# Graph Matching-Based Formation Reconfiguration of Networked Agents With Connectivity Maintenance

Zhen Kan, Leenhapat Navaravong, John M. Shea, Eduardo L. Pasiliao, Jr., and Warren E. Dixon

***Abstract*—Various applications require networked agents to cooperatively achieve specified formations. In this paper, formation reconfiguration for a group of identical agents with limited communication capabilities is considered. Since the considered agents are identical, their roles are interchangeable, and each position in the desired formation can be taken by any agent. To reduce the total amount of node movement required for formation reconfiguration, a weighted graph-matching-based node-mapping strategy is developed to specify the node correspondence between an arbitrary initial graph and the desired graph. After the node mapping is determined, agents are required to move physically to form the desired formation. Since agents are only able to communicate within a certain range, formation reconfiguration must be accomplished with network connectivity constraints (i.e., specified nodes remain within specified sensing and communication ranges). A decentralized control scheme is developed to guarantee network connectivity by maintaining a desired neighborhood determined by the node-mapping algorithm, and to ensure convergence of all agents to the desired configuration with collision avoidance among agents. The developed strategy is demonstrated through simulation results.**

***Index Terms*—Formation reconfiguration, graph matching, network connectivity.**

## I. Introduction

GREAT efficiency and operational capability can be realized by networked agents in various civilian and military applications. To enable these applications, agents are required to perform in a coordinated manner through their interactions, which are generally captured by the underlying network graph. The graph determines which agents can exchange and share information and how robust the group can behave in a dynamic environment. For example, the formations developed in [1] and [2] are beneficial in data gathering, data processing, and forecasting in surveillance and exploration. The graphs designed in [3] are robust with respect to maintaining a required level of network connectivity in the presence of node and link failures. In consensus applications (see [4] and [5] for a comprehensive literature review for consensus problems), different topologies yield different consensus rates [6]–[8]. Although designing an optimal graph with respect to the consensus rate or efficiency of information collection has attracted much research attention, achieving the designed optimal graph from an arbitrary initial graph with minimum movement and constraints on communication is still a problem of wide interest.

Graph matching is widely used in pattern recognition, such as computer vision, scene analysis, chemistry, and biology, where the relationships and interactions between objects are modeled as graphs. To identify the similarities between two different graphs (i.e., patterns), various graph-matching methods have been developed to identify vertex correspondence between graphs. If two graphs are isomorphic, Ullmann's algorithm [9] can be directly used to obtain node correspondence. However, Ullmann's algorithm is only applicable to isomorphic graphs, and the graphs are generally not isomorphic in most applications. In addition, the complexity of Ullmann's algorithm is $O(N^N)$, which requires significant computational resources for large graphs. An alternative to Ullmann's algorithm is to find an approximate solution of the optimal matching. In [10], an efficient Eigen-decomposition approach is developed to find approximated optimal matching in terms of minimizing the aggregated edge weights between two graphs. Other approximated approaches based on spectral representation of graphs include [11]–[13]. However, all of the aforementioned results focus on identifying the similarities between two graphs, without considering the mapping of nodes in terms of graph reconfiguration.

In this paper, the formation reconfiguration of networked agents from an arbitrary initial network graph to a desired graph is considered. Each agent is represented as a node, and the local interaction among agents is modeled by an undirected graph with each edge representing the neighborhood between two agents. The model is based on the assumption that each agent has limited communication capability (i.e., available information exchange by agents within a certain range). A connected graph indicates that the agents are able to exchange information with other agents and coordinate their motion to achieve the desired topology. Hence, two main objectives are as follows: 1) minimize the node movement required during formation

Manuscript received March 4, 2014; revised August 30, 2014; accepted October 3, 2014. Date of publication November 12, 2014; date of current version March 13, 2015. This work was supported in part by the National Science Foundation under award numbers 1161260 and 1217908, and a contract with the AFRL Mathematical Modeling and Optimization Institute. Recommended by Associate Editor Michael Chertkov.

Z. Kan and W. E. Dixon are with the Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL USA (e-mail: kanzhen0322@ufl.edu; wdixon@ufl.edu).

L. Navaravong and J. M. Shea are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL USA (e-mail: leenhapat@ufl.edu; jshea@ece.ufl.edu).

E. L. Pasiliao, Jr. is with Munitions Directorate, Air Force Research Laboratory, Eglin Air Force Base, FL 32542 USA (e-mail: pasiliao@eglin.af.mil).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCNS.2014.2367363

reconfiguration in terms of the number of edges that a node must traverse and 2) physically steer the agents to achieve the desired formation while preserving network connectivity and avoiding collisions among agents. In our previous work [14], the physical formation control of a group of agents with constraints on network connectivity is investigated. However, the initial topology in [14] is assumed to be a supergraph of the desired topology, which ensures that the agents are originally in a feasible interconnected state. The supergraph assumption may not be applicable given an arbitrary initial graph as in the current result. In this paper, based on the weighted graph-matching algorithm in [10], a node-mapping algorithm is developed to determine the node correspondence between the arbitrary initial topology and the specified desired topology, and build a tree on the initial graph such that the node movement required in topology reconfiguration is minimized and the routing algorithm developed from our previous work [15]–[17] can be applied to specify how the initial graph can be transformed to the desired graph.

After node correspondence is determined, nodes are required to physically move to perform network reconfiguration with limited communication. Generally speaking, formation control focuses on the control design for a group of agents to stabilize at a specific geometric formation or move in the environment, keeping a suitable relative configuration. The network reconfiguration problem is a subset of formation control problems which considers additional constraints, such as network connectivity. Some representative results in controlling mobile robot networks while preserving network connectivity are surveyed in the work of [18]. Other results on formation control include [19]–[26], where various control methods are developed to reorganize the formation of networked agents with limited communication capabilities.

In contrast to the results such as [19]–[23], this work considers identical agents which can interchange their roles during formation control, allowing the neighborhood for each agent to be dynamically determined, that is, which nodes in the initial graph that will take which positions in the final graph are not specified in advance; rather, the objective only requires that there be an agent in each position specified in the final graph. Although identical agents are considered in the works of [24]–[27] for formation control, the preservation of network connectivity is not considered in [24], and no effort is made in [25] and [26] to reduce the amount of node movement in formation reconfiguration. To preserve a connected network, network connectivity is modeled as an artificial obstacle. A navigation function (cf., [28]) based control scheme is developed to ensure the convergence of all agents to the desired configuration; collision avoidance among agents and network connectivity maintenance are achieved through only local communication. An information flow model is then proposed based on the work of [29] and [30] to specify the required movement for agents to their destination nodes. Compared to the works of [22], [23], and [27] the information flow-based approach generally provides a path with more freedom of motion without disconnecting the network and allows communication links to be formed or broken dynamically. Consensus is proven using Rantzer's Dual Lyapunov Theorem [31]. Simulation results are provided to demonstrate the developed network reorganization strategy.

## II. Problem Formulation

Consider $N$-networked agents moving in a workspace $\mathcal{F}$ according to

$$\dot{q}_i = u_i,\ i = 1, \cdots, N \tag{1}$$

where $q_i = [x_i\ \ y_i]^T \in \mathbb{R}^2$ and $u_i \in \mathbb{R}^2$ denote the position and velocity (i.e., the control input) of agent $i$, respectively. Assume that the workspace $\mathcal{F}$ is circular and bounded with radius $R \in \mathbb{R}^+$, and the agents in $\mathcal{F}$ are identical and have limited communication capabilities such that two agents can only exchange information through communication within an interdistance $R_c < R$. Communication between neighboring agents is assumed to be error and delay free in this work. A collision region is defined as a small disk area with radius $\delta_1 < R_c$ centered at agent $i$, such that the presence of any other agent $j$ within this region is considered as a potential collision for agent $i$. To ensure the availability of communication between agents $i$ and $j$, an escape region for each agent $i$ is defined as the outer ring of the communication area with radius $r$, where $R_c - \delta_2 < r < R_c$ and $\delta_2 \in \mathbb{R}^+$ is a predetermined buffer distance. Agent $i$ moves with the constraint of avoiding a collision with other agents located in the collision region, and preventing a break in the communication link between agents located in the escape region. The region within the collision region and the escape region (i.e., $\delta_1 < r < R_c - \delta_2$,) is a constraint-free region.

The interaction among agents is modeled as a simple graph $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t))$, with $\mathcal{V}$ denoting the set of nodes (i.e., agents) and $\mathcal{E}(t) = \{(v_i, v_j) \in \mathcal{V} \times \mathcal{V} | d_{ij}(t) \leq R_c\}$ denoting the set of edges, where $d_{ij}(t) = \|q_i - q_j\|$ is the Euclidean distance between $v_i$ and $v_j$. The neighbors of $v_i$ are defined as $\mathcal{N}_i(t) = \{v_j | v_j \in \mathcal{V}, (v_i, v_j) \in \mathcal{E}\}$. The initial and desired final graphs are represented by $\mathcal{G}_{\text{int}} = (\mathcal{V}_{\text{int}}, \mathcal{E}_{\text{int}})$ and $\mathcal{G}_f = (\mathcal{V}_f, \mathcal{E}_f)$, respectively. The final graph $\mathcal{G}_f$ is characterized by the relative positions $\{c_{ij} \in \mathbb{R}^2 | v_i, v_j \in \mathcal{N}_i^f, \}$ where each $c_{ij}$ is a predetermined constant that specifies the physical configuration of $\mathcal{G}_f$. Here, $\mathcal{N}_i^f$ is a predefined set of neighbors for $v_i$ in $\mathcal{G}_f$. That is, the desired position $q_{di}$ for $v_i$ in $\mathcal{G}_f$ is defined as $q_{di} = \{q_i | \|q_i - q_j - c_{ij}\|^2 = 0,\ v_j \in \mathcal{N}_i^f\}$. Note that the set $\mathcal{N}_i$ is time varying and depends on the relative distance of mobile agents, while $\mathcal{N}_i^f$ is constant and specified by $\mathcal{G}_f$. A graph is connected if a path exists that connects any two nodes. A tree is a particular topology on an undirected graph, where any two vertices are connected by exactly one simple path. A tree that contains all nodes in the graph is called a spanning tree. The control algorithm for repositioning nodes developed in this paper repositions the nodes by transforming a spanning tree of the initial graph into a spanning tree of the final graph. Thus, in most of what follows, we focus on the case of achieving a desired topology $\mathcal{G}_f$ that is a spanning tree, and we assume that the root is predetermined.

The control algorithm for repositioning the nodes uses an information flow between pairs of nodes that determines a path of movement along the nodes in the graph for at least

one of the nodes. In this paper, these paths of movement are determined by using an approach first proposed in our previous work [17]. In [17], a prefix labeling and routing algorithm is developed to "route" each autonomous vehicle through an initial tree topology to achieve the desired tree topology while preserving network connectivity. However, finding a good node mapping to reduce the amount of node movement in topology reconfiguration is not considered in [17]. In this paper, since $\mathcal{G}_f$ can be assumed to be a spanning tree, it is desirable to find a tree $\mathcal{G}_{\text{int}}^t$ in the specified $\mathcal{G}_{\text{int}}$ that minimizes the node movement required in topology reconfiguration, and the algorithm in [17] can be applied to determine how $\mathcal{G}_{\text{int}}$ can be transformed into $\mathcal{G}_f$. In particular, in Section III, weighed graph matching is used to find a node mapping between $\mathcal{G}_{\text{int}}$ and $\mathcal{G}_f$, based on which an initial tree $\mathcal{G}_{\text{int}}^t$ is then built, so that the amount of node movement in topology reconfiguration is minimized. After node correspondence is determined, a motion control algorithm is developed in Section IV to physically steer each agent toward the desired formation while maintaining network connectivity and avoiding collision among agents. To achieve these two objectives, the following assumptions are required.

*Assumption 1:* The initial graph $\mathcal{G}_{\text{int}}$ is connected and the initial positions do not coincide with some unstable equilibria (e.g., nodes colliding).

*Assumption 2:* The connected desired graph $\mathcal{G}_f$ is prespecified and achievable (i.e., $\delta_1 < \|c_{ij}\| < R_c - \delta_2$).

## III. Node-Mapping Strategy

To achieve the desired $\mathcal{G}_f$ from an arbitrary initial $\mathcal{G}_{\text{int}}$ while preserving network connectivity, the developed strategy consists of two stages: 1) a weighed graph-matching-based node-mapping algorithm over the network topology and 2) a potential field-based motion control algorithm on the physical graph. The node-mapping algorithm determines which node in the initial topology should take which position in the final topology, and specify how the initial topology can be transformed into the desired topology.

### A. Weighted Graph Matching

Since identical agents are considered and the mapping between the agents in $\mathcal{G}_{\text{int}}$ and $\mathcal{G}_f$ is not specified in advance, the goal of this section is to determine a bijective mapping $\Phi : \mathcal{V}_{\text{int}} \to \mathcal{V}_f$ such that the amount of movement (in terms of number of edges) required to achieve $\mathcal{G}_f$ from $\mathcal{G}_{\text{int}}$ is minimized. Similar to [10], $\Phi$ is chosen to minimize a cost function $J(\Phi)$ based on the differences of $\mathcal{G}_f$ and $\mathcal{G}_{\text{int}}$ as

$$J(\Phi) = \sum_{i=1}^{N}\sum_{j=1}^{N}\left[w_{\text{int}}(v_i, v_j) - w_f\left(\Phi(v_i), \Phi(v_j)\right)\right]^2 \qquad (2)$$

where $w_{\text{int}}(\cdot)$ and $w_f(\cdot)$ are weighting functions that specify the importance of edges in $\mathcal{G}_{\text{int}}$ and $\mathcal{G}_f$, respectively. Replacing $\Phi$ in (2) with a permutation matrix $\mathbf{P}$ yields

$$J(\mathbf{P}) = \left\|\mathbf{P}\mathbf{A}_{\mathcal{G}_{\text{int}}}\mathbf{P}^T - \mathbf{A}_{\mathcal{G}_f}\right\| \qquad (3)$$

where $\|\cdot\|$ denotes the Euclidean norm, and $\mathbf{A}_{\mathcal{G}_{\text{int}}}$ and $\mathbf{A}_{\mathcal{G}_f}$ are weighted adjacency matrices for $\mathcal{G}_{\text{int}}$ and $\mathcal{G}_f$. Note that if $\mathcal{G}_{\text{int}}$ and $\mathcal{G}_f$ correspond to a one-to-one isomorphism, then $J(\mathbf{P}) = 0$ and $\mathbf{P}\mathbf{A}_{\mathcal{G}_{\text{int}}}\mathbf{P}^T = \mathbf{A}_{\mathcal{G}_f}$. By relaxing the domain of $\mathbf{P}$ from the set of permutation matrices to the set of orthogonal matrices, (3) can be solved approximately using the graph spectra. Let the Eigen-decomposition[1] of $\mathbf{A}_{\mathcal{G}_{\text{int}}}$ and $\mathbf{A}_{\mathcal{G}_f}$ be $\mathbf{A}_{\mathcal{G}_{\text{int}}} = \mathbf{U}_{\text{int}}\Lambda_{\text{int}}\mathbf{U}_{\text{int}}^T$ and $\mathbf{A}_{\mathcal{G}_f} = \mathbf{U}_f\Lambda_f\mathbf{U}_f^T$, where $\mathbf{U}_{\text{int}}$ and $\mathbf{U}_f$ are orthogonal matrices, and $\Lambda_{\text{int}}$ and $\Lambda_f$ are diagonal matrices with distinct eigenvalues. Furthermore, let $\bar{\mathbf{U}}_{\text{int}}$ and $\bar{\mathbf{U}}_f$ be matrices for which each element is the absolute value of the corresponding element in $\mathbf{U}_{\text{int}}$ and $\mathbf{U}_f$, respectively. Letting $\mathbf{\Pi}$ denote the set of permutation matrices, the approximate solution to (3) is given by [10] as

$$\arg\max_{\mathbf{P}\in\mathbf{\Pi}} \operatorname{tr}\left(\mathbf{P}^T\bar{\mathbf{U}}_f\bar{\mathbf{U}}_{\text{int}}^T\right) \qquad (4)$$

which can be solved by the Hungarian method in $O(N^3)$ time [32].

The choice of weighting functions in (2), which determines the weighted adjacency matrices in (3), greatly affects the solution of (4). Inappropriate weighting functions may result in a mapping $\Phi$ where $\mathcal{G}_{\text{int}}$ does not preserve most of the edges in $\mathcal{G}_f$, leading to redundant movement for agents during graph reconfiguration. Different from the work in [10], where the weights of edges are known in advance, to facilitate graph reconfiguration in this paper, unequal priorities are assigned to the edges in $\mathcal{G}_f$ so that most edges in $\mathcal{G}_f$ are preserved in $\mathcal{G}_{\text{int}}$. For instance, an edge in $\mathcal{G}_f$ that is closer to the root node with many descendants should have greater priority than edges that are further down the tree with fewer descendants. Following this idea, the weighting function $w_f$ for an edge $(u, v) \in \mathcal{E}_f$, except for the edges attached to leaf nodes in $\mathcal{G}_f$, is defined as

$$w_f(u, v) = |c_v| \cdot (\text{maxdepth}(c_v) - \text{depth}(v) + 1)$$

where $u, v \in \mathcal{V}_f$ denotes a parent node and child node,[2] respectively, $|c_v|$ denotes the cardinality of $c_v$, where $c_v$ indicates the set of descendants of $v$ in $\mathcal{G}_f$, $\text{depth}(v)$ denotes the depth of node $v \in \mathcal{V}_f$, and $\text{maxdepth}(c_v)$ denotes the maximum node depth among all nodes in $c_v$. Edges attached to leaf nodes in $\mathcal{G}_f$ are assigned a weight of 1. To preserve most of the edges in $\mathcal{G}_{\text{int}}$, which correspond to the edges with large weights in $\mathcal{G}_f$, the weight of edges in $\mathcal{G}_{\text{int}}$ is assigned according to the weighting function

$$w_{\text{int}}(v_m, v_n) = \max_{(v_i, v_j)\in\mathcal{E}_f} w_f(v_i, v_j)$$

for $\forall(v_m, v_n) \in \mathcal{E}_{\text{int}}$, which implies that all edges in $\mathcal{G}_{\text{int}}$ are assigned the maximum weight of the edges in $\mathcal{G}_f$.

[1]The matrix $\mathbf{A}_{\mathcal{G}_{\text{int}}}$ and $\mathbf{A}_{\mathcal{G}_f}$ are assumed to have distinct eigenvalues, which is not a strong assumption as indicated in [10], since small perturbations on the entries of $\mathbf{A}_{\mathcal{G}_{\text{int}}}$ and $\mathbf{A}_{\mathcal{G}_f}$ will not affect the result of matching.

[2]Different from the classical definitions of child and parent in directed graphs, the roles of nodes in this paper are determined by their depths from the selected root in the tree. For instance, given a pair of nodes $(u, v) \in \mathcal{E}$, if $\text{depth}(u) = \text{depth}(v) - 1$, $u$ and $v$ are called the parent and child node, respectively.

### *B. Initial Tree Selection*

Since the desired $\mathcal{G}_f$ is a spanning tree, to facilitate the graph reconfiguration, an initial tree $\mathcal{G}^t_{\text{int}} \subset \mathcal{G}_{\text{int}}$ is built according to $\mathcal{G}_f$ based on the permutation $\Phi$ obtained from (4). The process starts from the node $v^r_i \in \mathcal{V}_i$ in $\mathcal{G}_{\text{int}}$ that maps onto the root node $v^r_f \in \mathcal{V}_f$ in $\mathcal{G}_f$ (i.e., $v^r_i = \Phi^{-1}(v^r_f)$), and $\mathcal{G}^t_{\text{int}}$ is grown in breadth-first fashion. The root node $v^r_f$ first checks if it has children $v^c_f \in \mathcal{V}_f$ in $\mathcal{G}_f$, and $v^r_i$ checks if it has neighbor nodes $v^n_i \in \mathcal{V}_{\text{int}}$ in $\mathcal{G}_{\text{int}}$. If $v^r_f$ and $v^r_i$ have a child $v^c_f$ and a neighbor node $v^n_i$, respectively, $v^r_i$ will first select $v^n_i$ with $\Phi(v^n_i) = v^c_f$ to form an edge $(v^r_i, v^n_i) \in \mathcal{E}_{\text{int}}$ in $\mathcal{G}^t_{\text{int}}$ which corresponds to the edge $(v^r_f, v^c_f) \in \mathcal{E}_f$. If a $v^n_i$ that satisfies $\Phi(v^n_i) = v^c_f$ does not exist, one of the unassigned neighbors $v^n_i$ will be assigned to form the edge $(v^r_i, v^n_i) \in \mathcal{E}_{\text{int}}$ in $\mathcal{G}^t_{\text{int}}$ which corresponds to the edge $(v^r_f, v^c_f) \in \mathcal{E}_f$ based on

$$\min_{\left(v^c_f, v^n_i\right)} d_{\mathcal{G}_f}\left(v^c_f, \Phi\left(v^n_i\right)\right) \tag{5}$$

where $d_{\mathcal{G}_f}(u, v)$ is a function of the distance in terms of hops between nodes $u, v \in \mathcal{V}_f$ in $\mathcal{G}_f$. The distance $d_{\mathcal{G}_f}$ between two nodes in $\mathcal{G}_f$ can be found by applying the breadth-first search algorithm [33]. The algorithm will map the neighbors $v^n_i$ to all of the children of the root $v^c_f$ in the same way, and then will proceed to map the nodes at consecutively deeper levels of the tree. If it is no longer possible to build edges for $\mathcal{G}^t_{\text{int}}$ and there still exists a remaining node $v_i \in \mathcal{V}_{\text{int}}$ in $\mathcal{G}_{\text{int}}$ that is disconnected from the existing $\mathcal{G}^t_{\text{int}}$, $v_i$ will be connected to one of the nodes $v^t_i \in \mathcal{V}_{\text{int}}$ in the existing $\mathcal{G}^t_{\text{int}}$ by forming the edge $(v_i, v^t_i) \in \mathcal{E}_{\text{int}}$ in $\mathcal{G}_{\text{int}}$ based on

$$\min_{\left(v_i, v^t_i\right) \in \mathcal{E}_{\text{int}}} d_{\mathcal{G}_f}\left(\Phi(v_i), \Phi\left(v^t_i\right)\right). \tag{6}$$

The pseudocode of the Initial Tree Selection Algorithm is shown in Algorithms 1 and 2 based on (5) and (6), respectively.

**Algorithm 1** Initial Tree Selection Algorithm (Part I)

**procedure** Input: $(\mathcal{G}_{\text{int}}; \mathcal{G}_f; \Phi)$; Output: $\mathcal{G}^t_{\text{int}} = (\mathcal{V}^t_{\text{int}}, \mathcal{E}^t_{\text{int}})$;
  $\mathcal{V}^t_{\text{int}} = \emptyset$; $\mathcal{E}^t_{\text{int}} = \emptyset$;
  DesiredTreeRootNode = GetRootNode($\mathcal{G}_f$);
  InitialGraphRootNode = $\Phi^{-1}$(DesiredTreeRootNode);
  DesiredTreeQ = DesiredTreeRootNode; InitialGraphQ = InitialGraphRootNode;
  MarkNode(DesiredTreeRootNode); MarkNode(InitialGraphRootNode);
  $\mathcal{V}^t_{\text{int}}$ = InitialGraphRootNode;
  **while** (InitialGraphQ $\neq \emptyset$) && (DesiredTreeQ $\neq \emptyset$) **do**
    DesiredTreeQTemp = $\emptyset$; InitialGraphQTemp = $\emptyset$;
    QueueLength = GetQueueLength(DesiredTreeQ);
    **for** $i = 1$ : QueueLength **do**
      ChildNodes = GetUnmarkedChildNodes(DesiredTreeQ[$i$], $\mathcal{G}_f$);
      NeighborNodes = GetUnmarkedNeighborNodes (InitialGraphQ[$i$], $\mathcal{G}_{\text{int}}$);
      **while** (ChildNodes $\neq \emptyset$) && (NeighborNodes $\neq \emptyset$) **do**
        MinimumDistance = $\infty$;
        **for all** $u \in$ ChildNodes **do**
          **for all** $v \in$ NeighborNodes **do**
            Distance = GetDistance($u, \Phi[v], \mathcal{G}_f$);
            **if** Distance < MinimumDistance **then**
              MinimumDistance = Distance;
              BestPair = $[u, v]$;
            **end if**
          **end for**
        **end for**
        $[u, v]$ = BestPair;
        **for all** $v_i \in \mathcal{V}_{\text{int}}$ **do**
          **if** $\Phi[v_i] = u$ **then**
            $\Phi[v_i] = \Phi_{\mathcal{V}_{\text{int}} \to \mathcal{V}_f}[v]$;
          **end if**
        **end for**
        $\Phi[v] = u$;
        Union($\mathcal{V}^t_{\text{int}}, v$); Union($\mathcal{E}^t_{\text{int}}$, (InitialGraphQ[$i$], $v$));
        MarkNode($u$); MarkNode($v$);
        EnqueueNode(DesiredTreeQTemp, $u$); EnqueueNode(InitialGraphQTemp, $v$);
        DesiredTreeChildNodes =(DesiredTreeChildNodes $\setminus u$);
        InitialGraphNeighborNodes = (InitialGraphNeighborNodes$\setminus v$);
      **end while**
    **end for**
    DesiredTreeQ = DesiredTreeQTemp; InitialGraphQ = InitialGraphQTemp;
  **end while**
**end procedure**

**Algorithm 2** Initial Tree Selection Algorithm (Part II)

**procedure** Input: $(\mathcal{G}_{\text{int}}; \mathcal{G}_f; )$; Output: $\mathcal{G}^t_{\text{int}} = (\mathcal{V}^t_{\text{int}}, \mathcal{E}^t_{\text{int}})$;
  UnMarkedNode = GetUnMarkedNode($\mathcal{V}_{\text{int}}$);
  **while** UnMarkedNode $\neq \emptyset$ **do**
    MinimumDistance = $\infty$;
    **for all** $u \in$ UnMarkedNode **do**
      MarkedNeighborNodes =GetMarkedNeighborNodes($u, \mathcal{G}_{\text{int}}$);
      **for all** $v \in$ MarkedNeighborNodes **do**
        Distance = GetDistance($\Phi(u), \Phi(v), \mathcal{G}_f$);
        **if** Distance < MinimumDistance **then**
          MinimumDistance = Distance;
          BestPair = $[u, v]$;
        **end if**
      **end for**
    **end for**
    $[u, v]$ = BestPair;
    Union($\mathcal{V}^t_{\text{int}}, u$); Union($\mathcal{E}^t_{\text{int}}$, $(u, v)$);
    MarkNode($u$);
    UnMarkedNode = (UnMarkedNode $\setminus u$);
  **end while**
**end procedure**

### *C. Example*

An example is provided to illustrate the algorithm described in the previous sections. Consider an initial graph $\mathcal{G}_{\text{int}}$ and the desired graph $\mathcal{G}_f$ in Fig. 1, respectively, where $\mathcal{G}_{\text{int}}$ is not a supergraph of $\mathcal{G}_f$. The nodes in $\mathcal{G}_{\text{int}}$ and $\mathcal{G}_f$ are labeled initially in Fig. 2. After applying the weighted graph-matching algorithm described in Section III-A, an initial mapping between $\mathcal{G}_{\text{int}}$ and $\mathcal{G}_f$ is generated, as shown in Fig. 2(a), where the label $a(b)$ indicates that the node $a$ in $\mathcal{G}_{\text{int}}$ is mapped to the node $b$ in $\mathcal{G}_f$.

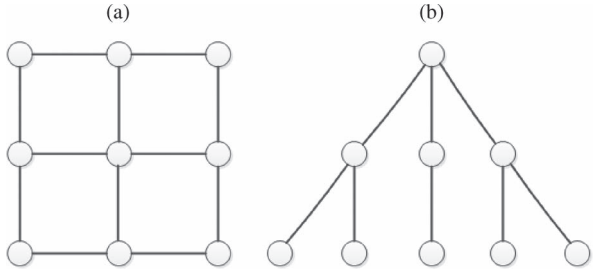


Fig. 1. (a) Initial graph and (b) the desired graph.

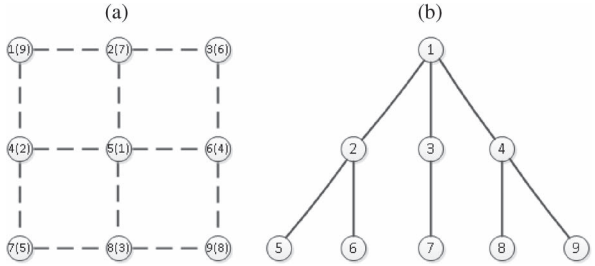


Fig. 2. (a) Initial graph and (b) the desired graph with labels. In (a), the label $x(y)$ indicates that the node $x$ in (a) is mapped to the node $y$ in (b), which is determined by the weighted graph-matching algorithm in (4). The dashed lines are the edges in (a), which will be decided later to be kept or not when building the tree $\mathcal{G}_{\rm int}^t$ according to (b).

The dashed lines connecting two nodes imply the edges in $\mathcal{G}_{\rm int}$, and will be decided later to be kept or not when building the tree $\mathcal{G}_{\rm int}^t$ according to $\mathcal{G}_f$.

Consider the root node $v_f^1$ in $\mathcal{G}_f$ and the corresponding node $v_i^5 = \Phi^{-1}(v_f^1)$ in $\mathcal{G}_{\rm int}$. Since $v_f^1$ has three child nodes, $v_f^2$, $v_f^3$, and $v_f^4$ in Fig. 2(b), and node $v_i^5$ has three neighbor nodes $v_i^4$, $v_i^6$, and $v_i^8$ in $\mathcal{G}_{\rm int}$, whose mappings are the child of $v_f^1$ in 2(a), the edges are built in the tree $\mathcal{G}_{\rm int}^t$ by connecting $v_i^5$ to its neighbor nodes $v_i^4$, $v_i^6$, and $v_i^8$, as shown in Fig. 3(a), where the newly formed edges in $\mathcal{G}_{\rm int}^t$ are indicated as solid lines. After all nodes with depth one in $\mathcal{G}_f$ (i.e., $v_f^2$, $v_f^3$, and $v_f^4$) are mapped in $\mathcal{G}_{\rm int}$, the algorithm proceeds to the nodes with depth two in $\mathcal{G}_f$. Following a similar procedure, the edge $(v_i^4, v_i^7)$ in $\mathcal{G}_{\rm int}$ can be added to $\mathcal{G}_{\rm int}^t$, since its corresponding nodes $v_f^2$ and $v_f^5$ are also connected in $\mathcal{G}_f$. Note that $v_i^4$ and $v_i^1$ are connected in $\mathcal{G}_{\rm int}$ while their corresponding nodes $v_f^2$ and $v_f^9$ are not immediate neighbors in $\mathcal{G}_f$. According to (5), the original mapping of $v_i^1 \to v_f^9$ is replaced by $v_i^1 \to v_f^6$ to form the desired edge $(v_f^2, v_f^6)$ in $\mathcal{G}_f$, as shown in Fig. 3(b). Following a similar procedure, the edges $(v_i^6, v_i^9)$ and $(v_i^6, v_i^3)$ can be added to $\mathcal{G}_{\rm int}^t$, since they correspond to the edges $(v_f^4, v_f^8)$ and $(v_f^4, v_f^9)$ in $\mathcal{G}_f$, as shown in Fig. 3(c). Finally, the tree $\mathcal{G}_{\rm int}^t$ is built as shown in Fig. 3(d), where most edges in $\mathcal{G}_f$ are preserved in $\mathcal{G}_{\rm int}$.

### D. Complexity

The worst-case complexity for the initial tree selection algorithm occurs when $\mathcal{G}_f$ and $\mathcal{G}_{\rm int}$ have depth one (i.e., star graphs), which indicates that all nodes except the root in $\mathcal{G}_f$ (or the mapped root node in $\mathcal{G}_{\rm int}$) are the children of the root (or the mapped root). To match each edge in $\mathcal{G}_f$ to an edge attached to the mapped root in $\mathcal{G}_{\rm int}$, the algorithm compares all of the

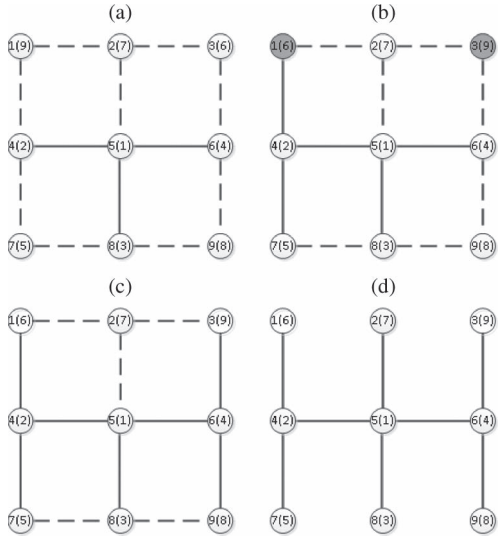


Fig. 3. Process of building a tree $\mathcal{G}_{\rm int}^t$ in $\mathcal{G}_{\rm int}$ according to $\mathcal{G}_f$. In (a), the edges {(5,4),(5,6),(5,8)} in $\mathcal{G}_{\rm int}$ are preserved to form the tree $\mathcal{G}_{\rm int}^t$, which are indicated by solid lines. In (b), the edges {(4,7),(4,1)} in $\mathcal{G}_{\rm int}$ are added to tree $\mathcal{G}_{\rm int}^t$, where the original mappings of nodes 1 and 3 (i.e., shaded nodes) are exchanged for desired neighbors. In (c), the edges {(6,9),(6,3)} in $\mathcal{G}_{\rm int}$ are added to tree $\mathcal{G}_{\rm int}^t$. In (d), the tree $\mathcal{G}_{\rm int}^t$ is finally formed by connecting nodes 5 and 2.

edges in $\mathcal{G}_f$ and $\mathcal{G}_{\rm int}$ and decides which pair of edges should be formed based on (5). Note that the star graph with $N$ nodes only has $N-1$ edges. Since the number of nodes connecting to the root in $\mathcal{G}_f$ and the mapped root in $\mathcal{G}_{\rm int}$ are $N-1$, the number of edges to be considered for matching in each topology is equal to $N-1$, which indicates that the star graphs are the worst cases for mapping. Hence, the complexity of the initial tree selection algorithm is $O(N^3)$ from

$$T(N) = \sum_{i=1}^{N-1} (N-1)^2 < \sum_{i=1}^{N-1} N^2 < N^3$$

which is also the complexity of the graph-matching step from the Eigen-decomposition.

## IV. Control Design

After the node mapping is determined in Section III, a decentralized control strategy is developed in this section to physically move all nodes to achieve the desired $\mathcal{G}_f$ with the preservation of network connectivity and collision avoidance among agents.

### A. Information Flow

When $\mathcal{G}_f$ is not isomorphic to a subgraph of $\mathcal{G}_{\rm int}$, there is no node mapping where every edge in $\mathcal{G}_f$ is preserved in $\mathcal{G}_{\rm int}$. After applying the node-mapping strategy, in most scenarios, most of the nodes in $\mathcal{G}_{\rm int}$ are mapped in such a way that their neighbors in $\mathcal{G}_{\rm int}$ are also neighbors in $\mathcal{G}_f$. However, there will be some edges in $\mathcal{G}_f$ that do not have a corresponding edge in $\mathcal{G}_{\rm int}$, that is, there exists at least one edge $(v_i, v_j) \in \mathcal{E}_f$ for which $(\Phi^{-1}(v_i), \Phi^{-1}(v_j)) \notin \mathcal{E}_{\rm int}$. For such nodes, it is required that $\Phi^{-1}(v_i)$ and $\Phi^{-1}(v_j)$ are able to communicate through

intermediate nodes, which will allow them to move to achieve the desired graph. After the initial tree selection algorithm finishes, the nodes in $\mathcal{G}_{\text{int}}^t$ can be labeled with a prefix labeling as in [17]. Then, the prefix routing can be used as in [17] to determine a path between $\Phi^{-1}(v_i)$ and $\Phi^{-1}(v_j)$ such that these nodes can move along the specified path to form the desired edge, while preserving network connectivity. An information flow $I_{ij}$ is introduced along the nodes in the specified route that enables communication between $\Phi^{-1}(v_i)$ and $\Phi^{-1}(v_j)$.

The information flow $I_{ij}$ can be realized by a series of nodes forming a path connecting $\Phi^{-1}(v_i)$ and $\Phi^{-1}(v_j)$. If the length of $I_{ij}$ is two, which indicates that $\Phi^{-1}(v_i)$ and $\Phi^{-1}(v_j)$ are connected by a mutual neighbor, the connectivity of $I_{ij}$ can be ensured by maintaining the connectivity of $\Phi^{-1}(v_i)$ and $\Phi^{-1}(v_j)$ with the mutual neighbor. If the length of $I_{ij}$ is greater than two, this indicates that node $\Phi^{-1}(v_i)$ and $\Phi^{-1}(v_j)$ are connected through more than one intermediate node. The connectivity between node $\Phi^{-1}(v_i)$ and $\Phi^{-1}(v_j)$ is not guaranteed by just maintaining the connection with its immediate neighbors, since the intermediate nodes have the potential to break the existing edge between themselves, resulting in the partition of $\Phi^{-1}(v_i)$ and $\Phi^{-1}(v_j)$. Therefore, the following development is based on the assumption that the length of $I_{ij}$ is, at most, two, which is not restrictive in the sense that an $I_{ij}$ with a path length greater than two can be partitioned into several connected partial paths (i.e., $I_{ik_1}, I_{k_1k_2}, \cdots, I_{k_nj}$ with $k_1, \cdots, k_n$ denoting the intermediate nodes of $I_{ij}$) with the length of each section being, at most, two. The node $\Phi^{-1}(v_i)$ can move in a step-by-step fashion by first approaching node $k_1$, then node $k_2, \cdots, k_n$, until achieving its destination $\Phi^{-1}(v_j)$.

An information flow $I_{ij}$ can be realized by several different paths, where the interest is not only maintaining the information flow $I_{ij}$, but also finding a short path to connect $\Phi^{-1}(v_i)$ and $\Phi^{-1}(v_j)$. The mutual node is called the *relay node*, since it is used to pass information between $\Phi^{-1}(v_i)$ and $\Phi^{-1}(v_j)$. To indicate the freedom of motion that each agent can take without disconnecting the communication link, inspired by the work of [29] and [34], a locally measurable edge robustness term $\delta_{mn}$ is defined as

$$\delta_{mn} = \frac{1}{2}(R_c - d_{mn}) \tag{7}$$

for any two immediate nodes $v_m$ and $v_n$ in the graph $\mathcal{G}$ (i.e., $(v_m, v_n) \in \mathcal{E}$). The edge robustness $\delta_{mn}$ is used to measure the robustness of the edge $(v_m, v_n)$, since $v_m$ and $v_n$ will remain connected with each other, unless both of them are displaced by a distance of $\delta_{mn}$. Therefore, a larger $\delta_{mn}$ indicates more freedom of motion. Due to node motion, some nodes may enter the communication zone of $\Phi^{-1}(v_i)$ and $\Phi^{-1}(v_j)$ at some time instant for an information flow $I_{ij}$, resulting in multiple options for the relay node. Using (7), the length of the two-edge path $l_{ij}$ is represented as $l_{ij} = d_{ir} + d_{rj} = 2R_c - 2(\delta_{ir} + \delta_{rj})$, where $\delta_{ir}$ and $\delta_{rj}$ are the robustness of each communication link $(\Phi^{-1}(v_i), v_r)$ and $(v_r, \Phi^{-1}(v_j))$ computed from (7), respectively. Finding the shortest path for $I_{ij}$ (i.e., minimizing $l_{ij}$) is equal to maximizing the addition of $\delta_{ir}$ and $\delta_{rj}$, since $R_c$ is a constant). Path robustness is defined as $\Delta_{I_{ij}} = \delta_{ir} + \delta_{rj}$, and the goal is to maximize the path robustness. Based on the previous discussion, a relay node is determined by

$$v_r = \arg \max_{v_r \in \mathcal{N}_i \cap \mathcal{N}_j} \Delta_{I_{ij}} \tag{8}$$

where the maximum taken over the intersection of communication neighbors $\mathcal{N}_i \cap \mathcal{N}_j$ aims to find a node providing the shortest path connecting $\Phi^{-1}(v_i)$ and $\Phi^{-1}(v_j)$.

To illustrate the proposed information flow, consider the example given in Section III-C. Nodes $v_f^3$ and $v_f^7$ are required to be neighbors in $\mathcal{G}_f$, while their mappings $v_i^2 = \Phi^{-1}(v_f^7)$ and $v_i^8 = \Phi^{-1}(v_f^3)$ in $\mathcal{G}_{\text{int}}$ are not immediate neighbors. The information flow $I_{ij}$ in this case is a series of paths connecting $v_i^2$ and $v_i^8$ (e.g., through the path $(v_i^2, v_i^5)$, $(v_i^5, v_i^8)$ or other longer path). A short path can be identified by choosing a relay node to maximize the path robustness in (8).

### *B. Navigation Function-Based Control Scheme*

A navigation function-based controller is developed to ensure the connectivity of the required communication links during formation reconfiguration. Consider a decentralized navigation function candidate $\varphi_i : \mathcal{F} \to [0, 1]$ for $v_i$ as

$$\varphi_i = \frac{\gamma_i}{(\gamma_i^\alpha + \beta_i)^{\frac{1}{\alpha}}} \tag{9}$$

where $\alpha \in \mathbb{R}^+$ is a tuning parameter, $\gamma_i : \mathbb{R}^2 \to \mathbb{R}^+$ is the goal function, and $\beta_i : \mathbb{R}^2 \to [0, 1]$ is a constraint function.

The goal function $\gamma_i$ in (9) drives the system to a desired configuration, specified in terms of the desired relative pose with respect to the information neighbor $v_j \in \mathcal{N}_i^f$. The goal function $\gamma$ is designed as

$$\gamma_i = \sum_{v_j \in \mathcal{N}_i^f} \|q_i - q_j - c_{ij}\|^2. \tag{10}$$

The gradient and Hessian matrix of $\gamma_i$ are given as

$$\nabla_{q_i} \gamma_i = \sum_{v_j \in \mathcal{N}_i^f} 2(q_i - q_j - c_{ij}) \tag{11}$$

and

$$\nabla_{q_i}^2 \gamma_i = 2I_2 \zeta_i \tag{12}$$

where $I_2$ is the identity matrix in $\mathbb{R}^{2\times 2}$, and $\zeta_i \in \mathbb{R}^+$ denote the number of information neighbors in the set $\mathcal{N}_i^f$. Since the Hessian matrix of $\gamma_i$ in (12) is always positive definite, the goal function (10) has a unique minimum, and the minimum is reached only when $\nabla_{q_i} \gamma_i = 0$, which implies that $q_i$ and $q_j$ achieve the desired relative pose from (11).

The constraint function $\beta_i$ in (9) is designed for $v_i$ as

$$\beta_i = B_{i0} \prod_{v_j \in \mathcal{N}_i^f} b_{ij}^r \prod_{v_k \in \mathcal{N}_i} B_{ik}. \tag{13}$$

In (13), $b_{ij}^r \triangleq b(q_i, q_r) : \mathbb{R}^2 \to [0, 1]$ ensures connectivity of an information flow $I_{ij}$ (i.e., guarantees that the relay node $v_r$ will

always be connected to $v_i$) and is designed as

$$b_{ij}^r = \begin{cases} 1 & d_{ir} \le R_c - \delta_2 \\ -\frac{1}{\delta_2^2}(d_{ir} + 2\delta_2 - R_c)^2 & \\ +\frac{2}{\delta_2}(d_{ir} + 2\delta_2 - R_c) & R_c - \delta_2 < d_{ir} < R_c \\ 0 & d_{ir} \ge R_c. \end{cases} \tag{14}$$

Node $v_i$ is aware of $\delta_{rj}$ and $\mathcal{N}_j$ in (8) through communication with $v_j$. Thus, $v_r$ can be determined locally from (8). Also, in (13), $B_{ik} \triangleq B(q_i, q_k) : \mathbb{R}^2 \to [0, 1]$, for point $v_k \in \mathcal{N}_i$, ensures that $v_i$ is repulsed from all nodes located within its sensing zone to prevent a collision, and is designed as

$$B_{ik} = \begin{cases} -\frac{1}{\delta_1^2} d_{ik}^2 + \frac{2}{\delta_1} d_{ik} & d_{ik} < \delta_1 \\ 1 & d_{ik} \ge \delta_1. \end{cases} \tag{15}$$

Similarly, the function $B_{i0}$ in (13) is used to model the potential collision of $v_i$ with the workspace boundary, where the positive scalar $B_{i0} \in \mathbb{R}$ is designed similar to $B_{ik}$ with the replacement of $d_{ik}$ by $d_{i0}$, where $d_{i0} \in \mathbb{R}^+$ is the relative distance of $v_i$ to the workspace boundary defined as $d_{i0} = R - \|q_i\|$.

Based on the definition of the navigation function candidate, a decentralized controller for each node is designed as

$$u_i = -K_i \nabla_{q_i} \varphi_i \tag{16}$$

where $K_i$ is a positive gain, and $\nabla_{q_i}\varphi_i$ is the gradient of $\varphi_i$ with respect to $q_i$, given as

$$\nabla_{q_i}\varphi_i = \frac{\alpha\beta_i \nabla_{q_i}\gamma_i - \gamma_i \nabla_{q_i}\beta_i}{\alpha\left(\gamma_i^\alpha + \beta_i\right)^{\frac{1}{\alpha}+1}}. \tag{17}$$

In (14) and (15), $b_{ij}^r$ and $B_{ik}$ are designed to be continuous and differentiable functions in $(0, R_c)$, with $b_{ij}^r$ achieving the minimum when the communication link $(v_i, v_r)$ is about to be broken (e.g., $d_{ir} = R_c$) and $B_{ik}$ achieves the minimum when $v_i$ and $v_k$ are about to collide. The constraint function only takes effect whenever $v_i$ has the potential to break an existing communication link or collide with other nodes. The gradient of $b_{ij}^r$ and $B_{ik}$ are the zero vector in the free motion region, (i.e., the interval of $(\delta_1, R_c - \delta_2)$), which indicates that $v_i$ is only driven by its goal function in (10) to form the desired relative pose with $v_j \in \mathcal{N}_i^f$ from (16) and (17). If $v_i$ dynamically builds new communication links or breaks existing links to the agents within the free motion region, the controller is still continuous from (17), since $\nabla_{q_i}\beta_i = 0$ and $\beta_i = 1$ in the free motion region. In contrast with the discontinuity introduced in the switching topology in current literature (cf., [35]), this highlighted feature enables a smooth transition between $v_i$ and other connected nodes.

### C. Connectivity and Convergence Analysis

The previous development indicates that $\mathcal{G}$ is connected if the information flow $I_{ij}$ is maintained in $\mathcal{G}$. The following proof indicates that the controller in (16) guarantees connectivity of the information flow $I_{ij}$ in $\mathcal{G}$.

*Proposition 1:* For any information flow $I_{ij}$ with $v_r$ as the relay node, the controller in (16) guarantees that $I_{ij}$ is maintained and, hence, $v_i$ and $v_j$ are connected in a communication path in $\mathcal{G}$.

*Proof:* An information flow $I_{ij}$ is realized in the communication graph $\mathcal{G}$ by a path from $v_i$ to $v_j$ through a mutual node $v_r$. From the definition of a relay node, $v_r \in \mathcal{N}_i \cap \mathcal{N}_j$, which means $v_r$ is located in the communication zone of $v_i$ and $v_j$. To show that the edge $(v_i, v_r)$ is maintained under the control law (16), consider $v_i$ located at a point $q_0 \in \mathcal{F}$ that causes $b_{ij}^r = 0$, which indicates that $v_i$ is about to disconnect with $v_r$. Since $b_{ij}^r = 0$, $\beta_i = 0$ from (13), and the navigation function achieves its maximum value from (9). Since $\varphi_i$ is maximized at $q_0$, no open set of initial conditions can be attracted to $q_0$ under the negated gradient control law designed in (16). Therefore, the communication link between node $v_i$ and $v_r$ is maintained by the controller in (16). Following the same procedure, the edge $(v_r, v_j)$ can be maintained by a similar control applied to $v_j$. Due to the motion of the nodes, some other node $v_k$ may provide a shorter path connecting $v_i$ and $v_j$ than node $v_r$ at some time instant. When this occurs, it is reasonable to create a new path from $v_i$ to $v_j$ through node $v_k$ to maintain the information flow $I_{ij}$. The relay node $v_k$ can be determined according to (8). Following the aforementioned analysis, the connectivity of the new path can also be guaranteed. ■

### D. Convergence Analysis

Our previous work in [14] proves that the proposed $\varphi_i$ in (9) is a qualified navigation function, which guarantees convergence of the system to the desired configuration. From [14], the controller in (16) ensures that almost all initial conditions are either brought to a saddle point or to the unique minimum $q_{di}$ on a compact connected manifold with boundary, as long as the tuning parameter $\alpha$ in (9) is selected that $\alpha > \max\{1, \Gamma(\varepsilon)\}$, where $\Gamma(\varepsilon)$ is a lower bound developed in [14] to ensure a qualified navigation function. The following development uses Rantzer's Dual Lyapunov Theorem [31] to show that the undesired critical points (i.e., saddle points) all measure zero, and the system can only converge to the unique minimum $q_{di}$. For the bounded workspace in this paper, a variation of Rantzer's Dual Lyapunov Theorem is stated as [36]:

*Theorem 1:* Suppose $x^* = 0 \in S$ where $S$ is an open, positively invariant, bounded subset of $\mathbb{R}^n$, which is a stable equilibrium point for $\dot{x}(t) = f(x(t))$, where $f \in C^1(S, \mathbb{R}^n)$, $f(0) = 0$. Furthermore, suppose there exists a function $\rho \in C^1(S - \{0\}, \mathbb{R})$ such that $\rho(x)f(x)/\|x\|$ is integrable on $\{x \in S : \|x\| \ge 1\}$ and

$$[\nabla \cdot (f\rho)] > 0 \text{ for almost all } x \in S. \tag{18}$$

Then, for almost all initial states $x(0) \in S$, the trajectory $x(t)$ exists for $t \in [0, \infty)$ and tends to zero as $t \to \infty$.[3]

Theorem 1 requires $x^* = 0 \in S$ to be a stable equilibrium point. The goal function evaluated at the desired point is $\gamma_i|_{q_{di}} = 0$ from (10), and $\nabla_{q_i}\gamma_i|_{q_{di}} = 0$ from (11), which can be used to conclude that $\nabla_{q_i}\varphi_i|_{q_{di}} = 0$ from (17). Thus, the desired point $q_{di}$ in the workspace $\mathcal{F}$ is a critical point of $\varphi_i$. Using the facts that $\gamma_i|_{q_{di}} = 0$ and $\nabla_{q_i}\gamma_i|_{q_{di}} = 0$ and the

[3]For a function $f : \mathbb{R}^n \to \mathbb{R}^n$, the notation of divergence is defined as $\nabla \cdot f = (\partial f_1/\partial x_1) + \cdots + (\partial f_n/\partial x_n)$.

Hessian of $\gamma_i$ is $\nabla^2_{q_i}\gamma_i = 2\zeta_i I_2$ from (12), the Hessian of $\varphi_i$ evaluated at $q_{di}$ is given by $\nabla^2_{q_i}\varphi_i|_{q_{di}} = 2\beta_i^{-(1/\alpha)} I_2 \zeta_i$. Since the constraint function $\beta_i > 0$ at the desired configuration by Assumption 2, and $\zeta_i$ is a positive number, the Hessian of $\varphi_i$ evaluated at $q_{di}$ is positive definite. Hence, the navigation function $\varphi_i$ is minimized at $q_{di}$.

*Proposition 2:* The closed-loop kinematics of system (1) with the controller in (16) are given by $\dot{\mathbf{q}} = f(\mathbf{q})$, where $\mathbf{q}$ denotes the stacked states of each node as $\mathbf{q} = [q_1^T \cdots q_N^T]^T$ and $f(\mathbf{q}) = [f_1^T \cdots f_N^T]$ with $f_i^T = -K_i \nabla_{q_i}\varphi_i$ for $\forall i \in \mathcal{N}$. Consider the system $\dot{\mathbf{q}} = f(\mathbf{q})$ for $\forall i \in \mathcal{N}$ and a density function as $\rho = -\varphi$, where $\varphi = \sum_{i=1}^{N} \varphi_i$ in Theorem 1. The undesired critical points are sets of measure zero from Theorem 1, provided $\alpha > \max\{1, \Gamma(\varepsilon), \varepsilon'\}$ at any saddle points, where $\alpha$ is a parameter in the navigation function (9).

*Proof:* The function $\rho$ is defined for all points in the workspace other than the desired equilibrium $q_{di}$, and each $\varphi_i$ is $C^2$ and takes a value in [0,1]. Thus, the function $\varphi$ and its gradient are bounded functions in the workspace, which indicates that the integrability condition in Theorem 1 is fulfilled. From the divergence criterion, $\nabla \cdot (f\rho) = (\nabla\rho)^T f + \rho \nabla \cdot (f)$, and from the definition of a critical point, $\nabla_{q_i}\varphi_i = 0$. Hence, $f_i^T = -K_i \nabla_{q_i}\varphi_i = 0$ for $\forall i \in \mathcal{N}$, which indicates that $f = 0$, and $\nabla \cdot (f\rho)$ can be simplified as

$$\nabla \cdot (f\rho) = \varphi \sum_{i=1}^{N} K_i \left( \frac{\partial^2 \varphi_i}{\partial x_i^2} + \frac{\partial^2 \varphi_i}{\partial y_i^2} \right). \tag{19}$$

Since $\varphi$ are positive at undesired critical points from (9), and $K_i$ is a positive gain, a sufficient condition for (19) to be strictly positive is $(\partial^2\varphi_i/\partial x_i^2) + (\partial^2\varphi_i/\partial y_i^2) > 0$. Using (17), $\partial^2\varphi_i/\partial x_i^2$ and $\partial^2\varphi_i/\partial y_i^2$ are computed as

$$\frac{\partial^2 \varphi_i}{\partial x_i^2} = \frac{\left( \frac{\partial \beta_i}{\partial x_i}\frac{\partial \gamma_i}{\partial x_i} + \beta_i \frac{\partial^2 \gamma_i}{\partial x_i^2} - \frac{1}{\alpha}\frac{\partial \beta_i}{\partial x_i}\frac{\partial \gamma_i}{\partial x_i} - \frac{\gamma_i}{\alpha}\frac{\partial^2 \beta_i}{\partial x_i^2} \right)}{(\gamma_i^\alpha + \beta_i)^{\frac{1}{\alpha}+1}} \tag{20}$$

$$\frac{\partial^2 \varphi_i}{\partial y_i^2} = \frac{\left( \frac{\partial \beta_i}{\partial y_i}\frac{\partial \gamma_i}{\partial y_i} + \beta \frac{\partial^2 \gamma_i}{\partial y_i^2} - \frac{1}{\alpha}\frac{\partial \beta_i}{\partial y_i}\frac{\partial \gamma_i}{\partial y_i} - \frac{\gamma_i}{\alpha}\frac{\partial^2 \beta_i}{\partial y_i^2} \right)}{(\gamma_i^\alpha + \beta_i)^{\frac{1}{\alpha}+1}}. \tag{21}$$

Observing that $\partial^2\varphi_i/\partial x_i^2$ and $\partial^2\varphi_i/\partial y_i^2$ have a similar structure, it suffices to show that $\partial^2\varphi_i/\partial x_i^2 > 0$ for $\forall i \in \mathcal{N}$, since the same results can be derived for $\partial^2\varphi_i/\partial y_i^2$. Since $\gamma_i$ and $\beta_i$ are positive from (10) and (13), and cannot be zero simultaneously from Assumption 2, the positivity of (20) can be proven by showing that the numerator on the right side of (20) is positive. Using the fact that $\partial\beta_i/\partial x_i = (\alpha\beta_i/\gamma)(\partial\gamma_i/\partial x_i)$ at a critical point, the following expression can be obtained from (20):

$$C_1\alpha^2 + C_2\alpha + C_3 > 0 \tag{22}$$

where $C_1 = (\beta_i/\gamma_i)(\partial\gamma_i/\partial x_i)^2$, $C_2 = (\beta_i/\gamma_i)((\gamma_i\partial^2\gamma_i/\partial x_i^2) - (\partial\gamma_i/\partial x_i)^2)$, and $C_3 = -(\gamma_i\partial^2\beta_i/\partial x_i^2)$. Note that $\beta_i = 0$ indicates $\varphi_i$ achieves its maximum from (9). However, since the set of initial conditions is open, and no open set of initial conditions can be attracted to the maxima of $\varphi_i$ along the negative gradient motion $-K_i\nabla_{q_i}\varphi_i$ [22], then $\beta_i \neq 0$. In addition, $\gamma_i$ is evaluated at the undesired critical points (i.e., except the $q_{di}$), so $\gamma_i \neq 0$ and $\partial\gamma_i/\partial x_i \neq 0$ from (10) and (11). To satisfy the condition in (22), two cases are considered for

$$C_1\alpha^2 + C_2\alpha + C_3 = 0. \tag{23}$$

Case 1: No solution of $\alpha$ exists for (23): Since $(\beta_i/\gamma_i)(\partial\gamma_i/\partial x_i)^2 > 0$ and $\alpha$ is a positive gain in (9), as long as $\alpha > 0$, the condition in (22) is valid in Case 1.

Case 2: Two solutions $S_1$ and $S_2$ exist in (23). In this case, the condition in (22) is satisfied as long as $\alpha > \max\{S_1, S_2, 0\}$. Combining Cases 1 and 2 indicates that if $\alpha > \max\{1, \Gamma(\varepsilon), \varepsilon'\}$, where $\varepsilon'$ is defined as $\varepsilon' = \max\{S_1, S_2, 0\}$, all saddle points measure zero, and the system will only converge to the desired configuration. ■

*Remark 1:* Since physical formation reconfiguration with minimum node movement is still an open problem, as a first attempt to these challenging problems, the developed node-mapping algorithm focuses on minimization of node movement in terms of the number of edges that an agent must traverse on the topology level, and a motion control law is then applied for formation reconfiguration with preservation of network connectivity. Note that agents that are separated by the same number of hops in the network topology may be at very different distances in the physical formation. Future work will consider including the node movement in the physical graph to the node-mapping algorithm so that the amount of node movement in physical formation reconfiguration can also be minimized.

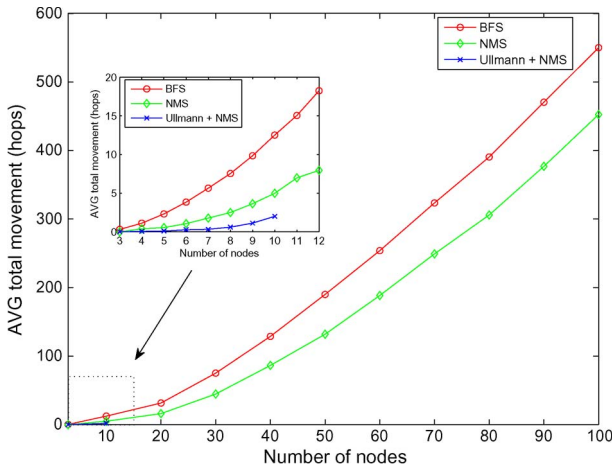


Fig. 4. Average movement (hops) required to achieve $\mathcal{G}_f$ for the NMS algorithm, the Ullmann's algorithm, and the BFS algorithm.

## V. Simulation

Simulation results are provided in this section to illustrate the performance of the node-mapping strategy (NMS) developed in Section III and the decentralized motion controller designed in Section IV. The performance of the NMS is evaluated by comparing the amount of movement (in terms of the number of edges that an agent must traverse) required to achieve $\mathcal{G}_f$ from $\mathcal{G}_{\text{int}}$ with Ullmann's algorithm in [9] and the breadth first strategy (BFS) in [15]. Specifically, 500 pairs of $\mathcal{G}_{\text{int}}$ and $\mathcal{G}_f$ are randomly generated for each network size which ranges from 3 to 100 nodes. For each pair of $\mathcal{G}_{\text{int}}$ and $\mathcal{G}_f$, the NMS,

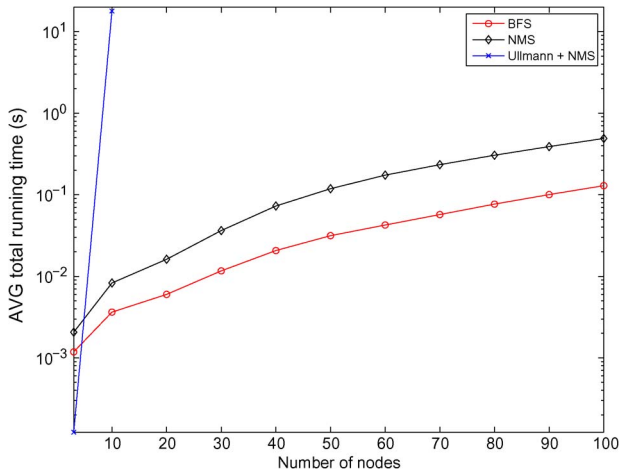


Fig. 5. Average running time to achieve $\mathcal{G}_f$ for the NMS algorithm, the Ullmann's algorithm, and the BFS algorithm.

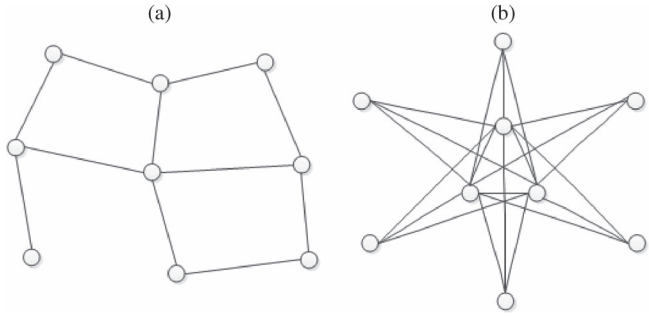


Fig. 6. (a) Initial graph and (b) the desired graph. The agents are represented as nodes and the solid lines indicate the neighborhood of agents.

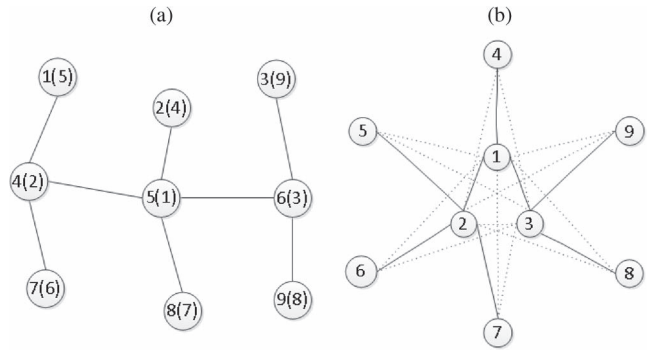


Fig. 7. Tree $\mathcal{G}_{\text{int}}^t$ is built on $\mathcal{G}_{\text{int}}$, where the label $x(y)$ on each node indicates that the node $x$ on $\mathcal{G}_{\text{int}}^t$ corresponds to the node $y$ on $\mathcal{G}_f$. A tree $\mathcal{G}_f$ is generated on the desired graph in (b), where the edges are indicated by the solid lines and all nodes are labeled.

BFS, and Ullmann's algorithm are applied to reconfigure $\mathcal{G}_{\text{int}}$ to $\mathcal{G}_f$, where the minimum amount of movement is provided by the Hungarian method in [32]. For each network size, the averaged amount of node movement from the 500 pairs of $\mathcal{G}_{\text{int}}$ and $\mathcal{G}_f$ is shown in Fig. 4, and the average running time for each algorithm is shown in Fig. 5, where the y-axis is in log scale due to the range of the results. In the NMS and BFS algorithms, a tree $\mathcal{G}_{\text{int}}^t$ is generated according to $\mathcal{G}_f$ first. It is clear that NMS significantly outperforms BFS as shown in Fig. 4, especially when the network size is large. Minimum movement is obtained by Ullmann's algorithm due to its optimal nature. However,

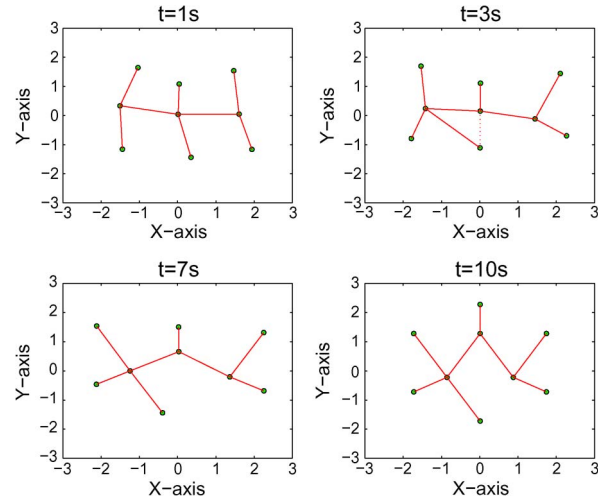


Fig. 8. Change of the network topology during network reconfiguration.

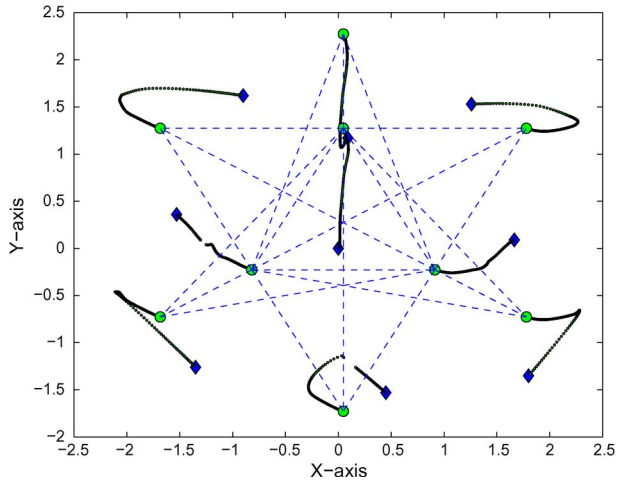


Fig. 9. Plot of node trajectories in achieving the desired formation, where the dashed line indicates the trajectory of each node, and diamonds and circles represent the initial and desired positions of nodes, respectively.

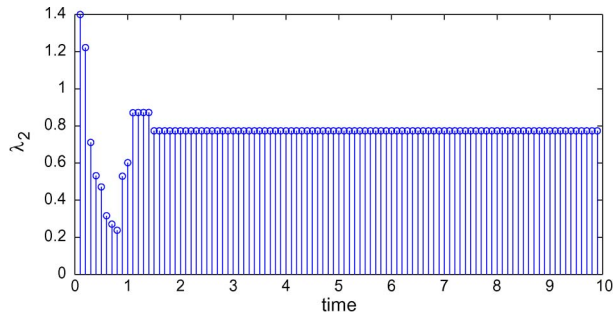


Fig. 10. Evolution of the Fiedler Value during the formation reconfiguration. A positive Fiedler Value indicates the network maintains connectivity.

Ullmann's algorithm is only applicable to isomorphic graphs, and Fig. 4 indicates that Ullmann's algorithm is valid for a network size with 10 or fewer nodes because of its exponential computational complexity. In Fig. 5, the average running time increases with respect to the network size for all algorithms. The average running time of BFS is less than NMS, since $\mathcal{G}_{\text{int}}^t$ is selected randomly in BFS while NMS runs graph matching

TABLE I
COMPARISON OF TOTAL NODE MOVEMENT FOR RANDOM ($R_1$ TO $R_{10}$) AND NODE-MAPPING STRATEGY (NMS) ASSIGNMENTS BETWEEN NODES IN INITIAL AND FINAL FORMATIONS

| | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ | $R_6$ | $R_7$ | $R_8$ | $R_9$ | $R_{10}$ | $R_{avg}$ | NMS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ANM (m) | 39.36 | 23.34 | 28.41 | 37.60 | 39.38 | 41.16 | 28.99 | 33.72 | 41.17 | 34.09 | 34.77 | 24.12 |

to improve the selection of $\mathcal{G}^t_{\text{int}}$. The average running time of Ullmann's algorithm is low when the network size is small. However, as the network size increases to seven or more nodes, the average running time of Ullmann's algorithm increases significantly and eventually makes this approach impractical due to its exponential complexity.

To demonstrate the performance of the decentralized motion controller, a network of nine identical agents is tasked with reconfiguration to a new formation. The initial graph and the desired graph are shown in Fig. 6(a) and (b), respectively, where the agents are represented as nodes and the solid lines indicate the neighborhood of agents. In [3], an attack tolerance graph is developed that can maintain a required level of network connectivity in the presence of node and link failures. In this simulation, the particular graph in [3] is taken as the desired graph in Fig. 6(b). Since a tree structure is required, we first build a tree $\mathcal{G}_f$ on the desired graph as indicated by the solid lines in Fig. 7(b). Given the tree $\mathcal{G}_f$ in Fig. 7(b), the NMS developed in Section III is applied to generate a tree $\mathcal{G}^t_{\text{int}}$ to map nodes between the initial and desired graph, as shown in Fig. 7(a), where the label $x(y)$ on each node indicates that the node $x$ on $\mathcal{G}^t_{\text{int}}$ corresponds to the node $y$ on $\mathcal{G}_f$, and the edges of $\mathcal{G}^t_{\text{int}}$ are represented by solid lines. Comparing Fig. 7(a) and (b), most edges in $\mathcal{G}_f$ are preserved in $\mathcal{G}^t_{\text{int}}$, except the edge connecting nodes 2 and 7.

The decentralized control law described in Section IV is implemented to physically move the nodes to form the desired graph. As the nodes move along the specified information flow, the network topology changes. The network topology is shown in Fig. 8 for several representative times during the reconfiguration. At the time $t = 3$ s, the dashed line in Fig. 8 indicates that node 7 preserves connectivity with node 1 to ensure the specified information flow while moving toward node 2 to build a new link with it. The evolution of node trajectories is shown in Fig. 9, where the initial and final positions of nodes are denoted by diamonds and circles, respectively. Fig. 10 is a plot of the Fiedler Value (i.e., the second smallest eigenvalue of the Laplacian matrix of the underlying time-varying graph). A positive Fiedler Value indicates that the graph remains connected [37].

The results in Table I compare the performance of the developed decentralized motion controller with either the NMS or a random mapping of the nodes in the initial formation to the roles in the desired formation. The numerical values are the total Euclidean distance traveled by all of the nodes. The results for $R_1$ to $R_{10}$ are the distances traveled for ten different initial trees $\mathcal{G}^t_{\text{int}}$ created from random node assignments. The total movement under the graph-matching-based NMS technique is 24.12, whereas the total movement with random assignments varies from 23.34 to 41.17. Thus, the benefit of using the NMS algorithm is demonstrated.

## VI. DISCUSSION AND CONCLUSION

The combined NMS and decentralized motion controller provides a novel solution to the problem of formation reconfiguration with identical agents. Simulation results show that using NMS decreases the amount of movement required in comparison to a random mapping in an example formation reconfiguration scenario. The simulations in this work included a network ranging from 3 to 100 nodes. When considering large networks (e.g., hundreds to millions of nodes), the complexity of the developed initial tree selection algorithm will approximately increase as $O(N^3)$, and the large size of the adjacency matrix may prohibit the use of standard Eigen-decomposition algorithms. However, as indicated in [38], large networks generally contain sparsity in the associated adjacency matrix. To facilitate efficient computation, various approaches have been developed to enable parallel computation by partitioning sparse matrices (e.g., [38] and [39]) or apply scalable approaches to reduce the complexity of large sparse matrices [40]. Further efforts could potentially be applied to take advantage of sparsity in the adjacency matrix to extend the current work to large networks as in [38]–[40].

## REFERENCES


[1] F. Zhang and N. Leonard, "Cooperative filters and control for cooperative exploration," *IEEE Trans. Autom. Control*, vol. 55, no. 3, pp. 650–663, Mar. 2010.

[2] P. Ogren, E. Fiorelli, and N. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment," *IEEE Trans. Autom. Control*, vol. 49, no. 8, pp. 1292–1302, Aug. 2004.

[3] A. Veremyev and V. Boginski, "Robustness and strong attack tolerance of low-diameter networks," in *Dynamics of Information Systems: Mathematical Foundations*. New York, USA: Springer, 2012, pp. 137–156.

[4] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control Syst. Mag.*, vol. 27, no. 2, pp. 71–82, Apr. 2007.

[5] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[6] Y. Kim and M. Mesbahi, "On maximizing the second smallest eigenvalue of a state-dependent graph Laplacian," *IEEE Trans. Autom. Control*, vol. 51, no. 1, pp. 116–120, Jan. 2006.

[7] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, vol. 53, no. 1, pp. 65–78, 2004.

[8] R. Dai and M. Mesbahi, "Optimal topology design for dynamic networks," in *Proc. IEEE Conf. Dec. Control*, 2011, pp. 1280–1285.

[9] J. R. Ullmann, "An algorithm for subgraph isomorphism," *J. ACM*, vol. 23, no. 1, pp. 31–42, 1976.

[10] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 5, pp. 695–703, Sep. 1988.

[11] L. S. Shapiro and J. Michael Brady, "Feature-based correspondence: An eigenvector approach," *Image Vis. Comput.*, vol. 10, no. 5, pp. 283–288, 1992.

[12] M. Carcassoni and E. R. Hancock, "Spectral correspondence for point pattern matching," *Pattern Recogn.*, vol. 36, no. 1, pp. 193–204, 2003.

[13] T. Caelli and S. Kosinov, "An eigenspace projection clustering method for inexact graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 4, pp. 515–519, Apr. 2004.

[14] Z. Kan, A. Dani, J. M. Shea, and W. E. Dixon, "Network connectivity preserving formation stabilization and obstacle avoidance via a decentralized controller," *IEEE Trans. Autom. Control*, vol. 57, no. 7, pp. 1827–1832, Jul. 2012.
[15] L. Navaravong, J. M. Shea, and W. E. Dixon, "Physical- and network-topology control for systems of mobile robots," in *Proc. Military Commun. Conf.*, Baltimore, MD, USA, Nov. 2011, pp. 1079–1084.
[16] L. Navaravong, J. M. Shea, E. L. Pasiliao, and W. E. Dixon, "Graph matching-based topology reconfiguration algorithm for systems of networked autonomous vehicles," presented at the Military Commun. Conf., San Diego, CA, USA, Nov., 2013.
[17] L. Navaravong, Z. Kan, J. M. Shea, and W. E. Dixon, "Formation reconfiguration for mobile robots with network connectivity constraints," *IEEE Netw.*, vol. 26, no. 4, pp. 18–24, Jul./Aug. 2012.
[18] M. Zavlanos, M. Egerstedt, and G. Pappas, "Graph theoretic connectivity control of mobile robot networks," *Proc. IEEE: Spec. Issue on Swarming in Nat. Eng. Syst.*, vol. 99, pp. 1525–1540, 2011.
[19] K. Do, "Bounded controllers for formation stabilization of mobile agents with limited sensing ranges," *IEEE Trans. Autom. Control*, vol. 52, no. 3, pp. 569–576, Mar. 2007.
[20] J. Huang, S. Farritor, A. Qadi, and S. Goddard, "Localization and follow-the-leader control of a heterogeneous group of mobile robots," *IEEE/ASME Trans. Mechatron.*, vol. 11, no. 2, pp. 205–215, Apr. 2006.
[21] H. Tanner and A. Kumar, "Towards decentralization of multirobot navigation functions," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 4132–4137.
[22] D. Dimarogonas and K. Johansson, "Bounded control of network connectivity in multi-agent systems," *Control Theory Appl.*, vol. 4, no. 8, pp. 1330–1338, Aug. 2010.
[23] M. Ji and M. Egerstedt, "Distributed coordination control of multiagent systems while preserving connectedness," *IEEE Trans. Robot.*, vol. 23, no. 4, pp. 693–703, 2007.
[24] M. M. Zavlanos and G. J. Pappas, "Distributed formation control with permutation symmetries," in *Proc. IEEE Conf. Decis. Control*, 2007, pp. 2894–2899.
[25] M. Turpin, N. Michael, and V. Kumar, "Trajectory planning and assignment in multirobot systems," in *Algorithmic Found. Robot. X*. New York, USA: Springer, 2013, pp. 175–190.
[26] M. Schuresko and J. Cortés, "Distributed tree rearrangements for reachability and robust connectivity," *SIAM J. Control. Optimiz.*, vol. 50, no. 5, pp. 2588–2620, 2012.
[27] H. Su, G. Chen, X. Wang, and Z. Lin, "Adaptive secondorder consensus of networked mobile agents with nonlinear dynamics," *Automatica*, vol. 47, no. 2, pp. 368–375, 2011.
[28] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. Robot. Autom.*, vol. 8, no. 5, pp. 501–518, Oct. 1992.
[29] D. Spanos and R. Murray, "Robust connectivity of networked vehicles," in *Proc. IEEE Conf. Decis. Control*, Dec. 2004, vol. 3, pp. 2893–2898.
[30] Z. Kan, A. Dani, J. Shea, and W. E. Dixon, "Information flow based connectivity maintenance of a multi-agent system during formation control," in *Proc. IEEE Conf. Decis. Control*, Orlando, FL, USA, 2011, pp. 2375–2380.
[31] A. Rantzer, "A dual to lyapunov's stability theorem," *Syst. Control Lett.*, vol. 42, no. 3, pp. 161–168, 2001.
[32] M. D. R. Burkard and S. Martello, *Assignment Problems*. Philadelphia, PA, USA: SIAM, 2009.
[33] T. Cormen, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2001.
[34] D. Spanos and R. Murray, "Motion planning with wireless network constraints," in *Proc. Amer. Control Conf.*, Jun. 2005, pp. 87–92.
[35] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Trans. Autom. Control*, vol. 52, no. 5, pp. 863–868, May 2007.
[36] D. Dimarogonas and K. Johansson, "Analysis of robot navigation schemes using Rantzer dual Lyapunov theorem," in *Proc. Amer. Control Conf.*, Jun. 2008, pp. 201–206.
[37] C. Godsil and G. Royle, *Algebraic Graph Theory*. New York, USA: Springer, 2001, ser. Graduate Texts in Mathematics.
[38] A. Pothen, H. D. Simon, and K.-P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM J. Matrix Anal. Appl.*, vol. 11, no. 3, pp. 430–452, 1990.
[39] A. Gupta, G. Karypis, and V. Kumar, "Highly scalable parallel algorithms for sparse matrix factorization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 8, no. 5, pp. 502–520, May 1997.
[40] U. Kang, B. Meeder, and C. Faloutsos, "Spectral analysis for billion-scale graphs: Discoveries and implementation," in *Advances in Knowledge Discovery and Data Mining*. New York, USA: Springer, 2011, pp. 13–25.

**Zhen Kan** received the B.S degree (Hons.) and the M.S. degree in mechanical engineering from Hefei University of Technology, Hefei, China, in 2005 and 2007, respectively, and the Ph.D. degree in mechanical and aerospace engineering at the University of Florida, Gainesville, FL, USA, in 2011.

Currently, he is a Postdoctoral Research Associate in the Research and Engineering Education Facility (REEF) at the University of Florida. His research interests are in the area of cooperative control of multiagent systems, Lyapunov-based nonlinear control, vision-based estimation and control, human-robot interaction, as well as human-assisted estimation, planning, and decision-making.

**Leenhapat Navaravong** received the B.E. degree in telecommunications engineering (Hons.) from Shinawatra International University, Pathumthani, Thailand, in 2007, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2010 and 2013, respectively.

After his graduation, he was a Postdoctoral Research Associate at the Air Force Research Laboratory (AFRL) at the Eglin Air Force Base, Eglin Air Force Base, FL, USA, and the University of Florida/Research and Engineering Education Facility (REEF), Shalimar, FL, from 2013 to 2014. His research interests are wireless communications and networking with applications to systems of autonomous vehicles.

Currently, he is a Project Manager in the Business Development Department of Thaicom PLC.

**John M. Shea** (S'92–M'99) received the B.S. (Hons.) degree in computer engineering and the M.S. and Ph.D. degrees in electrical engineering from Clemson University, Clemson, SC, USA, in 1993, 1995, and 1998, respectively.

Currently, he is an Associate Professor of Electrical and Computer Engineering at the University of Florida, Gainesville, FL, USA. Previously, he was an Assistant Professor at the University of Florida from 1999 to 2005 and a Postdoctoral Research Fellow at Clemson University in 1999. He is currently engaged in research on wireless and military communications and networked autonomous systems.

Dr. Shea received the Technical Achievement Award for contributions to military communications from the IEEE Military Communications Conference (MILCOM) in 2012. He received the Ellersick Award from the IEEE Communications Society for the Best Paper in the Unclassified Program of MILCOM in 1996 and 2013, and the Outstanding Young Alumni award from the Clemson University College of Engineering and Science in 2011. He was selected as a Finalist for the 2004 Eta Kappa Nu Outstanding Young Electrical Engineer. He has been an Editor for *IEEE Wireless Communications* magazine since 2010. He was an Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS from 2008 to 2012 and was an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY from 2002 to 2007. He was the Technical Program Chair for the Unclassified Program of MILCOM in 2010.

**Eduardo L. Pasiliao, Jr.** was born in the Philippines. He received the B.S. degree in mechanical engineering from Columbia University, New York, USA, the M.E. degree in coastal and oceanographic engineering, and the Ph.D. degree in industrial and systems engineering from the University of Florida, Gainesville, FL, USA.

Currently, he is a Senior Research Engineer with the Air Force Research Laboratory (AFRL) Munitions Directorate, Eglin Air Force Base, FL, USA, and is the Director of the AFRL Mathematical Modeling and Optimization Institute, Eglin Air Force Base, FL, USA. His research interest is in mathematical optimization with an emphasis on social and communication networks.

**Warren E. Dixon** received the Ph.D. degree in electrical and computer engineering from Clemson University, Clemson, SC, USA, in 2000.

After completing his doctoral studies, he was selected as an Eugene P. Wigner Fellow at Oak Ridge National Laboratory (ORNL), Oak Ridge, TN, USA. In 2004, he joined the University of Florida, Gainesville, FL, USA, in the Mechanical and Aerospace Engineering Department. He has published more than 300 refereed papers and several books in this area. His main research interest has been the development and applications of Lyapunov-based control techniques for uncertain nonlinear systems.

Dr. Dixon is currently a member of the U.S. Air Force Science Advisory Board and the Director of Operations for the Executive Committee of the IEEE CSS Board of Governors. Previously, he was an Associate Editor for several journals, and is currently an Associate Editor for *Automatica* and the *International Journal of Robust and Nonlinear Control*. His work has been recognized by the 2013 Fred Ellersick Award for Best Overall MILCOM Paper, 2012–2013 University of Florida College of Engineering Doctoral Dissertation Mentoring Award, 2011 American Society of Mechanical Engineers (ASME) Dynamics Systems and Control Division Outstanding Young Investigator Award, 2009 American Automatic Control Council (AACC) O. Hugo Schuck (Best Paper) Award, 2006 IEEE Robotics and Automation Society (RAS) Early Academic Career Award, a National Science Foundation CAREER Award (2006–2011), 2004 DOE Outstanding Mentor Award, and the 2001 ORNL Early Career Award for Engineering Achievement. He is an IEEE Control Systems Society (CSS) Distinguished Lecturer.