2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

Distributed Model-Free Policy Iteration for Networks of Homogeneous Systems

Shahriar Talebi, Siavash Alemzadeh, and Mehran Mesbahi

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9683331

***Abstract*—Control of large-scale networked systems often necessitates modeling complex interactions amongst agents. However, as the size of the network increases, modeling these interactions often becomes exponentially expensive in applications. In this paper, we propose a distributed model-free policy iteration algorithm to design a feedback mechanism for large networks of homogeneous systems. We assume that the networked system is built upon an underlying information-exchange graph allowing the distributed controller to synthesize a feedback signal using information from adjacent agents. This model-free approach provides a stabilizing distributed feedback controller through a learning phase. In particular, a data-driven control method is utilized to circumvent model uncertainties by directly synthesizing a controller based on data that is obtained from a relatively small subgraph of the original network. Additionally, a stability margin is learned from data which is then utilized to design a suboptimal distributed controller for the entire network even during the learning phase. We showcase the performance of our methodology by examining distributed control scenarios involving modeling errors.**

Keywords: Distributed policy iteration, Data-driven control, Networked control systems

I. Introduction

There has been a renewal of interest in distributed control during the past few years mainly due to the complexity in modeling and analysis of large-scale systems. From a modeling viewpoint, the distributed character of the problem can originate from the underlying interconnections, subsystems' objectives, and sparsity in the information exchange. However, distributed (structured) control is an NP-hard constrained optimization problem in general. As such, the main focus in the area of distributed (structured) control synthesis is not finding the optimal solution per se, but an efficient control mechanism by exploiting the problem structure. Such a trade-off has been rigorously examined for a broad range of control applications such as robotic swarms, power grids, and formation flight, just to name a few [1], [2]. In this direction, sufficient graph-theoretic conditions have been provided for stability of formations comprised of identical vehicles [3] and, along the same lines, graph-based distributed controller synthesis has been examined in works such as [4]–[8]. The topic has also been studied from a spatially distributed control viewpoint [9], [10] and a compositional layered design approach [11]. Moreover, from an agent-level perspective, the problem has been tackled for both homogeneous systems [4], [5], [7], and more recently, heterogeneous networks [12].

In this work, we propose an LQR-based distributed model-free policy iteration algorithm to find a set of stabilizing controllers for a homogeneous network of identical linear systems. Without the knowledge of system parameters, the synthesis process leads to a data-driven feedback mechanism for the entire networked system that is trained based on data collected from only a (possibly small) portion of the network. The proposed approach considers a system in the homogeneous network corresponding to the node with maximum degree in the underlying graph; we then build the control mechanism using a clique subnetwork containing this node via $\mathcal{Q}$-learning-based policy iteration. The "clique controller" is then extended to the entire network. Additionally, we learn a stability margin for the control system of a single agent in this network. The stability margin analysis proves instrumental for the proposed synthesis procedure, as it facilitates constructing a compounded suboptimal feedback that is stabilizing for the entire networked system. We also provide examples that showcase the usefulness of the proposed approach for uncertain networked systems.

The remainder of the paper is organized as follows. In §II, we introduce the problem setup, the challenges involved, and the motivations behind our approach. In §III, we present and explicate our algorithm. We provide theoretical results regarding the methodology in §IV; we refer to the extended version of this work [13] for notation and some of the proofs. Illustrative simulations are provided in §V, followed by concluding remarks in §VI.

II. Problem Setup

Consider a network of identical agents with interdependencies implied by a common network-level objective. In particular, we assume that the system contains N agents forming a graph $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$, where each node in $\mathcal{V}_\mathcal{G}$ represents a linear discrete-time (decoupled) dynamical system,

$$\boldsymbol{x}_{i,t+1} = A\boldsymbol{x}_{i,t} + B\boldsymbol{u}_{i,t}, \qquad i = 1, 2, \ldots, N, \quad (1)$$

with $\boldsymbol{x}_{i,t} \in \mathbb{R}^n$ and $\boldsymbol{u}_{i,t} \in \mathbb{R}^m$ denoting the states and inputs of agent i at time-step t, respectively. Furthermore, $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are *unknown* system parameters. The formulation can be compactly represented as,

$$\hat{\mathbf{x}}_{t+1} = \hat{\mathbf{A}}\hat{\mathbf{x}}_t + \hat{\mathbf{B}}\hat{\mathbf{u}}_t, \quad (2)$$

where $\hat{\mathbf{x}}_t \in \mathbb{R}^{Nn}$ and $\hat{\mathbf{u}}_t \in \mathbb{R}^{Nm}$ contain the states and inputs of entire networked systems, $\hat{\mathbf{x}}_t =$

The authors are with the William E. Boeing Department of Aeronautics and Astronautics at the University of Washington, Seattle. The research of the authors has been supported by AFOSR grant FA9550-20-1-0053. Emails:{shahriar, alems, mesbahi}@uw.edu

$\big[\boldsymbol{x}_{1,t}^\intercal;\ldots;\boldsymbol{x}_{N,t}^\intercal\big]^\intercal$, $\hat{\mathbf{u}}_t = \big[\boldsymbol{u}_{1,t}^\intercal;\ldots;\boldsymbol{u}_{N,t}^\intercal\big]^\intercal$, and $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ assume block diagonal forms $\hat{\mathbf{A}} = \mathrm{I}_N \otimes A$ and $\hat{\mathbf{B}} = \mathrm{I}_N \otimes B$.

The interconnections of these agents are captured by the set of edges $\mathcal{E}_\mathcal{G} \subseteq \mathcal{V}_\mathcal{G} \times \mathcal{V}_\mathcal{G}$ that provide the communication backbone for the distributed feedback controller design. We do *not* assume that $\mathcal{G}$ is necessarily connected; this setup becomes apparent in §III. Let $\mathcal{N}_i$ denote the set of neighbors of node i in $\mathcal{G}$ (excluding itself). Then, based on the communication graph and for any choice of positive integers a and b, we define a linear subspace of $\mathbb{R}^{aN\times bN}$,

$$\mathcal{U}_{a,b}^N(\mathcal{G}) \coloneqq \big\{\mathbf{M}\in\mathbb{R}^{aN\times bN} |\ [\mathbf{M}]_{ij} = \mathbf{0} \text{ if } j\notin\mathcal{N}_i\cup\{i\},\\ [\mathbf{M}]_{ij}\in\mathbb{R}^{a\times b},\ i,j=1,\cdots,N\big\},$$

where $[\mathbf{M}]_{ij}$ refers to its (ij)-th block-submatrix. Without having access to the system parameters A and B, we are interested in designing linear feedback gains with a desired sparsity pattern using input-state measurements from—ideally a small part of—the system in (2). Of particular interest in this work are the structured feedback gains consistent with the underlying communication network. Given an initial condition $\hat{\mathbf{x}}_1$, the distributed (structured) optimal control problem is posed as,

$$\begin{aligned}\min_{\hat{\mathbf{K}}}\quad & \textstyle\sum_{t=1}^\infty \hat{\mathbf{x}}_t^\intercal\hat{\mathbf{Q}}\hat{\mathbf{x}}_t + \hat{\mathbf{u}}_t^\intercal\hat{\mathbf{R}}\hat{\mathbf{u}}_t\\ \text{s.t.}\quad & (2),\quad \hat{\mathbf{u}}_t = \hat{\mathbf{K}}\hat{\mathbf{x}}_t,\quad \hat{\mathbf{K}}\in\mathcal{U}_{m,n}^N(\mathcal{G}),\end{aligned}\tag{3}$$

where $\hat{\mathbf{K}}$ stabilizes the pair $(\hat{\mathbf{A}},\hat{\mathbf{B}})$, $\hat{\mathbf{R}} = \mathrm{I}_N\otimes R$ and $\hat{\mathbf{Q}} = I_N\otimes Q_1 + \mathcal{L}_\mathcal{G}\otimes Q_2$ for some cost matrices $Q_1\succ 0$, $Q_2\succeq 0$, $R\succ 0$. Note that $\hat{\mathbf{Q}}\in\mathcal{U}_{n,n}^N(\mathcal{G})$ and positive definite. Such interdependent structure of the cost has been considered in a graph-based distributed control framework (see for instance [4], [5], [7], [14]). Intuitively, the first term in $\hat{\mathbf{Q}}$ indicates the absolute cost pertinent to regulating each agent, while the second term denotes the cost that arises from consensus error with respect to the neighbors' states.

The goal is to find a data-guided suboptimal solution to (3) without the knowledge of system parameters $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$. Indeed, not knowing the true underlying system parameters adds an additional layer of difficulty to this framework. Instead, input-state data of the system can be collected for distributed feedback control synthesis over the given underlying communication graph $\mathcal{G}$. In what follows, we summarize the challenges involving the analysis of this problem: I) In general, the constrained optimization problem (3) is NP-hard [15]. Based on the complete knowledge of the system parameters, this problem has been investigated under variety of assumptions [5], [9], [16], [17], or tackled directly with the aid of projected gradient-based policy updates [6], [18]. II) On the other hand, the main issue with the policy obtained from data-driven approaches is that, it will not necessarily respect the hard constraint of $\hat{\mathbf{K}}\in\mathcal{U}_{m,n}^N(\mathcal{G})$ in (3). Also, "projection" onto the intersection of this constraint and the set of stabilizing controllers is not straightforward due to the complicated geometry of the set of stabilizing controllers [19]. III) Another significant challenge for adopting data-driven methods for the entire network takes root in the "curse of dimensionality" inherent to networked systems. In fact, collecting data from the entire large-scale network can be prohibitive. IV) Finally, it is often impractical in standard applications to terminate a functioning network for data collection or decision-making purposes. Therefore, we seek an online algorithm that allows the network to continue operating during the learning process.

As the control system designer is left with no choice than leveraging the input-state observations to find a "reasonable" distributed policy for the networked system in the setup of interest, we summarize our approach as follows:

I) As the constraint $\hat{\mathbf{K}}\in\mathcal{U}_{m,n}^N(\mathcal{G})$ is strict in distributed control synthesis and the underlying problem is NP-hard in general, we shift our attention from the optimal solution of (3) towards a suboptimal stabilizing distributed controller with a reduced computational overhead. II) Inspired by a $\mathcal{Q}$-learning-based policy iteration algorithm, we propose a model-free distributed policy iteration scheme to obtain a reasonable solution to (3). In particular, we learn two control components that mirror "individual" and "cooperative" aspects of the design. We also simultaneously learn a stability margin related to these components that is utilized to prescribe a distributed control synthesis for the entire network. III) Inherent to our algorithm is a subproblem whose dimension is only related to the maximum degree of the underlying graph rather than the dimension of the original network. In particular, we will reason that for the learning phase, our method requires data collection only from a smaller portion of the underlying network $\mathcal{G}_d\subseteq\mathcal{G}$ with size $d = d_{\max}(\mathcal{G})+1$—in order to guarantee stability (see Theorem 1). This subgraph is substantially smaller than the original graph whenever $d_{\max}(\mathcal{G})\ll N$ which is the case in many real-world applications. IV) We allow temporary feedback links on $\mathcal{G}_d$ during the learning phase of our algorithm that would be removed subsequently. Hence, the proposed approach allows for an online implementation.

As described in the subsequent sections, the distributed control design part of our method follows a model-based framework previously studied in [5], [7]. Here, we provide a model-free distributed policy iteration algorithm that is not only computationally efficient, but also practical when model-based control in high dimensions is not feasible.

III. The Algorithm

In this section, we propose and discuss the main algorithm. Before proceeding to describe the main idea, we introduce a definition that streamlines the subsequent notation.

Definition 1. Given any subgraph $\mathcal{G}'\subseteq\mathcal{G}$ and a fixed labeling of the nodes, we let $\mathrm{Policy}\,(\mathcal{V}_{\mathcal{G}'})$ denote the concatenation of all the policies of the agents in $\mathcal{V}_{\mathcal{G}'}$, i.e.,

$$\mathrm{Policy}\,(\mathcal{V}_{\mathcal{G}'}) \coloneqq [\boldsymbol{u}_1^\intercal\ ;\ \boldsymbol{u}_2^\intercal\ ;\ \cdots\ ;\ \boldsymbol{u}_{|\mathcal{V}_{\mathcal{G}'}|}^\intercal]^\intercal,$$

where $\boldsymbol{u}_i$ is the feedback control policy of agent i in the subgraph $\mathcal{G}'$ as a mapping from $\{\boldsymbol{x}_j | j\in\mathcal{N}_i\cup\{i\}\}$ to $\mathbb{R}^m$. Furthermore, we use $\mathrm{Policy}(\mathcal{V}_{\mathcal{G}'})|_t$ to denote the realization of these policies at time t. Similarly, we define,

$$\mathrm{State}\,(\mathcal{V}_{\mathcal{G}'}) \coloneqq [\boldsymbol{x}_1^\intercal\ ;\ \boldsymbol{x}_2^\intercal\ ;\ \cdots\ ;\ \boldsymbol{x}_{|\mathcal{V}_{\mathcal{G}'}|}^\intercal]^\intercal.$$

Algorithm 1 Distributed Model-Free Policy Iteration

1: **Initialization** $(t \leftarrow 1,\ k \leftarrow 1)$
2: Given $\mathcal{G}$, choose $\mathcal{G}_d \subseteq \mathcal{G}$ with $d = d_{\max}(\mathcal{G}) + 1$
3: Obtain Q_1, Q_2, R, and set $\widetilde{Q} \leftarrow Q_1 + dQ_2$
4: Get $\tilde{\mathbf{x}}_1 \in \mathbb{R}^{dn}$ and $\widetilde{\mathbf{H}}_0 \in \mathbb{R}^{p\times p}$ with $p = d(n+m)$
5: Set $\mathcal{P}^\circ \leftarrow \beta \mathrm{I}_{p(p+1)/2}$ for large $\beta > 0$ and fix Σ
6: Set K_1 stabilizing (1), $L_1 = \mathbf{0}_{m\times n}$ and $\Delta K_1 = K_1$
7: Switch ON temporary links in $\mathcal{G}_d$ and set $\tau_1 \leftarrow 0$
8: **While** (K_k, L_k) **not converged, do** ("learning phase")
9: Set $\mathrm{Policy}_k(\mathcal{V}_\mathcal{G})$ such that for each $i \in \mathcal{V}_\mathcal{G}$,
$\boldsymbol{u}_i \leftarrow \Delta K_k \boldsymbol{x}_i + L_k \sum_{j\in\mathcal{N}_i} \frac{\tau_k}{d-1}\boldsymbol{x}_j$, if $i \in \mathcal{V}_{\mathcal{G}\setminus\mathcal{G}_d}$
$\boldsymbol{u}_i \leftarrow \Delta K_k \boldsymbol{x}_i + L_k \sum_{j\in\mathcal{V}_{\mathcal{G}_d}} \boldsymbol{x}_j$, if $i \in \mathcal{V}_{\mathcal{G}_d}$
10: Evaluate $\widetilde{\mathbf{H}}_k$ from Algorithm 2
$\widetilde{\mathbf{H}}_k \leftarrow \mathrm{SPE}\big(\mathcal{G},\ \mathcal{G}_{d,\mathrm{learn}},\ \mathrm{Policy}_k(\mathcal{V}_\mathcal{G}),\ \widetilde{\mathbf{H}}_{k-1}, \mathcal{P}^\circ\big)$
11: Recover X_1, X_2, Y_1, Y_2, Z_1 and Z_2 from $\widetilde{\mathbf{H}}_k$
$X_1 \leftarrow \widetilde{\mathbf{H}}_k[1:n,\ 1:n]$
$Y_1 \leftarrow \widetilde{\mathbf{H}}_k[dn+1:dn+m,\ dn+1:dn+m]$
$Z_1 \leftarrow \widetilde{\mathbf{H}}_k[dn+1:dn+m,\ 1:n]$
$X_2 \leftarrow \widetilde{\mathbf{H}}_k\big[1:n,\ n+1:2n\big]$
$Y_2 \leftarrow \widetilde{\mathbf{H}}_k[dn+1:dn+m,\ dn+m+1:dn+2m+1]$
$Z_2 \leftarrow \widetilde{\mathbf{H}}_k[dn+1:dn+m,\ n+1:2n]$
$\Delta X \leftarrow X_1 - X_2, \quad \Delta Y \leftarrow Y_1 - Y_2, \quad \Delta Z \leftarrow Z_1 - Z_2.$
12: Update the control components
$F^{-1} \leftarrow Y_1 - (d-1)Y_2\big(Y_1 + (d-2)Y_2\big)^{-1}Y_2$
$G \leftarrow (Y_1 + (d-1)Y_2)^{-1}\, Y_2\, (Y_1 - Y_2)^{-1}$
$K_{k+1} \leftarrow -FZ_1 + (d-1)GZ_2,$
$L_{k+1} \leftarrow -FZ_2 + GZ_1 + (d-2)GZ_2,$
$\Delta K_{k+1} \leftarrow K_{k+1} - L_{k+1}$
13: Obtain the stability margin
$\Xi_{k+1} \leftarrow \Delta X - \widetilde{Q} + \Delta K_{k+1}^\intercal \Delta Z + \Delta Z^\intercal \Delta K_{k+1} + \Delta K_{k+1}^\intercal \big(\Delta Y - R\big)\Delta K_{k+1}$
$\gamma_{k+1} \leftarrow \sigma_{\min}\Big(\Delta K_{k+1}^\intercal R \Delta K_{k+1} + \widetilde{Q}\Big) \Big/ \sigma_{\max}\Big(\Xi_{k+1} + L_{k+1}^\intercal (\Delta Y - R) L_{k+1}\Big)$
$\tau_{k+1} \leftarrow \sqrt{\gamma_{k+1}^2/(1+\gamma_{k+1})}$
14: Go to Line 8 and set $k \leftarrow k+1$
15: Switch OFF the temporary links and retrieve $\mathcal{G}_d$
16: Set $\mathrm{Policy}_k(\mathcal{V}_\mathcal{G})$ such that for each $i \in \mathcal{V}_\mathcal{G}$,
$\boldsymbol{u}_i \leftarrow \Delta K_k \boldsymbol{x}_i + \frac{\tau_k}{d-1} L_k \sum_{j\in\mathcal{N}_i} \boldsymbol{x}_j$

The proposed algorithm is detailed in Algorithm 1. We refer to the main loop of the algorithm in Line 8 as the *learning phase*. We consider a subgraph $\mathcal{G}_d \subseteq \mathcal{G}$, and during that phase we only collect data from this (smaller) portion of the network. Note that the size of the subgraph $\mathcal{G}_d$ depends on the maximum degree of the original graph $\mathcal{G}$. Then, by requiring temporary communication links within this subgraph (denoted by $\mathcal{G}_{d,\mathrm{learn}}$), we iteratively learn a stabilizing structured controller for the entire network allowing the rest of the network to evolve according to the original communication topology (see Figure 1). We distinguish this subgraph with temporary links by $\mathcal{G}_{d,\mathrm{learn}}$ where $\big|\mathcal{V}_{\mathcal{G}_{d,\mathrm{learn}}}\big| = |\mathcal{V}_{\mathcal{G}_d}|$ and $\mathcal{G}_{d,\mathrm{learn}} = \mathcal{K}(\mathcal{G}_d)$, and $\mathcal{K}(.)$ denotes the complete graph (clique) on corresponding nodes. The obtained policy is then optimal for the subgraph—yet, generally suboptimal for the entire network. After the learning phase, the temporary communication links can be removed and a suboptimal stabilizing distributed feedback policy will be proposed for the entire original network.

Algorithm 2 Subgraph Policy Evaluation (SPE)

1: **Input:** Graph $\mathcal{G}$, subgraph $\mathcal{G}' \subseteq \mathcal{G}$, $\mathrm{Policy}(\mathcal{V}_\mathcal{G})$, $\mathbf{H}$, $\mathcal{P}$
2: **Output:** Updated cost matrix $\mathbf{H}^+$ associated with $\mathcal{G}'$
3: **While** $\mathbf{H}$ **not converged, do**
4: Set $\tilde{\mathbf{x}}_t \leftarrow \mathrm{State}(\mathcal{V}_{\mathcal{G}'})|_t$ and $\tilde{\mathbf{u}}_t \leftarrow \mathrm{Policy}(\mathcal{V}_{\mathcal{G}'})|_t$
5: Choose $\boldsymbol{e}_t \sim \mathcal{N}(\mathbf{0}, \Sigma)$ and update $\mathrm{Policy}(\mathcal{V}_{\mathcal{G}'})|_t$ as $\tilde{\mathbf{u}}_t \leftarrow \tilde{\mathbf{u}}_t + \boldsymbol{e}_t$ for all $i \in \mathcal{V}_{\mathcal{G}'}$
6: Run $\mathcal{G}$ under policy $\mathrm{Policy}(\mathcal{V}_\mathcal{G})$
7: Collect $\mathrm{State}(\mathcal{V}_{\mathcal{G}'})|_{t+1}$ only from $\mathcal{G}'$ and set $\tilde{\mathbf{x}}_{t+1} \leftarrow \mathrm{State}(\mathcal{V}_{\mathcal{G}'})|_{t+1}$, $\tilde{\mathbf{u}}_{t+1} \leftarrow \mathrm{Policy}(\mathcal{V}_{\mathcal{G}'})|_{t+1}$
8: Set $\boldsymbol{\phi}_t \leftarrow [\tilde{\mathbf{x}}_t^\intercal\ \tilde{\mathbf{u}}_t^\intercal]^\intercal - [\tilde{\mathbf{x}}_{t+1}^\intercal\ \tilde{\mathbf{u}}_{t+1}^\intercal]^\intercal$
9: Compute $\boldsymbol{\zeta}_t = \mathrm{vech}(\boldsymbol{\phi}_t\boldsymbol{\phi}_t^\intercal)$ and $\mathcal{R}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) = \tilde{\mathbf{x}}_t^\intercal\big(\mathrm{I}\otimes\widetilde{Q} - \mathbb{1}\mathbb{1}^\intercal \otimes Q_2\big)\tilde{\mathbf{x}}_t + \tilde{\mathbf{u}}_t^\intercal\big(\mathrm{I}\otimes R\big)\tilde{\mathbf{u}}_t$
10: Set $\boldsymbol{\theta} \leftarrow \mathrm{vech}(\mathbf{H})$ and update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mathcal{P}\boldsymbol{\zeta}_t\big(\mathcal{R}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) - \boldsymbol{\zeta}_t^\intercal\boldsymbol{\theta}\big)/(1 + \boldsymbol{\zeta}_t^\intercal\mathcal{P}\boldsymbol{\zeta}_t)$, $\mathcal{P} \leftarrow \mathcal{P} - \mathcal{P}\boldsymbol{\zeta}_t\boldsymbol{\zeta}_t^\intercal\mathcal{P}/(1 + \boldsymbol{\zeta}_t^\intercal\mathcal{P}\boldsymbol{\zeta}_t)$
11: Find $\mathbf{H}^+ = \mathrm{vech}^{-1}(\boldsymbol{\theta})$, update $\mathbf{H} \leftarrow \mathbf{H}^+$ and $t \leftarrow t+1$

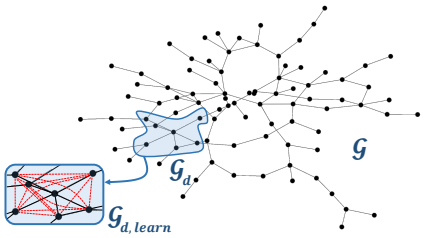


Fig. 1: Temporary communication links (red dashed lines) added to the subgraph $\mathcal{G}_d$ during the learning phase.

Inherent to Algorithm 1 is a policy iteration on $\mathcal{G}_{d,\mathrm{learn}}$ that finds K_k and L_k, intuitively representing "individual" and "cooperative" components of the policy at iteration k, respectively. Simultaneously, we also learn a stability margin τ_k related to these components. Then, even during the learning phase, we utilize these components together with the stability margin to design and update a stabilizing controller for the remainder of the agents in $\mathcal{G} \setminus \mathcal{G}_{d,\mathrm{learn}}$ (Line 9). Therefore, the entire network is also evolving during the learning phase. We do so by ensuring that information is unidirectionally exchanged from $\mathcal{G}_{d,\mathrm{learn}}$ to the rest of the network. As such, the policy of the agents in $\mathcal{G} \setminus \mathcal{G}_{d,\mathrm{learn}}$ is dependent on those in $\mathcal{G}_{d,\mathrm{learn}}$, but not vice versa.

At each iteration of the learning phase in Algorithm 1, we use a Recursive Least-Squares (RLS)-based recursion to evaluate the policy at hand by estimating the unknown parameter matrix at iteration k, referred to as $\widetilde{\mathbf{H}}_k$. This entire process is performed by the Subgraph Policy Evaluation (SPE) (Algorithm 2), given the graphs $\mathcal{G}$, $\mathcal{G}_{d,\mathrm{learn}}$, the mapping $\mathrm{policy}(\mathcal{V}_{\mathcal{G}_d})$, and the previous estimate of $\widetilde{\mathbf{H}}_{k-1}$. By construction, $\widetilde{\mathbf{H}}_k$ contains the required information to find K_k and L_k from data via recovering the block matrices X_1, X_2, Y_1, Y_2, Z_1, and Z_2 from $\widetilde{\mathbf{H}}_k$ as in Line 11. Such specific recovery of blocks from $\widetilde{\mathbf{H}}_k$ is due to the special matrix structure resulted from adding extra links to $\mathcal{G}_d$; see the proof of Theorem 2. Also, the matrix inversions on Line 12 is justified as a consequence of Proposition 1.

Remark 1. Adding temporary links within the subgraph $\mathcal{G}_d$ enables us to examine the all-to-all coupling amongst its nodes (agents). We also require K_1 to stabilize (1), however, the interested reader is referred to [20] for means of circumventing this requirement.

IV. Convergence and Stability

In this section, we provide the convergence and stability analysis of Algorithm 1. We defer the subtleties of the framework and detailed proofs to the extended version of this manuscript [13]. First, we study the structure and gain margins of each local controller and then establish the stability property of the proposed controller for the entire network throughout the learning process in Algorithm 1. Finally, we show the convergence of Algorithm 1 to a stabilizing suboptimal distributed controller.

We begin by introducing a linear group that plays an important role in our setup. First, for any integer $r \geq 2$, let us define the following linear subspace of $\mathbb{R}^{rn\times rn}$:

$$\mathrm{L}(r\times n,\mathbb{R}):=\Big\{\mathbf{N}_r \;\Big|\; \mathbf{N}_r=\mathrm{I}_r\otimes(A-B)+\mathbb{1}_r\mathbb{1}_r^\intercal\otimes B, \\ A,B\in\mathbb{R}^{n\times n}\Big\}.$$

Now, let us define the *patterned linear group* as follows

$$\mathrm{PL}(r\times n,\mathbb{R}):=\Big\{\mathbf{N}_r\in\mathrm{L}(r\times n,\mathbb{R})\cap\mathrm{GL}(rn,\mathbb{R}) \;\Big|\; \\ A\in\mathrm{GL}(n,\mathbb{R})\cap\mathbb{S}^n,\ B\in\mathbb{S}^n\Big\},$$

where $\mathbb{S}^n$ denotes the symmetric $n\times n$ matrices.

Proposition 1. *The patterned linear group* $\mathrm{PL}(r\times n,\mathbb{R})$ *is a subgroup of* $\mathrm{GL}(rn,\mathbb{R})$*, and* $\mathrm{L}(r\times n,\mathbb{R})$ *is a linear subspace of* $\mathbb{R}^{rn\times rn}$*, closed under matrix multiplication. Furthermore, for any Schur stable matrix* $A\in\mathrm{L}(r\times n,\mathbb{R})$*, let* P *denote the unique solution to the discrete-time Lyapunov equation for any* $0\prec Q\in\mathbb{S}^{rn}$*, i.e., the map induced by* $P=A^\intercal PA+Q$*. Then,* $P\in\mathrm{PL}(r\times n,\mathbb{R})$ *if and only if* $Q\in\mathrm{PL}(r\times n,\mathbb{R})$*.*

The fact that the patterned linear group is preserved under the Lyapunov equation is key to our analysis, facilitating efficient propagation of information in the proposed algorithm. Next, we demonstrate how a specific structure and stability of the controller for the subgraph $\mathcal{G}_{d,\mathrm{learn}}$, if initialized properly, can be preserved throughout Algorithm 1. The following standard assumption is required for the rest of the analysis.

Assumption 1. The initial controller K_1 is stabilizing for the controllable pair (A,B), and $\boldsymbol{e}_t$ in Algorithm 2 is such that $\mathrm{Policy}(\mathcal{V}_{\mathcal{G}'})|_t$ remains persistently exciting.

Proposition 2. *Define* $\widetilde{\mathbf{K}}_k\coloneqq\mathrm{I}_d\otimes\big(K_k-L_k\big)+\mathbb{1}\mathbb{1}^\intercal\otimes L_k$ *for all* $k\geq1$*, where* K_k *and* L_k *are obtained as in Algorithm 1. Under Assumption 1 and throughout the learning phase (for all* $k\geq1$*),* $\widetilde{\mathbf{K}}_k$ *is stabilizing for the system in* $\mathcal{G}_{d,\mathrm{learn}}$ *and* $\mathrm{Policy}_k(\mathcal{V}_{\mathcal{G}_{d,learn}})|_t=\widetilde{\mathbf{K}}_k\,\mathrm{State}(\mathcal{V}_{\mathcal{G}_{d,learn}})|_t$*, for all* t*. Furthermore,* $\Delta K_k\coloneqq K_k-L_k$ *stabilizes the dynamics of a single agent, i.e.,* $A+B\Delta K_k$ *is Schur stable.*

The structured policy proposed in Proposition 2 becomes very useful in designing a suboptimal distributed controller. This will become more clear after the following characterization of a particular gain margin.

Proposition 3. *At each iteration* $k\geq1$*, let* K_k*,* L_k *and* τ_k *be obtained via Algorithm 1. Then,* $A+B(K_k-\alpha L_k)$ *is Schur stable for all* α *satisfying* $|\alpha-1|\leq\tau_k$*.*

The stability margin τ_k guaranteed above is upper-bounded by the stability margin of the pair $(A+B(K_k-L_k),B)$. This implies that, if the original closed-loop system of each agent does not have a "good" stability margin, then τ_k must be small; thus, reducing the influence of its neighbors in its policy (Line 16 of Algorithm 1). As a result of Propositions 2 and 3, we can now utilize this local "model-free" gain margin to guarantee stability for the entire network during the learning phase as follows.

Theorem 1. *Suppose* K_k*,* L_k *and* τ_k *are defined as in Algorithm 1. Then, under Assumption 1, the control policy* $\mathrm{Policy}_k(\mathcal{V}_{\mathcal{G}})$ *designed during the learning phase (line 9), stabilizes the entire network* $\mathcal{G}$ *at each iteration for any choice of* $\mathcal{V}_{\mathcal{G}_d}$*.*

Proof. From Definition 1 (according to a consistent choice of labeling of the nodes so that the last d nodes are chosen as $\mathcal{G}_d$) and Proposition 2, the feedback policy in line 9 of Algorithm 1 can be cast in the compact form,

$$\mathrm{Policy}_k(\mathcal{V}_{\mathcal{G}})=\begin{bmatrix}\mathrm{Policy}_k(\mathcal{V}_{\mathcal{G}\setminus\mathcal{G}_{d,\mathrm{learn}}})\\ \mathrm{Policy}_k(\mathcal{V}_{\mathcal{G}_{d,\mathrm{learn}}})\end{bmatrix} \\ =\begin{bmatrix}\hat{\mathbf{K}}_{\mathcal{G}\setminus\mathcal{G}_d} & \hat{\mathbf{L}}_{\mathcal{G}\setminus\mathcal{G}_d}\\ \mathbf{0} & \widetilde{\mathbf{K}}_k\end{bmatrix}\begin{bmatrix}\mathrm{State}_k(\mathcal{V}_{\mathcal{G}\setminus\mathcal{G}_{d,\mathrm{learn}}})\\ \mathrm{State}_k(\mathcal{V}_{\mathcal{G}_{d,\mathrm{learn}}})\end{bmatrix}=:\hat{\mathbf{K}}_k\mathrm{State}_k(\mathcal{G}),$$

where $\hat{\mathbf{L}}_{\mathcal{G}\setminus\mathcal{G}_d}\coloneqq\frac{\tau_k}{d-1}[\mathcal{A}_{\mathcal{G}}]_{12}\otimes L_k$, $\widetilde{\mathbf{K}}_k\coloneqq\mathrm{I}_d\otimes(K_k-L_k)+\mathbb{1}_d\mathbb{1}_d^\intercal\otimes L_k$, $\hat{\mathbf{K}}_{\mathcal{G}\setminus\mathcal{G}_d}\coloneqq\mathrm{I}_{N-d}\otimes K_k-\Big(\mathrm{I}_{N-d}-\frac{\tau_k}{d-1}\mathcal{A}_{\mathcal{G}\setminus\mathcal{G}_d}\Big)\otimes L_k$, $\mathcal{A}_{\mathcal{G}}$ denotes the adjacency matrix of $\mathcal{G}$, and $[\mathcal{A}_{\mathcal{G}}]_{12}$ is its submatrix capturing the interconnection of $\mathcal{G}\setminus\mathcal{G}_d$ and $\mathcal{G}_d$:

$$\mathcal{A}_{\mathcal{G}}=\begin{bmatrix}\mathcal{A}_{\mathcal{G}\setminus\mathcal{G}_d} & [\mathcal{A}_{\mathcal{G}}]_{12}\\ * & \mathcal{A}_{\mathcal{G}_d}\end{bmatrix}.$$

Note that the structure of $\hat{\mathbf{K}}_k$ emanates from the fact that the information exchange is unidirectional during the learning phase. Now consider the closed-loop system of $\mathcal{G}$,

$$\hat{\mathbf{A}}_{\mathcal{G}}|_{cl}\coloneqq\hat{\mathbf{A}}+\hat{\mathbf{B}}\hat{\mathbf{K}}_k \\ =\begin{bmatrix}\mathrm{I}_{N-d}\otimes A+\big(\mathrm{I}_{N-d}\otimes B\big)\hat{\mathbf{K}}_{\mathcal{G}\setminus\mathcal{G}_d} & \big(\mathrm{I}_{N-d}\otimes B\big)\hat{\mathbf{L}}_{\mathcal{G}\setminus\mathcal{G}_d}\\ \mathbf{0} & \widetilde{\mathbf{A}}_{\widetilde{\mathbf{K}}_k}\end{bmatrix},$$

where $\widetilde{\mathbf{A}}_{\widetilde{\mathbf{K}}_k}\coloneqq\widetilde{\mathbf{A}}+\widetilde{\mathbf{B}}\widetilde{\mathbf{K}}_k$ is the closed-loop system of $\mathcal{G}_d$. Define $S=\mathrm{I}_{N-d}-\frac{\tau_k}{d-1}\mathcal{A}_{\mathcal{G}\setminus\mathcal{G}_d}$ and let J be the Jordan form of S according to the similarity transformation $\mathcal{T}\in\mathbb{R}^{(N-d)\times(N-d)}$ such that $\mathcal{T}S\mathcal{T}^{-1}=J$. Now consider the following similarity transformation of $\hat{\mathbf{A}}_{\mathcal{G}}|_{cl}$,

$$\begin{bmatrix}\mathcal{T}\otimes\mathrm{I}_n & \mathbf{0}\\ \mathbf{0} & \mathrm{I}_d\otimes\mathrm{I}_n\end{bmatrix}\Big(\hat{\mathbf{A}}_{\mathcal{G}}|_{cl}\Big)\begin{bmatrix}\mathcal{T}\otimes\mathrm{I}_n & \mathbf{0}\\ \mathbf{0} & \mathrm{I}_d\otimes\mathrm{I}_n\end{bmatrix}^{-1} \\ =\begin{bmatrix}\mathrm{I}_{N-d}\otimes\big(A+BK_k\big)-J\otimes BL_k & *\\ \mathbf{0} & \widetilde{\mathbf{A}}_{\widetilde{\mathbf{K}}_k}\end{bmatrix}.$$

Note that $\mathrm{I}_{N-d} \otimes \big(A + BK_k\big) - J \otimes BL_k$ is a block upper triangular matrix whose diagonal blocks are equal to $A + B(K_k - \lambda_i(S)L_k)$ for $i = 1, \cdots, N-d$. We already know from Proposition 2 that $\widetilde{\mathbf{A}}_{\widetilde{\mathbf{K}}_k}$ is Schur stable. Hence, in order to show that $\hat{\mathbf{A}}_{\mathcal{G}}|_{cl}$ is Schur stable, it suffices to show that $\rho\left(A + B(K - \lambda_i(S)L)\right) < 1$ for $i = 1, \cdots, N-d$. Recall that $|\lambda_i(\mathcal{A}_{\mathcal{G}})| \leq d_{\max}$ [21], thus by definition of S and the fact that $d_{\max} = d - 1$, we conclude that $|\lambda_i(S) - 1| \leq \tau_k$. The rest of the proof now follows directly from Proposition 3. □

The latter theorem establishes the fact that the proposed feedback mechanism stabilizes the entire network throughout the learning phase. This facilitates control of the agents outside of $\mathcal{G}_d$, while the learning process is in effect for the subgraph. Nonetheless, the practicality and suboptimality characteristics of the algorithm heavily depend on the convergence of the learning phase. This will be addressed in the following theorem.

Theorem 2. *Under Assumption 1, Algorithm 1 converges, i.e., $K_k \to K^*$, $L_k \to L^*$, and $\tau_k \to \tau^*$ as $k \to \infty$, where $\widetilde{\mathbf{K}}^* = \mathrm{I}_d \otimes \big(K^* - L^*\big) + \mathbb{1}\mathbb{1}^\intercal \otimes L^*$ is the optimal solution to the infinite-horizon state-feedback LQR problem with system parameters $\big(\widetilde{\mathbf{A}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{Q}}, \widetilde{\mathbf{R}}\big)$ defined as $\widetilde{\mathbf{A}} = \mathrm{I}_d \otimes A$, $\widetilde{\mathbf{B}} = \mathrm{I}_d \otimes B$, $\widetilde{\mathbf{Q}} = \mathrm{I}_d \otimes (Q_1 + dQ_2) - \mathbb{1}\mathbb{1}^\intercal \otimes Q_2$, and $\widetilde{\mathbf{R}} = \mathrm{I}_d \otimes R$.*

Proof. At iteration k of the learning phase in Algorithm 1, by Proposition 2, the stabilizing feedback policy $\widetilde{\mathbf{K}}$ for subgraph $\mathcal{G}_{d,\text{learn}}$ can be written in a compact form as $\widetilde{\mathbf{K}}_k \coloneqq \mathrm{I}_d \otimes \big(K_k - L_k\big) + \mathbb{1}\mathbb{1}^\intercal \otimes L_k$. Then, its associated cost matrix $\widetilde{\mathbf{P}}_k$ satisfies the following Lyapunov equation:

$$\widetilde{\mathbf{P}}_k = (\widetilde{\mathbf{A}} + \widetilde{\mathbf{B}}\widetilde{\mathbf{K}}_k)^\intercal \widetilde{\mathbf{P}}_k (\widetilde{\mathbf{A}} + \widetilde{\mathbf{B}}\widetilde{\mathbf{K}}_k) + \widetilde{\mathbf{Q}} + \widetilde{\mathbf{K}}_k^\intercal \widetilde{\mathbf{R}} \widetilde{\mathbf{K}}_k.$$

Furthermore, we can verify that $\widetilde{\mathbf{A}} + \widetilde{\mathbf{B}}\widetilde{\mathbf{K}}_k \in \mathrm{L}(d \times n, \mathbb{R})$ which is also Schur stable, and $\widetilde{\mathbf{Q}} + \widetilde{\mathbf{K}}_k^\intercal \widetilde{\mathbf{R}} \widetilde{\mathbf{K}}_k \in \mathrm{PL}(d \times n, \mathbb{R})$. Thus, by Proposition 1, we conclude that $\widetilde{\mathbf{P}}_k \in \mathrm{PL}(d \times n, \mathbb{R})$. Next, we consider the parameter matrix $\widetilde{\mathbf{H}}_k$ obtained from Algorithm 2 at iteration k. We claim that

$$[\widetilde{\mathbf{H}}_k]_{11} = \widetilde{\mathbf{Q}} + \widetilde{\mathbf{A}}^\intercal \widetilde{\mathbf{P}}_k \widetilde{\mathbf{A}}, \quad [\widetilde{\mathbf{H}}_k]_{21} = [\widetilde{\mathbf{H}}_k]_{12}^\intercal = \widetilde{\mathbf{B}}^\intercal \widetilde{\mathbf{P}}_k \widetilde{\mathbf{A}},$$
$$[\widetilde{\mathbf{H}}_k]_{22} = \widetilde{\mathbf{R}} + \widetilde{\mathbf{B}}^\intercal \widetilde{\mathbf{P}}_k \widetilde{\mathbf{B}},$$

and thus, again by Proposition 1, we conclude that $[\widetilde{\mathbf{H}}_k]_{11}, [\widetilde{\mathbf{H}}_k]_{22} \in \mathrm{PL}(d \times n, \mathbb{R})$ and $[\widetilde{\mathbf{H}}_k]_{21} \in \mathrm{L}(d \times n, \mathbb{R})$. Then, by unravelling the structure of matrices constructing $\widetilde{\mathbf{H}}_k$, the recovery of X_i, Y_i and Z_i for $i = 1, 2$ in Line 11 of Algorithm 1 corresponds to

$$X_1 = Q_1 + (d-1)Q_2 + A^\intercal P_1 A, \quad X_2 = -Q_2 + A^\intercal P_2 A,$$
$$Y_1 = R + B^\intercal P_1 B, \quad Y_2 = B^\intercal P_2 B,$$
$$Z_1 = B^\intercal P_1 A, \quad Z_2 = B^\intercal P_2 A,$$

where $\widetilde{\mathbf{P}}_k = \mathrm{I}_d \otimes \big(P_1 - P_2\big) + \mathbb{1}\mathbb{1}^\intercal \otimes P_2$. Finally, by definition of K_{k+1} and L_{k+1} in Line 12, one can validate that $\widetilde{\mathbf{K}}_{k+1} = -[\widetilde{\mathbf{H}}_k]_{22}^{-1}[\widetilde{\mathbf{H}}_k]_{21}$, which coincides with the policy iteration in the Hewer's algorithm [22] for the system in $\mathcal{G}_{d,\text{learn}}$. For the details of these validations and the rest of convergence argument we refer to [13]. □

As we remove the temporary links that were provided for the learning phase, the structure of the interconnections in $\mathcal{G}$ are returned to its original topology. Therefore, it is vital to provide stability guarantees after the learning phase in Algorithm 1 concludes and the temporary links have been removed. This is outlined in the following corollary.

Corollary 1. *If the learning phase of Algorithm 1 converges, then* $\mathrm{Policy}(\mathcal{V}_{\mathcal{G}})$ *(Line 16) stabilizes the system in* (2).

Proof. Compare the structure of policy in Line 9 for $\mathcal{V}_{\mathcal{G}\setminus\mathcal{G}_d}$ during the learning phase, with that of Line 16 for the entire $\mathcal{V}_{\mathcal{G}}$ after temporary links are removed. The proof then follows similar steps as that of Theorem 1. □

V. Simulation Results

In this section, we examine the performance and convergence of the algorithm proposed in §III. We present the results of two experiments using Algorithm 1 on homogeneous networks of agents with unknown and unstructured model uncertainties.

In the first experiment, we sample continuous-time system parameters of a single agent (A, B) from a zero-mean normal distribution with unit covariance such that $A \in \mathbb{R}^{5\times5}$ and $B \in \mathbb{R}^{5\times3}$. Then, we consider a path-graph of 10 agents and demonstrate how Algorithm 1 converges for these different instances of system parameters. In the second experiment, we use the dynamics of an aircraft with continuous-time system parameters (A, B) (as reported in Appendix F in [23]). We consider random d-regular graph topologies of different sizes. Then, we demonstrate the efficacy of Algorithm 1 by illustrating the cost associated with the proposed distributed controller as a function of nodes in the graph. In both experiments, the continuous-time system dynamics of a single agent is discretized with a sampling rate of $\Delta T = 0.1s$. The dimensions are $n = 5$ and $m = 3$ for each agent and we also set $Q_1 = 0.2Q_2 = \mathrm{I}_n$ and $R = \mathrm{I}_m$. We introduce an uncertainty into the system parameters as follows. Each agent follows a (completely unknown) LTI system dynamics similar to (1) with A replaced by $A + \Delta A$, where entries of ΔA are sampled from a normal distribution $\mathcal{N}(0, 0.05)$. We assume a random exploration signal sampled from a normal distribution $e_t \sim \mathcal{N}(0, \sigma^2)$ where the variance σ^2 is chosen accordingly for different input channels. Given any number of agents $N \geq 3$, the maximum degree of a path graph is $d_{\max} = 2$, and hence, $d = d_{\max} + 1 = 3$. Thus, we take the first three agents ($i = 1, 2, 3$) as the nodes of $\mathcal{G}_d$. Assuming the designer has only access to a stabilizing controller for the system with nominal parameters A and B, we set the initial stabilizing controller to be the LQR optimal controller with parameters (A, B, Q_1, R).

Finally, for assessing optimality, we report the trace of cost matrices, $\mathrm{Tr}(\hat{\mathbf{P}}_k)$, associated with the proposed distributed controller learned by our algorithm at iteration k. As the optimal distributed design is unknown, we compare these results against the optimal cost for the unconstrained LQR problem, $\mathrm{Tr}(\hat{\mathbf{P}}_{\text{LQR}})$, as the solution of the Algebraic Riccati Equation with parameters $(\hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{Q}}, \hat{\mathbf{R}})$. Note that

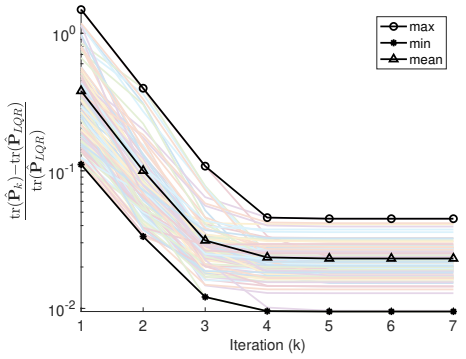
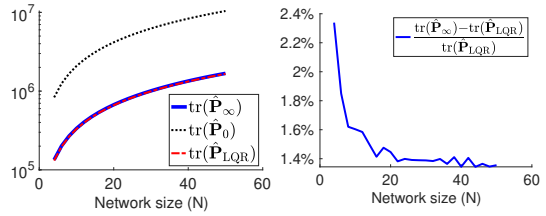


Fig. 2: Convergence of Algorithm 1 for 100 random system parameters (A,B).



(a) (b)

Fig. 3: Suboptimality of the distributed controller learned by Algorithm 1 for random 3-regular graphs of homogeneous aircraft with even number of nodes from 4 to 50.

this is an unfeasible solution to the problem in (3) but still provides a theoretical lowerbound for performance of any feasible solution (including the optimal one).

Figure 2 shows the results of the first experiment, which illustrates the normalized suboptimality error of the proposed controller for the entire network with respect to the (unfeasible) LQR controller. It illustrates the result of 100 random simulations for different system parameters and shows the progress of our method at each iteration in Algorithm 1. The actual simulations are plotted in faded color, whereas its statistical characteristics are plotted in solid black. Also note that almost all realizations have converged after 5 iterations of the learning phase, due to its quadratic convergence rate.

Figure 3 shows the results of the second experiment, illustrating how the cost of proposed controller is changing with respect to the number of nodes in random 3-regular graph topologies with even number of nodes. Figure 3a compares the cost associated with our design $\hat{\mathbf{P}}_\infty$ against the cost of initial controller $\hat{\mathbf{P}}_0$, and the (unfeasible) LQR controller $\hat{\mathbf{P}}_{\text{LQR}}$. Figure 3b illustrates the evolution of the normalized suboptimality of our proposed algorithm (with respect to the unfeasible LQR controller) as a function of number of nodes in the graphs.

VI. Conclusion

In this paper, we proposed a model-free distributed policy iteration algorithm to find structured controllers for large-scale networked systems from data. In this direction, we incorporated ideas from distributed control and data-driven optimal control literature to address the model-free distributed control problem. We then analyzed the structure of the distributed policy iteration algorithm and show how the data-driven nature of our algorithm can address unknown model uncertainty.

References

[1] F. Bullo, J. Cortes, and S. Martinez, *Distributed Control of Robotic Networks*, vol. 27. Princeton University Press, 2009.

[2] M. Mesbahi and F. Y. Hadaegh, "Formation flying control of multiple spacecraft via graphs, matrix inequalities, and switching," in *Proceedings of the 1999 IEEE International Conference on Control Applications*, vol. 2, pp. 1211–1216, IEEE, 1999.

[3] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, 2004.

[4] P. Massioni and M. Verhaegen, "Distributed control for identical dynamically coupled systems: A decomposition approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 124–135, 2009.

[5] F. Borrelli and T. Keviczky, "Distributed LQR design for identical dynamically decoupled systems," *IEEE Transactions on Automatic Control*, vol. 53, no. 8, pp. 1901–1912, 2008.

[6] K. Mårtensson and A. Rantzer, "Gradient methods for iterative distributed control synthesis," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 549–554, IEEE, 2009.

[7] W. Wang, F. Zhang, and C. Han, "Distributed linear quadratic regulator control for discrete-time multi-agent systems," *IET Control Theory & Applications*, vol. 11, no. 14, pp. 2279–2287, 2017.

[8] F. L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das, *Cooperative Control of Multi-agent Systems: optimal and adaptive design approaches*. Springer Science & Business Media, 2013.

[9] B. Bamieh, F. Paganini, and M. A. Dahleh, "Distributed control of spatially invariant systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 7, pp. 1091–1107, 2002.

[10] N. Motee and A. Jadbabaie, "Optimal control of spatially distributed systems," *IEEE Transactions on Automatic Control*, vol. 53, no. 7, pp. 1616–1629, 2008.

[11] A. Chapman, A. D. González, M. Mesbahi, L. Dueñas-Osorio, and R. M. D'Souza, "Data-guided control: Clustering, graph products, and decentralized control," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 493–498, 2017.

[12] Y. R. Sturz, A. Eichler, and R. S. Smith, "Distributed control design for heterogeneous interconnected systems," *IEEE Transactions on Automatic Control*, pp. 1–1, 2020.

[13] S. Alemzadeh, S. Talebi, and M. Mesbahi, "D3PI: Data-driven distributed policy iteration for homogeneous interconnected systems," *arXiv preprint arXiv:2103.11572*, 2021.

[14] P. Deshpande, P. P. Menon, C. Edwards, and I. Postlethwaite, "A distributed control law with guaranteed LQR cost for identical dynamically coupled linear systems," in *American Control Conference (ACC)*, pp. 5342–5347, IEEE, 2011.

[15] C. H. Papadimitriou and J. Tsitsiklis, "Intractable problems in control theory," *SIAM journal on control and optimization*, vol. 24, no. 4, pp. 639–654, 1986.

[16] V. Gupta, B. Hassibi, and R. M. Murray, "A sub-optimal algorithm to synthesize control laws for a network of dynamic agents," *International Journal of Control*, vol. 78, no. 16, pp. 1302–1313, 2005.

[17] M. Rotkowitz and S. Lall, "A characterization of convex problems in decentralized control," *IEEE transactions on Automatic Control*, vol. 50, no. 12, pp. 1984–1996, 2005.

[18] J. Bu, A. Mesbahi, M. Fazel, and M. Mesbahi, "LQR through the lens of first order methods: Discrete-time case," *arXiv preprint arXiv:1907.08921*, 2019.

[19] J. Bu, A. Mesbahi, and M. Mesbahi, "On topological properties of the set of stabilizing feedback gains," *IEEE Transactions on Automatic Control*, vol. 66, no. 2, pp. 730–744, 2021.

[20] S. Talebi, S. Alemzadeh, N. Rahimi, and M. Mesbahi, "Online regulation of unstable LTI systems from a single trajectory," *arXiv preprint arXiv:2006.00125*, 2020.

[21] B. Bollobás, *Modern Graph Theory*, vol. 184. Springer Science & Business Media, 2013.

[22] G. Hewer, "An iterative technique for the computation of the steady state gains for the discrete optimal regulator," *IEEE Transactions on Automatic Control*, vol. 16, no. 4, pp. 382–384, 1971.

[23] Y. S. Hung and A. MacFarlane, *Multivariable Feedback: A quasi-classical approach*. Springer-Verlag New York, Inc., 1982.