

Network Topology Identification using Supervised Pattern Recognition Neural Networks

Aniruddha Perumalla^a, Ahmet Taha Koru^b and Eric Norman Johnson

Department of Aerospace Engineering, Pennsylvania State University, University Park, PA, U.S.A.

Keywords: Multi-Agent Systems, Neural Networks, Topology Identification, Machine Learning, Counterintelligence, Countersurveillance, Autonomous Systems, Graph Theory, Linear Consensus, Pattern Recognition, Supervised Learning.

Abstract: This paper studies the network topology identification of multi-agent systems with single-integrator dynamics using supervised pattern recognition networks. We split the problem into two classes: (i) small-scale systems, and (ii) large-scale systems. In the small-scale case, we generate all connected (undirected) graphs. A finite family of vectors represent all possible initial conditions by gridding the interval 0 and 1 for each agent. The system responses for all graphs with all initial conditions are the training data for the supervised pattern recognition neural network. This network is successful in identification of the most connected node in up to nearly 99% of cases involving small-scale systems. We present the accuracy of the trained network for network topology identification with respect to grid space. Then, an algorithm predicated on the pattern recognition network, which is trained for a small-scale system, identifies the most connected node in large-scale systems. Monte Carlo simulations estimate the accuracy of the algorithm. We also present the results for these simulations, which demonstrate that the algorithm succeeds in finding the most connected node in more than 60% of the test cases.

1 INTRODUCTION


The theoretical advances in cooperative control of multi-agent systems lead swarm systems to take place in engineering applications. Autonomous transportation (Teodorovic, 2003), complex power networks (Bidram et al., 2014), water distribution (de Roo et al., 2015), multi-robot systems (Ota, 2006), and modern warfare (Scharre, 2018) are among the examples. Intelligent group behaviors emerge from the swarming of a group of individuals with poor abilities (Tan and Zheng, 2013). Therefore, swarming fulfills the demand to accomplish complex tasks.


The agents of a swarm share their sensor measurements through a communication network to cooperate. However, the network topology is not directly available in some applications, for instance the interactions between brain neurons (Valdés-Sosa et al., 2005) and gene regulatory networks in biological systems (Julius et al., 2009). Another interesting example is the case of *counter-swarm systems*. Such a sys-

tem may require the knowledge of the opponent network topology to efficiently neutralize.

Although it is a system identification application; yet, conventional methods may fail to identify some network topologies (see the discussion in subsection 2.3). Consequently, the *network topology identification* studies draw attention of researchers. The authors of (Gonçalves and Warnick, 2008) study the network topology identification of linear time-invariant systems. A node-knockout procedure identify the topology in consensus-type network (Nabi-Abdolyousefi and Mesbahi, 2012). Reference (Rahimian et al., 2013) discusses the identifiability of links and nodes of multi-agent systems under the agreement protocol. Stochastic perturbations recover the underlying topologies of noise-contaminated complex dynamical networks (Wu et al., 2015). The authors of (Sun and Dai, 2015) reformulate the network topology identification problem as a quadratic optimization problem. The constrained Lyapunov equations establish network reconstruction algorithms (van Waarde et al., 2019).

In this paper, we utilize neural networks to identify an unknown network topology. We consider the

^a  <https://orcid.org/0000-0002-2618-7032>

^b  <https://orcid.org/0000-0001-8191-2324>

single-integrator dynamics with a consensus protocol over a fixed connected (undirected) graph. We split the problem into two: (i) identification of the entire graph for small-scale systems, and (ii) identification of the most connected node for large-scale systems. For problem (i), we trained a pattern recognition neural network to identify the topology of a four-node network using simulated data of the observed state of the network over a small period of time. We propose an algorithm predicated on the neural networks trained in a small-scale setup to reveal the most connected node(s) in a large-scale system.

1.1 Motivation

Countersurveillance is the primary motivator of this algorithm. If one wishes to attack an opponent swarm of vehicles, knowledge of the swarm's network topology may significantly increase the efficiency and effectiveness of one's attack. For instance, if it is known that the topology of the swarm adheres to a spoked-hub or star/"leader-follower" network structure (visualized in Figure 1, with node 10 being the leader), focusing the attack on the leader of the swarm may be sufficient to incapacitate the swarm. In general, knowledge of the opponent network topology allows a valuable additional basis on which countersurveillance measures may weigh the importance of specific members of the opponent system and thereafter distribute their resources more effectively.

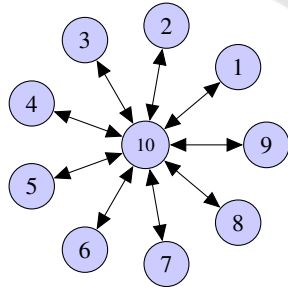


Figure 1: An example leader-follower network topology.

2 PRELIMINARIES

2.1 Notation

The notation of this paper is standard. Specifically, \mathbb{R} , \mathbb{R}^n , $\mathbb{R}^{n \times m}$, \mathbb{Z}_{++} respectively stand for the sets of all real numbers, n -dimensional real column vectors, $n \times m$ real matrices, and positive integers. The symbol " \triangleq " denotes equality by definition. We also write I_n for the $n \times n$ identity matrix, $0_{n \times m}$ for the $n \times m$ matrix

with zero entries, $\mathbf{1}_n$ for the $n \times 1$ vector of all ones, \otimes for Kronecker product. The symbol $\mathcal{P}(X)$ denotes the power set and $|X|$ denotes cardinality of a set X .

2.2 Graph Theory

Consider a fixed (i.e. time-invariant) directed or undirected graph (representing network topology) $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$ is a non-empty finite set of N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges. There is an edge rooted at node v_j and ended at v_i (i.e. $(v_j, v_i) \in \mathcal{E}$) if and only if v_i receives information from v_j . $\mathcal{A}(\mathcal{G}) = [a_{ij}] \in \{0, 1\}^{N \times N}$ denotes the adjacency matrix. We only consider boolean graphs; that is, $a_{ij} = 1 \Leftrightarrow (v_j, v_i) \in \mathcal{E}$ and $a_{ij} = 0$ otherwise. Furthermore, repeated edges and self-loops are not allowed; that is, $a_{ii} = 0$, $\forall i \in I_N$. If $(v_i, v_j) \in \mathcal{E}$, then the nodes i and j are neighbours. The set of neighbors of node v_i is denoted as $\mathcal{N}_i = \{j \mid (v_j, v_i) \in \mathcal{E}\}$. A graph with the property that $a_{ij} = a_{ji}$ is undirected. The in-degree matrix is defined by $\mathcal{D}(\mathcal{G}) = \text{diag}(d_i)$ with $d_i = \sum_{j \in \mathcal{N}_i} a_{ij}$. A directed path from node v_i to node v_j is a sequence of successive edges in the form $\{(v_i, v_p), (v_p, v_q), \dots, (v_r, v_j)\}$. The Laplacian of the graph \mathcal{G} is defined as $\mathcal{L}(\mathcal{G}) = \mathcal{D}(\mathcal{G}) - \mathcal{A}(\mathcal{G})$. A *connected graph* is an undirected graph which has at least one vertex and between every pair of vertices of which there is a path.

2.3 Agreement Protocol

Consider the consensus problem of a group of N agents over a fixed connected network topology \mathcal{G} with the single-integrator dynamics given by

$$\dot{x}_i(t) = u_i(t), \quad x_i(0) = x_{i0}, \quad t \geq 0, \quad (1)$$

where $x_i(t) \in \mathbb{R}$ is the state, and $u_i(t) \in \mathbb{R}$ is the input for each agent $i \in \{1, \dots, N\}$. The local control protocol for each agent i is given by

$$u_i(t) = \sum_{j \in \mathcal{N}_i} a_{ij}(x_j - x_i). \quad (2)$$

We represent the overall closed-loop system in the compact form as

$$\dot{x}(t) = -\mathcal{L}(\mathcal{G})x(t) \quad (3)$$

where $x(t) \triangleq [x_1(t), \dots, x_N(t)]^T$. The solution of (3) is

$$x(t) = e^{-\mathcal{L}(\mathcal{G})t}x(0). \quad (4)$$

The local voting protocol in (2) guarantees consensus of the single-integrator dynamics in (1) if and only if \mathcal{G} has a spanning tree (see Theorem 2.2 in (Lewis

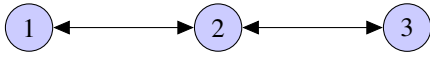


Figure 2: The system in (4) is not identifiable from $x(0)$ for this network topology.

et al., 2014)). It is well-known that each finite connected graph has at least one spanning tree.

This paper studies the identification of $\mathcal{L}(\mathcal{G})$ from a single solution $x(t)$. This identification reveals the underlying network topology \mathcal{G} . However, conventional methods may fail to identify $\mathcal{L}(\mathcal{G})$ from (4). A Laplacian matrix $\mathcal{L}(\mathcal{G})$ is identifiable with (3) from $x(0)$ if and only if the matrix given by

$$R(\mathcal{G}) \triangleq [I - \mathcal{L}(\mathcal{G}) \quad (-\mathcal{L}(\mathcal{G}))^2 \quad \dots \quad (-\mathcal{L}(\mathcal{G}))^{N-1}] x(0) \quad (5)$$

is full rank for all $\mathcal{L}(\mathcal{G})$ in an open set (see Theorem 2.5 in (Stanhope et al., 2014)). Consider any initial condition $x(0) = [x_{10}, x_{20}, x_{30}]^T \in \mathbb{R}^3$ and the network topology \mathcal{G}_1 seen in Figure 2 where

$$\mathcal{L}(\mathcal{G}_1) = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \quad (6)$$

For this particular network topology,

$$R(\mathcal{G}_1) = \begin{bmatrix} x_{10} & x_{20} - 2x_{10} + x_{30} & 6x_{10} - 3x_{20} - 3x_{30} \\ x_{20} & x_{10} - 2x_{20} + x_3 & 6x_{20} - 3x_{10} - 3x_{30} \\ x_{30} & x_{10} + x_{20} - 2x_{30} & 6x_{30} - 3x_{20} - 3x_{10} \end{bmatrix} \quad (7)$$

and $\text{rank}(R(\mathcal{G}_1)) = 2$. The system in (3) with $\mathcal{L}(\mathcal{G}_1)$ faces collinearity problems in least squares estimation for any initial condition $x(0) \in \mathbb{R}^3$.

To overcome this, we train a supervised pattern recognition neural network to identify a network topology. Thus, we avoid identifying \mathcal{L} in $\mathbb{R}^{N \times N}$. Instead, the trained neural network distinguishes $\mathcal{L}(\mathcal{G})$ in the set of all possible N -dimensional communication network topologies.

3 SMALL-SCALE SYSTEMS

We define systems with less than or equal to five nodes (i.e., $N \leq 5$) as *small-scale* systems. The reason is that the number of connected undirected graphs grows dramatically as N increases¹. Due to limitations in training data memory and network training time brought on by this dramatic increase, efficient

¹The integer sequence A001187 in the On-Line Encyclopedia of Integer Sequences presents the number of connected undirected graphs (i.e., the length of $\mathcal{G}_{\text{list}}^N$) for $N \in \{0, 1, 2, \dots\}$.

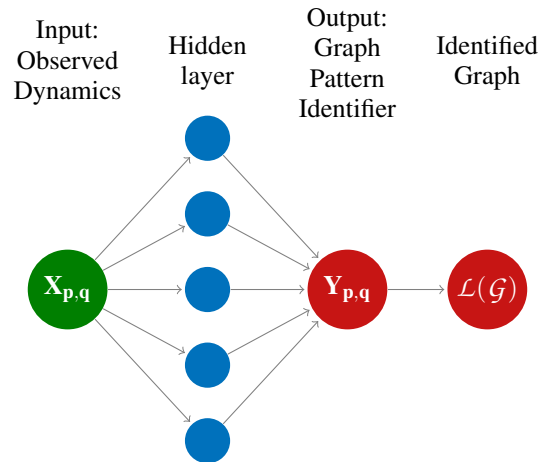


Figure 3: Illustration of the proposed pattern recognition network for small-scale system network topology identification.

training is not manageable when $N > 5$. Although it is possible for the network training dataset to be expansive enough to cover at most 728 such graphs when $N \leq 5$ (e.g., our training dataset for a 4-node system occupied roughly 20 MB), it is expensive in terms of both memory and training time to construct a training dataset that sufficiently captures the dynamics of cases involving all such graphs when $N = 6$ (26704), $N = 7$ (1866256), or greater.

The purpose of the neural network is to identify the graph \mathcal{G} , which is illustrated in Fig 3. Thus, $x(t)$ is the input and \mathcal{G} is the output of the neural network system. We train the neural network with solutions in (4) covering as many as possible scenarios. To this end, we use all of the connected undirected graphs with degree N . Instead of considering all \mathbb{R}^N , we focus on $[0, 1]^N$ to get samples from all possible initial conditions $x(0)$. Any solution in (4) can be scaled and translated such that $x(0) \in [0, 1]^N$. We present the details throughout this section.

3.1 The Training Data Set

3.1.1 Generating All Connected Undirected Graphs

For N number of nodes, there are $\binom{N}{2}$ many possible undirected graphs; we generate all of them. An undirected graph is connected if and only if the rank of the corresponding Laplacian matrix is $N - 1$ (see Remark 4 in (Olfati-Saber and Murray, 2004)). Using this theorem, we choose the connected undirected graphs with degree N among all of the possible graphs with degree N and store them in a set $\mathcal{G}_{\text{list}}^N$.

3.1.2 Generating Initial Conditions

The set $x_{\text{samples}} \triangleq \{k/K \mid k \in \{0, \dots, K\}, K \in \mathbb{Z}_{++}\}$ contains real numbers $0, 1/K, 2/K, \dots, 1$ where $1/K$ represents the grid spacing. We generate initial conditions $x(0)$ using the elements in x_{samples} such that $x_{10} = 0, x_{N0} = 1, x_{i0} \in x_{\text{samples}}$, and

$$x_{10} \leq x_{20} \leq \dots \leq x_{N0}. \quad (8)$$

For instance, $[0, 0, 1]^T$, $[0, 0.5, 1]^T$, and $[0, 1, 1]^T$ are the initial conditions for $N = 3$ and $K = 2$. We store the initial conditions in a set $x_{0,\text{list}}$.

Note that any initial condition $x(0) \in \mathbb{R}^N$ is consistent with (8) after proper relabeling of the nodes. Therefore, this method avoids the redundant solutions and decreases the memory requirements.

3.1.3 Generating the Training Dataset

Let $\Delta t > 0$ be the chosen sampling period (timestep length), n_s be the number of samples. The product of n_s and Δt forms the total duration of the simulated dynamics. Let also e_p^N be the N -dimensional vector of all zeros except its p^{th} entry. e_p^N serves as an ‘‘identifier’’ for the p^{th} graph. For all $\mathcal{G}_p \in \mathcal{G}_{\text{list}}$ and $x_q \in x_{0,\text{list}}$, the inputs

$$X_{p,q} \triangleq [x_q, e^{-\mathcal{L}(\mathcal{G}_p)\Delta t} x_q, \dots, e^{-\mathcal{L}(\mathcal{G}_p)n_s \Delta t} x_q] \quad (9)$$

and the outputs $Y_{p,q} \triangleq e_p^N$ construct the training data set $D = \{(X_{p,q}, Y_{p,q}) \mid p = 1, \dots, |\mathcal{G}_{\text{list}}^N|, q = 1, \dots, |x_{0,\text{list}}|\}$. We trained the supervised pattern recognition neural networks using all input-output data in D using MATLAB.

3.2 Validation

We train the network with the initial condition samples which belong to set $[0, 1]^N$. However, the initial condition belongs to \mathbb{R}^N in general. The following lemma is useful to scale and translate any solution (4) to modify its initial condition to be consistent with training data.

Lemma 1. For any graph \mathcal{G} , $x(t) = e^{-\mathcal{L}(\mathcal{G})t} x(0)$ if and only if $S(x(t) + T\mathbf{1}_N) = e^{-\mathcal{L}(\mathcal{G})t} S(x(0) + T\mathbf{1}_N)$ where $S \neq 0$ and $S, T \in \mathbb{R}$.

Proof. The matrix exponential is the power series as

$$e^{-\mathcal{L}(\mathcal{G})t} = I + \sum_{k=1}^{\infty} \frac{-\mathcal{L}(\mathcal{G})t^k}{k!} \quad (10)$$

by definition. As per graph theory, $\mathcal{L}(\mathcal{G})\mathbf{1}_N = 0$ since $\mathbf{1}_N$ is the right eigenvector corresponding to the

eigenvalue 0 of $\mathcal{L}(\mathcal{G})$ (Lewis et al., 2014). Thus, $e^{-\mathcal{L}(\mathcal{G})t}\mathbf{1}_N = \mathbf{1}_N$ from (10). The rest of the proof is algebraic manipulations. \square

Consider that we observe the outputs $\bar{x}_i(t)$ of a multi-agent system with an underlying communication topology \mathcal{G}_1 . We assume $\bar{x}_1(0) \leq \dots \leq \bar{x}_N(0)$ without loss of generality. Otherwise, we can relabel the nodes to satisfy this constraint. Let the observation matrix be

$$\bar{X} = [\bar{x}(0), \bar{x}(\Delta t), \dots, \bar{x}(n_s \Delta t)], \quad (11)$$

$S = \max_i \bar{x}_i(0) - \min_i \bar{x}_i(0)$, and $T = -\min_i \bar{x}_i(0)$. Note that $x(0) \in [0, 1]^N$ where $x(t) = S(\bar{x}(t) + T\mathbf{1}_N)$. Thus, $X = S(\bar{X} + T(\mathbf{1}_N \otimes \mathbf{1}_{n_s}))$ is an appropriate observation matrix to identify \mathcal{G}_1 using trained pattern recognition network. The underlying graph is same for both X and \bar{X} from Lemma 1. Thus, the output of the pattern recognition neural network is $\mathcal{L}(\mathcal{G}_1)$.

4 LARGE-SCALE SYSTEMS

We define systems with more than five nodes, $N > 5$, as *large-scale systems*. In this case, we present an algorithm to detect the most connected node (‘‘primary’’ node) of a communication network topology. To this end, we use a pattern recognition network which is trained for a small-scale system.

4.1 Determining the Most-connected Node of Large-scale Systems

Consider the outputs $\hat{x}_i(t)$ of a large-scale multi-agent system. The following algorithm aims to reveal the most connected node of its communication network topology.

Step 0: This step contains offline training and computations which are performed once and stored.

Step 0.1: Set $K > 0$ and $3 \leq N_c \leq 5$. Generate all connected undirected graphs $\mathcal{G}_{\text{list}}^{N_c}$, generate initial condition samples list $x_{0,\text{list}}$ with N_c and K , and generate the data set D (see Section 3). Train the pattern recognition network using D .

Step 0.2: Let $I_N \triangleq \{1, 2, \dots, N\}$. Compute all N_c combinations of I_N , $C = \{C_r \in \mathcal{P}(I_N) \mid |C_r| = N_c\}$.

Step 1: Set the number of links $\alpha_i = 0, i = 1, \dots, N$.

Step 2: Iterate for all $C_r \in C$ where $r = 1, \dots, |C|$.

Step 2.1: Let $\bar{C}_r \triangleq I_N \setminus C_r$, and $x_c(t) \triangleq \sum_{i \in \bar{C}_r} \hat{x}_i(t)$. Compute $\bar{x}_i(t) \triangleq (N_c \hat{x}_i(t) + x_c(t))/N, i \in C_r$.

Step 2.2: Consider the observation of $\bar{x}_i(t)$ from a small-scale system with degree N_c . Construct the observation matrix X by reordering, scaling and translating the set of $\bar{x}_i(t)$ (see Subsection 3.2).

Step 2.3: Identify $\mathcal{L}(\mathcal{G})$ using the trained pattern recognition network. Increment the corresponding number of links α_i using $\mathcal{L}(\mathcal{G})$.

Step 3: The most connected node is the one owning the largest number of links $\arg \max_i \alpha_i$.

5 RESULTS

Some aspects of our testing procedure remained the same for both small-scale and large-scale systems. Training of the pattern recognition neural network in each case was done through the `patternnet` function in MATLAB’s Deep Learning toolbox (The MathWorks, Inc., 2019), with a single hidden layer and otherwise default arguments. The size of this hidden layer was varied from 5 to 75 to view its effect on accuracy of the small-scale systems, but was set to 50 for the large-scale systems. In the small-scale system, in which generation of all possible graphs $\mathcal{G}_{\text{list}}^3$ for training and testing purposes was possible due to its small length², population of $\mathcal{G}_{\text{list}}^3$ was done ourselves. In the large-scale system application for $N \in \{7, 8, 9\}$, the training with graphs from $\mathcal{G}_{\text{list}}^4$ was accomplished similarly. However, generation of $\mathcal{G}_{\text{list}}^N$ for large-scale systems imposes a high computational load and therefore requires a long duration. We avoid this requirement by rather drawing a small random sample \mathcal{G}_n^N of size n from $\mathcal{G}_{\text{list}}^N$ for the testing of the large-scale systems. For this sampling, we utilize **nauty** (McKay and Piperno, 2013), a tool that can quickly generate graphs; in particular, we make use of **nauty**’s `genrang` program.

5.1 Small-scale Systems

Several pattern-recognition networks were trained to predict the exact network topology of the small-scale systems. We varied the size of the training dataset by manipulating K , since K influences the length of $x_{0,\text{list}}$, which corresponds to the number of initial conditions generated. As K increases, the number of combinations of initial conditions and graphs used to generate the training data grows. In addition, as previously mentioned, we varied the size of the hidden layer to check its impact on the accuracy of the predictions. We tested the trained pattern recognition

²See the aforementioned A001187.

network using a dataset assembled similarly to the training data, but whose initial conditions were not restricted to the “sample” space x_{sample} . The results of the testing for each combination of K and hidden layer size are presented in Table 1.

Table 1: Identification accuracy for \mathcal{G}^3 with respect to various K and size of the hidden layer.

		Size of the Hidden Layer			
		5	25	50	75
K	5	31.08%	65.59%	78.96%	95.05%
	10	61.36%	98.26%	96.78%	94.31%
	15	97.53%	98.76%	98.51%	97.03%
	20	97.61%	98.76%	97.05%	99.19%

The network becomes more accurate as the number of initial conditions used for training grows; we expect this trend, as the discrepancy of the network’s initial condition space decreases with increasing K . A similar trend is observed for the increase in hidden layer size. However, it is appropriate to be wary of the tradeoff between high network accuracy and both overfitting as the size of the hidden layer grows and higher memory requirements as K grows.

As an aside, it should be noted again that the distinction between small-scale and large-scale systems is largely motivated by neural network training time. As an example: a network trained for a 4-node system with $K = 10$ and 5 hidden layers required just 52 seconds for training, whereas a network trained with the same configuration for a 5-node system required 172 minutes for training, so the increase in training time is quite significant.

5.2 Large-scale Systems

Using the process put forth in Section 3.2, we trained a pattern-recognition network using initial condition samples belonging to $[0, 1]^4$ (i.e., data from a 4-node small-scale system). Thereafter, through the following method we generated a testing dataset through which this small-scale-trained network was used to predict the primary and secondary nodes of a system with $N = 7$. A set of five initial conditions x_5 for $N = 7$ was first generated using the method outlined in Section 3. A subset \mathcal{G}_n^7 of size n was drawn from $\mathcal{G}_{\text{list}}^7$ through **nauty**. For every combination (i, j) of initial condition $x_i, i \in \{1, \dots, 5\}$ and graph $\mathcal{G}_n^j, j \in \{1, \dots, n\}$, the test inputs X were generated through Equation 9. The algorithm described in Section 4.1 was then used to identify the most connected (primary), as well as the second-most-connected (secondary), nodes for each combination (i, j) making up the test dataset.

The primary “success” of the algorithm for a given graph sample size n was defined as the percentage of correct primary node identification across all combinations (i, j) ; the secondary success was defined similarly. This entire testing process was repeated 3 times for each sample size $n \in \{10, 20, \dots, 250\}$ and the average success was taken to be the mean of the primary success over the 3 repetitions for each sample size.

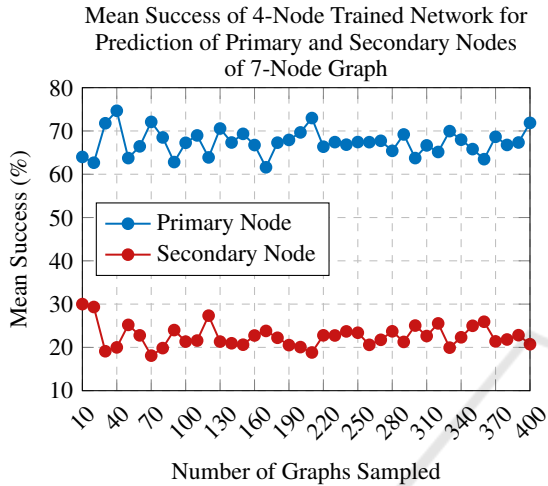


Figure 4: Results of testing a pattern-recognition network trained on data from a small-scale (4-node) system on large-scale (7-node) data.

Figure 4 displays the results of the large-scale testing for $N = 7$ as the testing dataset grows in size proportionally to n , the number of samples from $\mathcal{G}_{\text{list}}^7$. The network trained on small-scale data is successful in identification of the primary node in roughly 65% of cases; in addition, it is successful in identification of the secondary node in roughly 25% of the cases.

The same process was also used to test the small-scale network for primary/secondary node prediction of large-scale systems of size $N \in \{8, 9\}$. We include the visualizations of those results (Figs. 6 and 7), which yielded similar results to the 7-node test case, in the appendix.

We also repeated the large-scale testing for the 7-node configuration with noise in the single-integrator consensus dynamics included to test the robustness of our method to uncertainty. Whereas the original configuration assumed “perfect” observation of agent dynamics, white Gaussian noise models the uncertainty in the observations of the positions of each agent, which would be present in a real countersurveillance scenario. The results of the simulation including white Gaussian noise with variances 0.1 and 0.25 in observations of each agent are displayed in Figure 5. Comparison to Figure 4 indicates that our method is generally robust to this uncertainty, as the accuracy

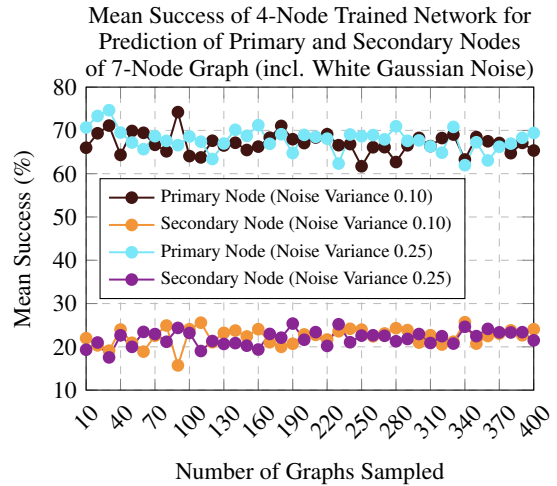


Figure 5: Results of testing a pattern-recognition network trained on data from a small-scale (4-node) system on large-scale (7-node) data, including white Gaussian noise with variances 0.1 and 0.25 in the single-integrator consensus dynamics.

in identification of the primary and secondary nodes does not change significantly for either uncertainty variance examined.

We approximate the convergence of our method via these simulations. Based on the results of the Monte Carlo simulations for $N \in \{7, 8, 9\}$ sampling up to 400 graphs from the pools, our method is generally successful in identification of the most connected node in at least 60% of cases for large-scale ($N > 5$) systems, even in the presence of uncertainty modelled as white Gaussian noise. It is also generally successful in identification of the second-most connected node in at least 20% of cases.

6 CONCLUSIONS

We found that using our method, the pattern-recognition network was highly accurate in predicting the network topology of small-scale systems. Furthermore, such a pattern-recognition network was successful in recognizing the most connected nodes of large-scale systems. Future work may focus on training the network to recognize network topologies of systems with different closed-loop dynamics apart from that of the linear consensus protocol.

REFERENCES

- Bidram, A., Lewis, F. L., and Davoudi, A. (2014). Distributed control systems for small-scale power networks: Using multiagent cooperative control theory. *IEEE Control Systems Magazine*, 34(6):56–77.
- de Roo, F. V., Tejada, A., van Waarde, H., and Trentelman, H. L. (2015). Towards observer-based fault detection and isolation for branched water distribution networks without cycles. In *2015 European Control Conference (ECC)*, pages 3280–3285. IEEE.
- Gonçalves, J. and Warnick, S. (2008). Necessary and sufficient conditions for dynamical structure reconstruction of lti networks. *IEEE Transactions on Automatic Control*, 53(7):1670–1674.
- Julius, A., Zavlanos, M., Boyd, S., and Pappas, G. J. (2009). Genetic network identification using convex programming. *IET systems biology*, 3(3):155–166.
- Lewis, F. L., Zhang, H., Hengster-Movric, K., and Das, A. (2014). *Cooperative control of multi-agent systems: Optimal and adaptive design approaches*. Springer-Verlag, London, England.
- McKay, B. D. and Piperno, A. (2013). Nauty and traces user’s guide (version 2.5). *Computer Science Department, Australian National University, Canberra, Australia*.
- Nabi-Abdolyousefi, M. and Mesbahi, M. (2012). Network identification via node knockout. *IEEE Transactions on Automatic Control*, 57(12):3214–3219.
- Olfati-Saber, R. and Murray, R. M. (2004). Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on automatic control*, 49(9):1520–1533.
- Ota, J. (2006). Multi-agent robot systems as distributed autonomous systems. *Advanced engineering informatics*, 20(1):59–70.
- Rahimian, M. A., Ajorlou, A., Tutunov, R., and Aghdam, A. G. (2013). Identifiability of links and nodes in multi-agent systems under the agreement protocol. In *2013 American Control Conference*, pages 6853–6858. IEEE.
- Scharre, P. (2018). How swarming will change warfare. *Bulletin of the atomic scientists*, 74(6):385–389.
- Stanhope, S., Rubin, J. E., and Swigon, D. (2014). Identifiability of linear and linear-in-parameters dynamical systems from a single trajectory. *SIAM Journal on Applied Dynamical Systems*, 13(4):1792–1815.
- Sun, C. and Dai, R. (2015). Identification of network topology via quadratic optimization. In *2015 American Control Conference (ACC)*, pages 5752–5757. IEEE.
- Tan, Y. and Zheng, Z. (2013). Research advance in swarm robotics. *Defence Technology*, 9(1):18–39.
- Teodorovic, D. (2003). Transport modeling by multi-agent systems: a swarm intelligence approach. *Transportation planning and Technology*, 26(4):289–312.
- The MathWorks, Inc. (2019). *Deep Learning Toolbox, MATLAB 2019a*. Natick, Massachusetts, U.S.
- Valdés-Sosa, P. A., Sánchez-Bornot, J. M., Lage-Castellanos, A., Vega-Hernández, M., Bosch-Bayard, J., Melie-García, L., and Canales-Rodríguez, E. (2005). Estimating brain functional connectivity with sparse multivariate autoregression. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1457):969–981.
- van Waarde, H. J., Tesi, P., and Camlibel, M. K. (2019). Topology reconstruction of dynamical networks via constrained lyapunov equations. *IEEE Transactions on Automatic Control*, 64(10):4300–4306.
- Wu, X., Zhao, X., Lü, J., Tang, L., and an Lu, J. (2015). Identifying topologies of complex dynamical networks with stochastic perturbations. *IEEE Transactions on Control of Network Systems*, 3(4):379–389.

APPENDIX

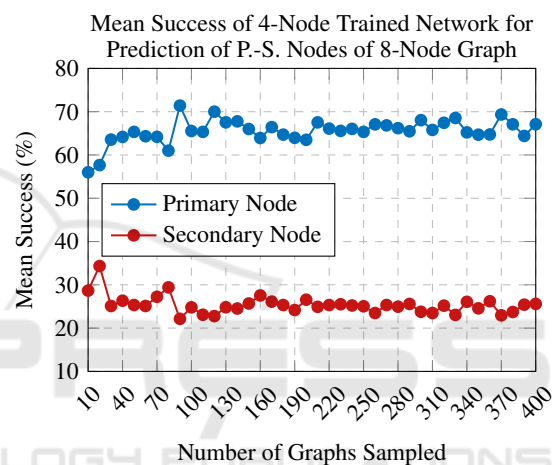


Figure 6: Results of testing a pattern-recognition network trained on data from a small-scale (4-node) system on large-scale (8-node) data.

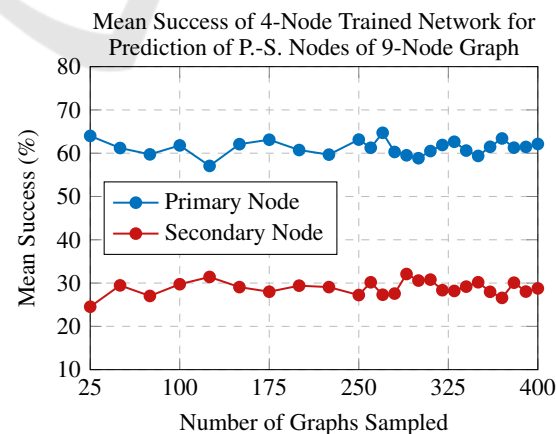


Figure 7: Results of testing a pattern-recognition network trained on data from a small-scale (4-node) system on large-scale (9-node) data.