} - \mathbf{x}_{fj}$.

References

[1] W. Ren, R. Beard, E. Atkins, A survey of consensus problems in multi-agent coordination, IEEE American Control Conference (2005) 1859–1864.

[2] W. Ren, Multi-vehicle consensus with a time-varying reference state, Systems & Control Letters 56 (2007) 474–483.

[3] X. Ge, Q. Han, D. Ding, X. Zhang, B. Ning, A survey on recent advances in distributed sampled-data cooperative control of multi-agent systems, Neurocomputing 275 (2018) 1684–1701.

[4] R. Olfati-Saber, J. Fax, R. Murray, Consensus and cooperation in networked multi-agent systems, Proceedings of the IEEE 95 (2007) 215–233.

[5] M. Gennaro, A. Jadbabaie, Decentralized control of connectivity for multi-agent systems, IEEE Conference on Decision and Control (2006) 3628–3633.

[6] M. Zavlanos, G. Pappas, Controlling connectivity of dynamic graphs, Proc. of the IEEE Decision and Control (2005) 6388–6393.

[7] R. Dutta, L. Sun, M. Kothari, R. Sharma, D. Pack, A cooperative formation control strategy maintaining connectivity of a multi-agent system, IEEE/RSJ International Conference on Intelligent Robots and Systems (2014) 1189–1194.

[8] M. Zavlanos, G. Pappas, Potential fields for maintaining connectivity of mobile networks, IEEE Transactions on robotics 23 (2007) 812–816.

[9] L. Barnes, M. Fields, K. Valavanis, Unmanned ground vehicle swarm formation control using potential fields, IEEE Mediterranean Conference on Control & Automation (2007) 1–8.

[10] W. Hu, L. Liu, G. Feng, Consensus of linear multi-agent systems by distributed event-triggered strategy, IEEE Transactions on Cybernetics 46 (2015) 148–157.

[11] K. Saulnier, D. Saldana, A. Prorok, G. Pappas, V. Kumar, Resilient flocking for mobile robot teams, IEEE Robotics and Automation letters 2 (2017) 1039–1046.

[12] J. Zhang, Z. Wu, L. Hong, X. Xu, Connectivity recovery of multi-agent systems based on connecting neighbor set, Physica A: Statistical Mechanics and its Applications 390 (2011) 4596–4601.

[13] R. Olfati-Saber, Flocking for multi-agent dynamic systems: Algorithms and theory, IEEE Transactions on Automatic Control 51 (2006) 401–420.
[14] R. Ramachandran, N. Fronda, G. Sukhatme, Resilience in multi-robot target tracking through reconfiguration, IEEE International Conference on Robotics and Automation 2 (2020) 4551–4557.
[15] Y. Kantaros, M. Guo, M. Zavlanos, Temporal logic task planning and intermittent connectivity control of mobile robot networks, IEEE Transactions on Automatic Control 64 (2019) 4105–4120.
[16] Y. Tang, H. Gao, W. Zou, J. Kurths, Temporal logic task planning and intermittent connectivity control of mobile robot networks, IEEE Transactions on Cybernetics 43 (2012) 358–370.
[17] X. Yu, M. Hsieh, Synthesis of a time-varying communication network by robot teams with information propagation guarantees, IEEE Robotics and Automation Letters 5 (2020) 1413–1420.
[18] F. Bourgault, T. Furukawa, H. Durrant-Whyte, Optimal search for a lost target in a bayesian world, Field and service robotics (2003) 209–222.
[19] E. Wong, F. Bourgault, T. Furukawa, Multi-vehicle bayesian search for multiple lost targets, Proceedings of the 2005 IEEE international conference on robotics and automation (2005) 3169–3174.
[20] J. S. H. Kandath, S. Sundaram, Mutli-agent consensus under communication failure using actor-critic reinforcement learning, IEEE Symposium Series on Computational Intelligence (2018) 1461–1465.
[21] L. Li, D. Ho, J. Lu, A consensus recovery approach to nonlinear multi-agent system under node failure, Information Sciences 367 (2016) 975–989.
[22] D. Acharya, S. K. Mishra, Optimal consensus recovery of multi-agent system subjected to agent failure, International Journal on Artificial Intelligence Tools 29 (2020) 2050017.
[23] Y. Karteek, I. Kar, S. Majhi, Consensus of multi-agent systems using back-tracking and history following algorithms, arXiv preprint arXiv:2011.08990.
[24] M. Fiedler, Algebraic connectivity of graphs, Czechoslovak Mathematical Journal 23 (1973) 298–305.
[25] K. Murphy, Machine learning: a probabilistic perspective, MIT Press, 2012.
[26] D. Ding, Z. Wang, Q. Han, Neural-network-based consensus control for multiagent systems with input constraints: the event-triggered case, IEEE Transactions on Cybernetics 50 (2019) 3719–3730.
[27] S. Yang, W. Bai, T. Li, Q. Shi, Y. Yang, Y. Wu, C. Chen, Neural-network-based formation control with collision, obstacle avoidance and connectivity maintenance for a class of second-order nonlinear multi-agent systems, Neurocomputing 439 (2021) 243–255.
[28] D. Liu, C. Li, H. Li, D. Wang, H. Ma, Neural-network-based decentralized control of continuous-time nonlinear interconnected systems with unknown dynamics, Neurocomputing 165 (2015) 90–98.
[29] K. Vamvoudakis, F. Lewis, Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem, Automatica 46 (2010) 878–888.
[30] C. Arvind, J. Senthilnath, Autonomous vehicle for obstacle detection and avoidance using reinforcement learning, Soft Computing for Problem Solving (2020) 55–66.
[31] H. Zhang, L. Cui, X. Zhang, Y. Luo, Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method, IEEE Transactions on Neural Networks 22 (2011) 2226–2236.
[32] C. Mu, Z. Ni, C. Sun, H. He, Data-driven tracking control with adaptive dynamic programming for a class of continuous-time nonlinear systems, IEEE Transactions on Cybernetics 47 (2016) 1460–1470.
[33] C. Godsil, G. F. Royle, Algebraic graph theory, 207th Edition, Springer Science & Business Media, 2013.
[34] D. Kempe, F. McSherry, A decentralized algorithm for spectral analysis, Journal of Computer and System Sciences 74 (2008) 70–83.