Policy Gradient with Self-Attention for Model-Free Distributed Nonlinear Multi-Agent Games

Eduardo Sebastián[1] Maitrayee Keskar[2] Eeman Iqbal[2]
Eduardo Montijano[3] Carlos Sagüés[3] Nikolay Atanasov[2]

***Abstract*— Multi-agent games in dynamic nonlinear settings are challenging due to the time-varying interactions among the agents and the non-stationarity of the (potential) Nash equilibria. In this paper we consider model-free games, where agent transitions and costs are observed without knowledge of the transition and cost functions that generate them. We propose a novel distributed policy structure that follows the communication constraints in multi-team games, with multiple agents per team, and learned through policy gradients. Our formulation is inspired by the structure of distributed policies in linear quadratic games, which take the form of time-varying linear feedback gains. In the nonlinear case, we model the policies as nonlinear feedback gains, parameterized by self-attention layers to account for the time-varying multi-agent communication topology. We demonstrate that our approach achieves strong performance in several settings, including distributed linear and nonlinear regulation, and simulated and real multi-robot pursuit-and-evasion games.**

I. Introduction

Multi-robot problems encompass a variety of expected behaviors [1]–[4], including cooperative, conflicting or competitive actions. For instance, in a perimeter-defense setting [5], [6], multiple teams must coordinate to effectively defend a region from potential attackers (Fig. 1). These kinds of problems can be formulated as multi-team dynamic games [7], [8], where each multi-agent team is viewed as a player with specific goals and constraints, and where agents interact with teammates (intra-team interactions) and agents on other teams (inter-team interactions). These settings are typically nonlinear and dynamic, requiring complex interactions that evolve with time as a function of how the agents play the game. These features are specially relevant when we seek distributed policies subject to the communication constraints imposed by the teams' topology; and in the absence of a mathematical description of the game dynamics and costs, demanding model-free approaches that only rely on transition and cost samples. Inspired by solutions in linear quadratic games, we present a novel distributed policy structure for nonlinear dynamic games learned via policy gradients.

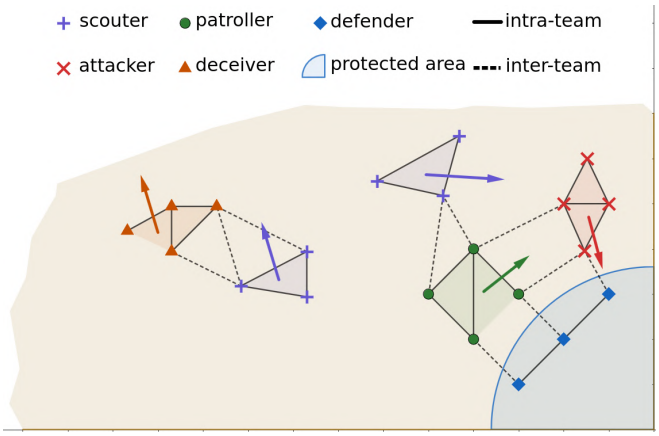


Fig. 1: Multi-team perimeter-defense game. Each team has a different goal, number of agents, and constraints (e.g., the scouter team inspects the region to detect attackers whereas the deceiver team aims at confusing the scouters), leading to global cooperative and competitive behaviors (e.g., patrollers and defenders help each other to prevent attackers from crossing the perimeter). The agents only have access to local information, either from teammates (intra-team interactions) or agents on other teams (inter-team interactions). In general, the problem is nonlinear and dynamic with time-varying interactions and unknown transition and cost model.

Related Work. Model-based methods for nonlinear multi-agent dynamic games rely on iterative linearization of the system dynamics and quadratic approximation of the game cost [9]. This allows for fast computation with guarantees of convergence to a saddle configuration [10] but imposes a centralized calculation that limits the applicability in distributed settings, where the agent communication is restricted according to a graph topology. To overcome such limitations, it is possible to restrict the class of nonlinear multi-agent dynamic games to potential games [11], [12], where it is assumed that a potential function exists such that the relative incentives in modifying one agent's policy is equal to the difference in value of the potential function. Under this constraint, algorithms can be derived that compute open-loop optimal trajectories for the agents under centralized [13] or distributed topological constraints [11]. However, open-loop policies lack robustness and require knowledge on how the multi-agent topology will evolve with time. In contrast, we propose a novel policy parameterization that is distributed by construction and does not require network topology prediction. In all previous cases, a model of the dynamics and the cost function is needed to compute the actions. Instead, to address

[1]Department of Computer Science and Technology, University of Cambridge, UK (e-mail: es2121@cam.ac.uk).
[2]Department of Electrical and Computer Engineering, University of California San Diego, USA (e-mails: {mmkeskar, eiqbal, natanasov}@ucsd.edu).
[3]RoPeRt group, at DIIS - I3A, Universidad de Zaragoza, Spain (e-mails: {emonti, csagues}@unizar.es).
This work has been supported by ONR N00014-23-1-2353, ONR Global grant N62909-24-1-2081, NSF CCF-2402689 (ExpandAI), in part by Spanish projects PID2024-159284NB-I00, PID2024-159279OB-I00, PID2021-124137OB-I00, and TED2021-130224B-I00 funded by MCIN/AEI/10.13039/501100011033, by ERDF A way of making Europe and by the European Union NextGenerationEU/PRTR, DGA T45-23R, a Spanish grant FPU19-05700 and a US-Spain Fulbright grant.

general nonlinear multi-agent dynamic games, we propose a model-free policy gradient approach that relies only on transition and cost samples.

Model-free solutions for games are limited due to the non-stationarity of the Nash equilibria (if one exists) [14]. Traditional approaches either focus on providing theoretical guarantees of convergence or addressing practical settings assuming the existence of such Nash equilibria. An instance of the former is [15], where distributed linear quadratic regulators are learned assuming that the sequence of graphs representing the communication structure of the game is known. From a different perspective, when the linear cost function is known and the strategies of all players are available, the problem can be posed as a multi-team distributed optimization program [16], [17]. In practical settings, existing solutions rely on multi-agent reinforcement learning algorithms [18] that consider independent heterogeneous agents to apply policy gradient methods [19], [20]. In this work, we bring together the benefits of both alternatives by proposing a self-attention-based policy parameterization built from first principles and which enforces distributed execution constraints. The distributed policy is trained using a policy gradient learning method that considers, simultaneously, the policies of all the agents from all different teams, addressing the non-stationarity of the game in practice.

Closer to our setup, attention mechanisms have been combined with policy gradients to aggregate neighbor observations in cooperative multi-robot navigation [21]. We differ in two key respects: (i) we address multi-team dynamic games with both intra- and inter-team (competitive) interactions and heterogeneous roles, rather than a single cooperative swarm; and (ii) we do not use attention as a generic observation encoder, but to parameterize a nonlinear feedback gain derived from the structure of optimal distributed LQR policies.

Contributions. Our main contribution is a method for learning distributed policies in model-free nonlinear multi-agent dynamic games (Sec. II). Our approach uses a nonlinear feedback gain formulation of the agent policies, parameterized using self-attention layers (Sec. III). The use of self-attention enables to enforce intra- and inter-team graph constraints, handling time-varying communication and achieving invariance with respect to the total number of agents. Furthermore, a neural network parameterization of the policies motivates the use of a policy gradient method to learn in the model-free setting. The method also allows to learn heterogeneous policies per team, such that the teams adjust to specific goals. We demonstrate that our method applies broadly, from linear quadratic settings under topology constraints to multi-agent reinforcement learning in competitive games with simulated and real robots (Sec. IV).

II. Problem Statement

We consider a discrete-time game among N multi-agent teams. Each team has M_i agents, where index i refers to a specific team. The total number of agents is $\mathsf{M} = \sum_{i=1}^{\mathsf{N}} \mathsf{M}_i$. The agents interact in a distributed manner, described by a time-varying directed graph $\mathcal{G}(k) = (\mathcal{V}, \mathcal{E}(k))$. In graph $\mathcal{G}(k)$, k refers to the discrete time step and $\mathcal{V}$ is the set of agents, such that $|\mathcal{V}| = \mathsf{M}$. We denote by $\mathcal{V}_i$ the set of agents of team i, such that $|\mathcal{V}_i| = \mathsf{M}_i$, $\mathcal{V} = \bigcup_{i=1}^{\mathsf{N}} \mathcal{V}_i$ and $\bigcap_{i=1}^{\mathsf{N}} \mathcal{V}_i = \emptyset$. The set of edges is $\mathcal{E}(k)$ and $(l,p) \in \mathcal{E}(k)$ if and only if the information of agent p is available to agent l at instant k. We allow intra-team and inter-team interactions, defined by edge $(l,p) \in \mathcal{E}(k)$ with $l,p \in \mathcal{V}_i$, and edge $(l,p) \in \mathcal{E}(k)$ with $l \in \mathcal{V}_i$ and $p \in \mathcal{V}_j$. We define the intra-neighbors as $\mathcal{N}_i^l(k) = \{p \in \mathcal{V}_i | (l,p) \in \mathcal{E}(k)\}$, the inter-neighbors as $\mathcal{N}_{-i}^l(k) = \{p \in \bigcup_{i' \neq i} \mathcal{V}_{i'} | (l,p) \in \mathcal{E}(k)\}$, and $\mathcal{N}^l(k) = \mathcal{N}_i^l(k) \cup \mathcal{N}_{-i}^l(k)$.

Each agent l in team i is characterized by a state vector $\mathbf{x}_i^l(k)$. The collection of states of team i is given by $\mathbf{x}_i(k) = [(\mathbf{x}_i^1(k))^\top, \dots, (\mathbf{x}_i^{\mathsf{M}_i}(k))^\top]^\top$, and the collection of states of all the teams except team i is

$$\mathbf{x}_{-i}(k) = [(\mathbf{x}_1(k))^\top, \dots, (\mathbf{x}_{i-1}(k))^\top, (\mathbf{x}_{i+1}(k))^\top, \dots, (\mathbf{x}_{\mathsf{N}}(k))^\top]^\top.$$

Each agent acts in the environment by means of an action vector $\mathbf{u}_i^l(k)$. As with $\mathbf{x}_i(k)$ and $\mathbf{x}_{-i}(k)$, the collection of actions of team i is $\mathbf{u}_i(k)$, and the collection of actions of all teams except team i is $\mathbf{u}_{-i}(k)$. The dynamics of the game are nonlinear and unknown to the agents. Agents choose their actions based on a policy defined as

$$\mathbf{u}_i^l(k) = \pi_i^l(\mathbf{x}_{\mathcal{N}^l}^l(k), \boldsymbol{\theta}_i^l). \tag{1}$$

In equation (1), $\mathbf{x}_{\mathcal{N}^l}^l(k)$ refers to the state information of the intra- and inter-neighbors of agent l, and $\boldsymbol{\theta}_i^l$ are parameters that modulate the policy. We consider heterogeneous policies across teammates and teams. The collection of policies of team i is denoted as π_i, with parameters $\boldsymbol{\theta}_i$.

The goal of the teams is to find a sequence of actions that minimize their respective infinite-horizon cost

$$\min_{\mathbf{u}_i(k), 0 \leq k < \infty} \sum_{k=0}^{\infty} \gamma c_i(\mathbf{x}_i(k), \mathbf{u}_i(k), \mathbf{x}_{-i}(k), \mathbf{u}_{-i}(k)), \tag{2}$$

where $0 < \gamma < 1$ is a discount factor. The *stage* cost of team i, $c_i(\bullet)$, is bounded and depends on the states and actions of all the teams of the game at instant k. Since we consider an infinite-horizon game, we formulate the task of minimizing (2) as finding policies for the agents that follow $\mathcal{G}(k)$. The definition of the policy in equation (1) leads to a reformulation of (2) in terms of $\pi_i(\bullet)$:

$$J_i^\infty(\mathbf{x}_i(0), \mathbf{x}_{-i}(0), \pi_i(\bullet, \boldsymbol{\theta}_i), \{\pi_j(\bullet)\}_{j=1, j\neq i}^{\mathsf{N}}) = \min_{\boldsymbol{\theta}_i} \sum_{k=0}^{\infty} c_i(\mathbf{x}_i(k), \pi_i(\bullet), \mathbf{x}_{-i}(k), \{\pi_j(\bullet)\}_{j=1, j\neq i}^{\mathsf{N}}). \tag{3}$$

III. Multi-team Distributed Policy Parameterization

In the absence of a known dynamic model and cost structure, our goal is to learn $\boldsymbol{\theta}_i \ \forall i$ leveraging the only signal available: the cost signal. Therefore, the alternative is to apply a policy gradient method, i.e., use the derivative of $J_i^\infty(\bullet)$ with respect to $\boldsymbol{\theta}_i$ to update $\pi_i(\bullet)$. A policy gradient approach for model-free nonlinear multi-agent dynamic games in distributed settings requires a parameterization of $\pi_i(\bullet)$

that supports the graph constraints and is able to integrate information from intra- and inter-neighbors. We propose a parameterization for $\pi_i(\bullet)$ that satisfies this requirement, together with a policy gradient algorithm for learning the policy parameters when only a cost signal and a set of transition tuples $(\mathbf{x}_i(k), \mathbf{u}_i(k), \mathbf{x}_i(k+1))$ is available.

A. Optimal distributed policies in linear quadratic games

Let $\Pi(\mathbf{x}(k), \boldsymbol{\theta}) = [\pi_1(\bullet)^\top, \dots, \pi_{\mathsf{N}}(\bullet)^\top]^\top$ be the collection of all the team policies, where $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{\mathsf{N}}]^\top$ is the collection of all parameters and $\mathbf{x} = [\mathbf{x}_1, \dots, \mathbf{x}_{\mathsf{N}}]^\top$ is the collection of states of all teams. Model-based infinite games with known linear dynamics, known quadratic cost, and a fixed fully connected communication topology are optimally solved using a linear feedback gain

$$\Pi(\mathbf{x}(k), \boldsymbol{\theta}^*) = -\mathbf{K}^* \mathbf{x}(k), \tag{4}$$

where $\boldsymbol{\theta}^* = \mathbf{K}^*$ is the solution of the discrete-time algebraic Ricatti equation of appropriate dimension [22]. The first challenge comes when the system dynamics are unknown and subject to topological constraints that evolve with time. This is observed, e.g., in [15], where the model-free learning of the globally optimum linear quadratic regulator in finite-horizon games with time-varying graph constraints is proven.

Theorem 1 (adapted from [15])**.** *Assume unknown linear system dynamics, a quadratic game cost that admits an appropriate compact sub-level set, and a known* finite-horizon *sequence of topological constraints given by* $\{\mathcal{G}(k)\}_{k=0}^{\mathsf{K}}$ *for some finite-time horizon* K*. Furthermore, assume that the cost gradient is locally dominant, i.e.,* $\mu(J(\Pi(\mathbf{x}(k), \boldsymbol{\theta})) - J^*) \leq ||\nabla J(\Pi(\mathbf{x}(k), \boldsymbol{\theta}))||_2^2$ *for all* $\Pi(\bullet)$ *and some* $\mu > 0$*, where* J^* *is the global minimum of the game and* $J(\Pi(\mathbf{x}(k), \boldsymbol{\theta}^*))$ *is the cost associated to the learned distributed policy. Then, there exist constants* $\delta > 0$ *and* $\epsilon > 0$ *such that a distributed policy can be learned, through policy gradient, that satisfies* $J(\Pi(\mathbf{x}(k), \boldsymbol{\theta}^*)) - J^* < \epsilon$ *with probability* $1 - \delta$*.*

Theorem 1 sheds light on three key aspects. First, it is possible to learn an optimal distributed policy under time-varying topological constraints in finite-horizon games. In finite-horizon games, the policy in Eq. (4) is replaced by

$$\Pi(\mathbf{x}(k), \boldsymbol{\theta}^*) = -\mathbf{K}^*(k) \mathbf{x}(k), \tag{5}$$

where the optimal gain is now a sequence of optimal linear feedback gains such that $\boldsymbol{\theta}^* = \{\mathbf{K}^*(k)\}_{k=0}^{\mathsf{K}}$ for some finite-time horizon K. Second, in the absence of a known dynamics model, Theorem 1 requires the evolution of the team graph to be known a priori, which is a limitation in general multi-agent games. Finally, Theorem 1 assumes linear dynamics with quadratic cost. Overall, Theorem 1 is a motivating result to seek optimal distributed policies for general multi-agent dynamic games through learning, but key challenges must be addressed to derive a solution for general games.

B. Optimal distributed policies as nonlinear feedback gains

It is not possible to extend the controller learned using the result in Theorem 1 to an infinite-horizon problem directly because the approach requires a priori knowledge of the graph connectivity $\{\mathcal{E}(k)\}_k$ and an infinite sequence of control gains $\{\mathbf{K}^*(k)\}_k$, neither of which is possible without additional restrictive assumptions such as periodicity. Therefore, we present two contributions in this regard: (1) our policy parameterization allows time-varying communication in an infinite-horizon problem and (2) extends the formulation in Theorem 1 to a nonlinear game.

Specifically, we propose a nonlinear policy of the form

$$\Pi(\mathbf{x}(k), \boldsymbol{\theta}) = -\mathbf{K}(\mathbf{x}(k), \boldsymbol{\theta}) \mathbf{x}(k), \tag{6}$$

where $\mathbf{K}(\mathbf{x}(k), \boldsymbol{\theta})$ is a nonlinear function that outputs a feedback gain at each instant and that is parameterized by $\boldsymbol{\theta}$. This formulation is a combination of Eqs. (4) and (5), accounting for the state-dependency and the infinite-horizon nature of the game, since $\mathbf{K}(\mathbf{x}(k), \boldsymbol{\theta})$ can be called infinite times. On the other hand, $\boldsymbol{\theta}$ are parameters to be learned from a general cost signal in order to overcome the need of a known (linear) system model and (quadratic) cost. However, it is yet to be specified how to make $\mathbf{K}(\mathbf{x}(k), \boldsymbol{\theta})$ distributed.

Elaborating on $\mathbf{K}(\mathbf{x}(k), \boldsymbol{\theta})$, Eq. (6) is equivalent to

$$\Pi(\mathbf{x}(k), \boldsymbol{\theta}) = \begin{pmatrix} \mathbf{k}_{1,1}^{1,1}(\mathbf{x}(k), \boldsymbol{\theta}) & \dots & \mathbf{k}_{1,\mathsf{N}}^{1,\mathsf{M}_\mathsf{N}}(\mathbf{x}(k), \boldsymbol{\theta}) \\ \mathbf{k}_{1,1}^{2,1}(\mathbf{x}(k), \boldsymbol{\theta}) & \dots & \mathbf{k}_{1,\mathsf{N}}^{2,\mathsf{M}_\mathsf{N}}(\mathbf{x}(k), \boldsymbol{\theta}) \\ \vdots & \vdots & \vdots \\ \mathbf{k}_{\mathsf{N},1}^{\mathsf{M}_\mathsf{N},1}(\mathbf{x}(k), \boldsymbol{\theta}) & \dots & \mathbf{k}_{\mathsf{N},\mathsf{N}}^{\mathsf{M}_\mathsf{N},\mathsf{M}_\mathsf{N}}(\mathbf{x}(k), \boldsymbol{\theta}) \end{pmatrix} \mathbf{x}(k), \tag{7}$$

where $\mathbf{k}_{i,j}^{l,p}(\mathbf{x}(k), \boldsymbol{\theta})$ is the block-element matrix that relates agent l of team i with agent p of team j. To impose the scalability constraints on the policy, we first approximate $\mathbf{k}_{i,j}^{l,p}(\mathbf{x}(k), \boldsymbol{\theta})$ by $\mathbf{k}_{i,j}^{l,p}(\mathbf{x}(k), \boldsymbol{\theta}_i^l)$, such that each agent has a different set of parameters associated to its role. Next, we enforce $\mathbf{k}_{i,j}^{l,p}(\mathbf{x}(k), \boldsymbol{\theta}_i^l) = \mathbf{0}$ if $(l,p) \notin \mathcal{E}(k)$ to avoid infeasible propagation of information from non-neighboring agents. However, there is still leakage from non-neighboring information in the dependency of $\mathbf{k}_{i,j}^{l,p}(\mathbf{x}(k), \boldsymbol{\theta}_i^l)$ on $\mathbf{x}(k)$. To address this, we approximate $\mathbf{k}_{i,j}^{l,p}(\mathbf{x}(k), \boldsymbol{\theta}_i^l)$ by $\mathbf{k}_{i,j}^{l,p}(\mathbf{x}_{\mathcal{N}^l}^l(k), \boldsymbol{\theta}_i^l)$. Therefore, Eq. (7) is approximated such that the action of an agent at time k is given by

$$\mathbf{u}_i^l(k) = \sum_{l'=1}^{|\mathcal{N}_i^l(k)|} \mathbf{k}_{i,i}^{l,l'}(\mathbf{x}_{\mathcal{N}^l}^l(k), \boldsymbol{\theta}_i^l) \mathbf{x}_i^{l'} + \sum_{p=1,j=1}^{|\mathcal{N}_{-i}^l(k)|} \mathbf{k}_{i,j}^{l,p}(\mathbf{x}_{\mathcal{N}^l}^l(k), \boldsymbol{\theta}_i^l) \mathbf{x}_j^p. \tag{8}$$

Note that the execution of (8) is fully distributed. To parameterize $\mathbf{k}_{i,j}^{l,p}(\mathbf{x}_{\mathcal{N}^l}^l(k), \boldsymbol{\theta}_i^l)$, it is important to note that the number of neighbors changes with time. Thus, we propose to model $\mathbf{k}_{i,j}^{l,p}(\mathbf{x}_{\mathcal{N}^l}^l(k), \boldsymbol{\theta}_i^l)$ using self-attention networks [23]. Specifically, the implementation of the expression in (8) is as follows. The proposed architecture is composed by a sequence of layers, indexed using the subscript $w = 1, \dots, \mathsf{W}$, that

perform the following operations:

$$\begin{aligned}
&\mathbf{Q}^i_w = \mathbf{A}^i_w \mathbf{X}^i_{w-1},\ \mathbf{K}^i_w = \mathbf{B}^i_w \mathbf{X}^i_{w-1},\ \mathbf{V}^i_w = \mathbf{C}^i_w \mathbf{X}^i_{w-1} \\
&\mathbf{Y}^i_w = \chi\left(\mathsf{softmax}\left(\beta(\mathbf{Q}^i_w)\beta((\mathbf{K}^i_w)^\top)\right)\beta(\mathbf{V}^i_w)\right), \\
&\mathbf{X}^i_w = \psi(\mathbf{D}^i_w \mathbf{Y}^i_w),
\end{aligned} \tag{9}$$

where $\beta(\bullet)$, $\chi(\bullet)$, and $\psi(\bullet)$ are nonlinear activation functions. On the other hand, $\boldsymbol{\theta}_i = \{\mathbf{A}^i_w, \mathbf{B}^i_w, \mathbf{C}^i_w, \mathbf{D}^i_w\}^{\mathsf{W}}_{w=1}$ are matrices to be learned and of fixed size, regardless of the number of neighbors. To match (8) and (9), the dimensions of the matrices of the last layer are such that $\mathbf{X}^i_{\mathsf{W}}$ has $U \cdot X$ rows and $|\mathcal{N}^l|$ columns, with U and X the dimensions of the action and state space of the agents. This is because each column of $\mathbf{X}^i_{\mathsf{W}}$ corresponds to the block-element $\mathbf{k}^{l,p}_{i,j}(\mathbf{x}^l_{\mathcal{N}^l}(k), \boldsymbol{\theta}^l_i)$. Therefore, after a reshaping operation, the robot has all the block-elements $\mathbf{k}^{l,p}_{i,j}(\mathbf{x}^l_{\mathcal{N}^l}(k), \boldsymbol{\theta}^l_i)$ and state information to compute (8).

The use of an attention mechanism to learn the nonlinear feedback gains is dictated by the need of handling time-varying teams of agents. Specifically, attention mechanism are favored by graph topologies induced by proximity, as it is common in multi-robot teams, since spatial proximity correlates with the learned attention weights. In terms of the impact of attention in the stability of the control policies, one can easily enforce additional constraints in $\mathbf{k}^{l,p}_{i,j}(\mathbf{x}^l_{\mathcal{N}^l}(k), \boldsymbol{\theta}^l_i)$ to ensure positive definiteness of $\mathbf{K}(\mathbf{x}(k), \boldsymbol{\theta})$, as it is done in [4], [15]. On the other hand, the self-attention mechanism allows parameter sharing across agents from the same team, which further enhances sample efficiency during training.

The parameters $\boldsymbol{\theta}_i\ \forall i \in \{1, \dots, \mathsf{N}\}$ are unknown. We propose a policy gradient method to learn the optimal parameters $\{\boldsymbol{\theta}^*_i\}^{\mathsf{N}}_{i=1}$. In the case of games with known cost and dynamics, one can find a quadratic approximation to compute the discrete-time Ricatti equation. In model-free cases under unknown cost, only the sequence of costs per team $\{J^\infty_i(\bullet)\}^{\mathsf{N}}_{i=1}$. Algorithm 1 describes the policy gradient training procedure to learn optimal parameters $\{\boldsymbol{\theta}^*_i\}^{\mathsf{N}}_{i=1}$. At each iteration $t \in \{1, \dots, \mathsf{T}\}$, the parameters $\boldsymbol{\theta}^t_i$ change as a function of the evolution of the cost signal obtained by running game roll-outs. The game is first initialized with random initial conditions $\mathbf{x}(0)$. Then, given $\{\boldsymbol{\theta}^t_i\}^{\mathsf{N}}_{i=1}$, the game is played, using the policy in (8). Notably, Algorithm 1 is distributed itself, as the optimization steps can be distributed across teams. Furthermore, to favor exploration, the deterministic policy is embedded in an unbiased stochastic structure, leading to a policy $\rho^l_i(\mathbf{x}^l_{\mathcal{N}^l}(k), \boldsymbol{\theta}^l_i)$ such that

$$\begin{aligned}
\mathbb{E}[\rho^l_i(\mathbf{x}^l_{\mathcal{N}^l}(k), \boldsymbol{\theta}^l_i)] &= \pi^l_i(\rho_i(\mathbf{x}^l_{\mathcal{N}^l}(k), \boldsymbol{\theta}^l_i)), \\
\mathbb{E}[\rho^l_i(\bullet)\rho^l_i(\bullet)^\top] &= \psi^l_i(\mathbf{x}^l_{\mathcal{N}^l}(k), \boldsymbol{\alpha}^l_i),
\end{aligned} \tag{10}$$

with $\mathbb{E}[\bullet]$ the expectancy operator and $\psi^l_i(\bullet)$ neural approximators with learned parameters $\boldsymbol{\alpha}^l_i$. The actions during training are sampled from the distributions $\rho^l_i(\bullet)$, whereas the deterministic policies $\pi^l_i(\bullet)$ are used at deployment time. Then, the cost signals $c_i(\bullet)$ are collected, the gradients of the parameters with respect to the cost signals $\partial J^\infty_i(\bullet)/\partial \boldsymbol{\theta}^t_i$ are computed, and the parameters are updated using gradient descent with learning rate $\eta > 0$. In this sense, we recall that the cost $J^\infty_i(\bullet)$ is assumed bounded, ensuring finite gradients. In practice, we use a multi-agent version of proximal policy optimization (PPO) [24] that implements Algorithm 1 and has been proved to be effective in computing the gradients $\partial J^\infty_i(\bullet)/\partial \boldsymbol{\theta}^t_i$ and stabilize training, although other centralized, distributed, or value-decomposition multi-agent reinforcement learning algorithms such as QMIX, COMA or DARL1N [25]–[27] can be employed. However, we remark that the focus of the paper is on the policy design, and not on the implementation of the policy gradient algorithm.

Algorithm 1 Policy Gradient with Self-Attention for Model-Free Distributed Nonlinear Multi-Agent Games

1: **Parameters**: initial parameters $\boldsymbol{\theta}^0_i\ \forall i \in \{1, \dots, \mathsf{N}\}$, learning rate $\eta > 0$, training steps $\mathsf{T} > 0$, time horizon $\mathsf{K} > 0$
2: **for** iteration $t = 1$ to $t = \mathsf{T}$ **do**
3: Roll out game with random initial conditions $\mathbf{x}(0)$ under policies $\rho^l_i(\bullet)$ in (10), parameterized as in (8)
4: Get cost signals $\{c_i(\bullet)\}^{\mathsf{K}}_{k=0}$
5: Compute parameter gradients $\partial J^\infty_i(\bullet)/\partial \boldsymbol{\theta}^t_i$
6: Update parameters $\boldsymbol{\theta}^{t+1}_i = \boldsymbol{\theta}^t_i - \eta \dfrac{\partial J^\infty_i(\bullet)}{\partial \boldsymbol{\theta}^t_i}$
7: **end for**
8: Set $\boldsymbol{\theta}^*_i = \boldsymbol{\theta}^{\mathsf{T}}_i\ \forall i \in \{1, \dots, \mathsf{N}\}$

IV. Results

We evaluate our multi-team distributed policy parameterization in different game settings of increasing complexity. In Sec. IV-A, we validate the optimality gap of our policy model in a distributed linear quadratic regulation setting, where an optimal LQR baseline exists. Next, in Sec. IV-B, we compare our policy in a nonlinear dynamic multi-agent game against an existing distributed optimal method under known transition and cost models, allowing us to study the optimality gap between model-free and model-based approaches. In Sec. IV-C, we evaluate our policy in a two-team pursuit-evasion game, comparing performance against other existing policy parameterizations. Finally, Sec. IV-D assesses the effectiveness of our learned policies in a real robot deployment.

A. Distributed linear quadratic regulation

The first experiment considers a linear quadratic regulation game under network constraints. We set $\mathsf{N} = 5$, $\mathsf{M}_i = 1$ for all i and consider unknown linear system dynamics:

$$x_i(k) = x_i(k-1) + b_i u_i(k-1) + \omega(k) \quad \forall i \in \mathcal{V}$$

with $x_i(k), u_i(k)$ scalars, $\omega(k)$ uniformly distributed in the interval $[-10^{-3}, 10^{-3}]$ as in [15], b_i uniformly random in $[0, 1]$, and $x_i(0)$ uniformly distributed in the range $[-0.1, 0.1]$ for all agents. The cost is $J^\infty_i(\bullet) = \sum^{\mathsf{K}}_{k=0} \mathbb{E}\big[(\mathbf{x}(k))^\top \mathbf{M}(k)\mathbf{x}(k) + (\mathbf{u}(k))^\top \mathbf{R}(k)\mathbf{u}(k)\big]$, with $\mathsf{K} = 30$ and $\mathbf{M}(k), \mathbf{R}(k)$ random positive definite matrices with eigenvalues uniformly distributed in the interval $[0.2, 1.0]$. The set of edges $\mathcal{E}(k)$ is randomly generated as a sparse graph for all time steps.

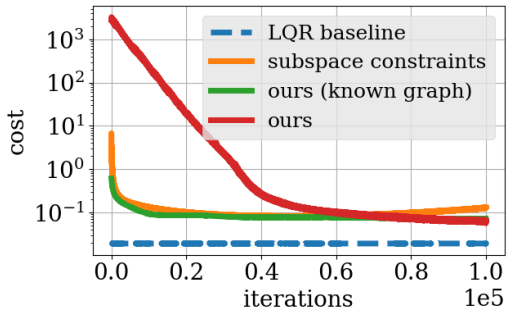


Fig. 2: Comparison of our method with a centralized LQR baseline, a model-free zeroth-order optimal LQR under subspace constraints [15], and our method with known graph constraints.

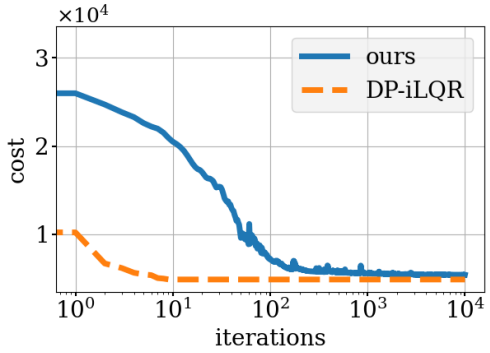


Fig. 3: Comparison of our method with the Distributed Potential iterative Linear Quadratic Regulator (DP-iLQR) [11].

We compare our method against three baselines. The first one computes the centralized linear quadratic regulator (LQR) for finite-horizon games, which is optimal when the network topology is known. The second is the method proposed in [15], which learns the optimal linear quadratic regulator under known subspace (network) constraints, using a zeroth-order approach. The third one is our method but assuming having access to the graph topology and with $\eta = 0.00001$ as learning rate. The fourth method is ours, the same as the previous but does not assume known network topology and instead considers a proximity graph such that $(i,j) \in \mathcal{E}(k)$ if and only if $|x_i(k) - x_j(k)| < 0.2$. For the last three methods, we enforce sparsity on the cost by replacing $\mathbf{M}(k)$ by $\mathbf{M}(k) \odot \mathbf{W}(k)$ and $\mathbf{R}(k)$ by $\mathbf{R}(k) \odot \mathbf{W}(k)$, with $\odot$ the element-wise matrix multiplication and $\mathbf{W}(k)$ the adjacency matrix of $\mathcal{G}(k)$.

In Fig. 2, we can observe that there is an optimality gap between the centralized baseline and the three distributed graph-constrained methods since the pay-off associated to all interactions among the agents is unknown to them. When the graph topology is known at all times, both the method in [15] and ours quickly converge to the optimum under the network constraints, aligned with the theoretical results in [15]. Training of [15] becomes unstable after 60000 iterations due to the zeroth-order optimization, which uses noisy gradient approximations. Lastly, our policy gradient method with unknown graph topologies is able to convergence to the same cost as the one obtained under the known topology assumption. These results show that there exists an optimality gap between distributed and centralized method and that our method is able to recover the performance under network constraints in multi-agent linear quadratic games.

B. Multi-agent nonlinear games

Next, we evaluate our method in a multi-agent game with nonlinear dynamics and non-quadratic cost. In particular, we reproduce the multi-agent navigation game from [28], where a set of N robots navigate in a 2D environment toward assigned locations while avoiding collisions with the other robots. As baseline, we use the Distributed Potential iterative Linear Quadratic Regulator (DP-iLQR) proposed in [11]. Given known nonlinear cost and nonlinear multi-agent dynamics, DP-iLQR finds optimal open-loop trajectories for nonlinear multi-agent games that a admit formulation as a potential game. To match the specifications of the baseline, the game runs for $\mathsf{K} = 100$ steps with $\mathsf{M}_i = 1\ \forall i \in \{1, \dots, \mathsf{N} = 7\}$, and the robots follow unicycle dynamics. The cost $J_i^{\infty}(\bullet)$ is

$$J_i^{\infty}(\bullet) = \sum_{k=0}^{\mathsf{K}} (\mathbf{x}_i(k) - \bar{\mathbf{x}}_i(k))^{\top} \mathbf{M}_i(k) (\mathbf{x}_i(k) - \bar{\mathbf{x}}_i(k)) + \sum_{k=0}^{\mathsf{K}-1} (\mathbf{u}(k))^{\top} \mathbf{R}_i(k) \mathbf{u}(k) + \sum_{j=1}^{\mathsf{N}} C(d_{i,j}(k)),$$

with reference states $\{\bar{\mathbf{x}}_i(k)\}_{k=0}^{\mathsf{K}}$, $d_{i,j}(k) = \|\mathbf{x}_i(k) - \mathbf{x}_j(k)\|$,

$$C(d_{i,j}(k)) = \begin{cases} \beta(d_{i,j}(k) - d_{\text{prox}}(k))^2 & d_{i,j}(k) < d_{\text{prox}}(k) \\ 0 & \text{otherwise} \end{cases}.$$

We set $\mathbf{M}_i(k) = \mathbf{I}$, $\mathbf{R}_i(k) = \mathbf{I}$ for all k except K, with $\mathbf{M}_i(\mathsf{K}) = 100\mathbf{I}$. We use Algorithm 1 with $\eta = 0.001$ and communication radius $r_{\text{comm}} = d_{\text{prox}} = 0.5$ m.

As observed in Fig. 3, our policy parameterization achieves near-optimal performance in 100 training iterations, compared to the 10 optimization iterations of DP-iLQR. In contrast with the baseline, our solution operates with unknown cost and dynamics, relying only on samples. Qualitatively, in Fig. 4, we can observe that the differences between the near-optimal performance of our policy and the baseline come from the initial transient behavior. However, in terms of collision avoidance and goal-reaching performance, both methods obtain comparable results, despite the differences in the available information. We also test scalability of our policy across number of agents, $\mathsf{N} = \{4, 8, 10, 15\}$. Fig. 5 shows that trajectories arrive to their goals and maintain smoothness for all numbers of agents, with a similar behavior as with 7 agents in Fig. 3. Collision avoidance constraint violation increases with the number of agents, as density of agents around goals increases. We also run evaluations over 20 random instantiations of the game, for each number of agents. We compute the mean and standard deviation of J_i^{∞}, normalized by the arena size because the spawning area enlarges with the number of agents to ensure agents do not collide at time 0. As Table I shows, normalized cost remains similar across team sizes. We leave a more systematic scalability analysis as future work.

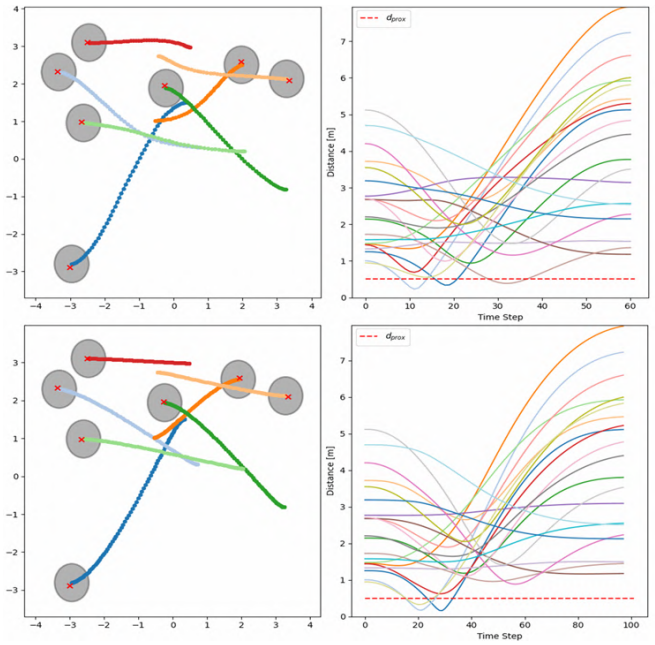


Fig. 4: Comparison between DP-iLQR at iteration 10 (**top row**) and our method at iteration 100 (**bottom row**). The agents are shown as grey circles, the goals as red crosses, and the agent trajectories as colored curves in the two pannels. The trajectory evolution (**left**) and temporal evolution (**right**) of the agents show that both methods achieve similar position and velocity profiles, as well as security margins with respect to d_{prox}.

C. Pursuit and evasion in BenchMARL

Next, we study the performance of our policy model in a pursuit-and-evasion game, Simple Tag, which was introduced in the Multi Particle Environment (MPE) [29], is implemented in the Vectorized Multi Agent Simulator (VMAS) [30] and BenchMARL [31], a multi-agent reinforcement learning suite for training and benchmarking multi-agent policies. The game evolves in a 2×2 m^2 arena with two circular obstacles of radius 0.2 m. We consider $\mathsf{N} = 2$ teams with $\mathsf{M}_i = 3$ holonomic agents each. The agents observe their own position and velocity, the centers of the two obstacles, and the position of all other agents within a radius of 1 m. The same radius is used for communication with teammates. The agents can also observe the velocities of their neighboring evaders. The evaders (pursuers) receive a positive (negative) reward proportional to their distances from the pursuers (evaders), and are penalized (rewarded) every time they are caught by a pursuer. An episode takes $\mathsf{K} = 100$ steps.

We compare our policy with a multi-layer perceptron (MLP) and a graph neural network (GNN) built as a graph attention network [32]. The MLP, with hidden dimensions of $[256, 256, 256]$, is a centralized architecture in the sense that it collects the observations of all agents of all teams as input. The GNN is a distributed architecture that uses only local observations as inputs and also uses hidden dimensions of $[256, 256, 256]$ for the feature encoders. Our policy model uses hidden dimensions of $[64, 64]$ for the attention layers. All networks use hyperbolic tangents as nonlinear activations. To train these policies, we create policy instances for both teams and train them simultaneously using Multi-Agent Proximal

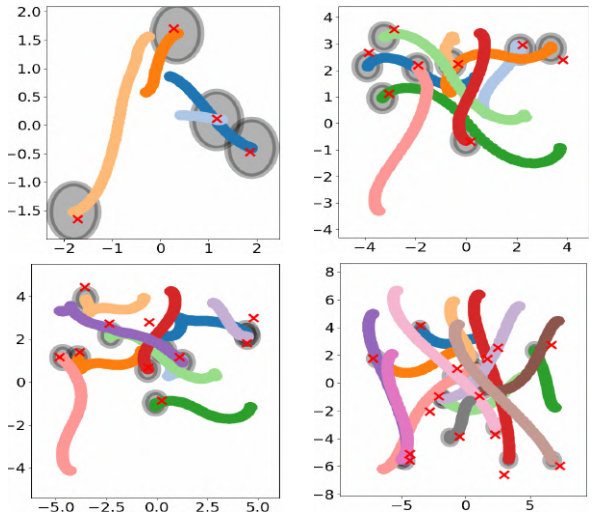


Fig. 5: Scalability in the number of agents. Our policy, applied to $\{4, 8, 10, 15\}$ agents, behaves similarly as in the 7 agents experiment of Fig. 3.

TABLE I: Mean and standard deviation of the total cost J_i^∞ over 20 episodes, normalized by area size.

4 agents	7 agents	8 agents	10 agents	15 agents
277 ± 116	241 ± 103	308 ± 134	248 ± 84	290 ± 73

Policy Optimization (MAPPO) [33] and Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [29] with the default parameters in BenchMARL for 3×10^6 iterations. We train all policies with two different algorithms to ensure that the results are not specific to a training protocol.

To assess performance, we conduct a nonparametric binomial paired test [34], [35] as follows. We first set Ours as the candidate architecture for the pursuer team (evader team). Then, we pick either Ours, MLP or GNN as challenged architecture for the pursuer team as well (evader team). With the two challenged policies for the pursuer team, we run randomly initialized 500 episodes with the same seed against an evader team (pursuer team) using MLP or GNN as baselines. To compare the performance of the two challenged architectures, we consider the cumulative number of catches per episode, $\sum_{i=1}^{\mathsf{M}_{\text{eva}}} \left[\min_{j \in \{1,\dots,\mathsf{M}_{\text{pur}}\}} \|\mathbf{q}^l(k) - \mathbf{q}^p(k)\| \leq 0.125\right]$ with $[\bullet]$ denoting the Iverson bracket. A challenged architecture wins in an episode if the pursuers catch more (less) evaders than the other architecture. The significance test, therefore, checks the following null hypothesis: does Ours lose more than the challenged architecture over the baseline? Rejection of the null hypothesis means that there is sufficient evidence of Ours improving upon the challenged architecture.

As shown in Figures 6 and 7, our method is the best model for evaders and pursuers. Ours demonstrates a clear performance edge over MLP and GNN, particularly when trained with MAPPO, where the null hypothesis is rejected 9 out of 12 times. This dominance is most evident in the pursuer's role, where Ours frequently secured win rates exceeding 80%. While MLP consistently proved to be the weakest baseline across all tests, the comparison with GNN is more nuanced. Ours generally leads in on-policy MAPPO settings but GNN remains a robust and highly competitive

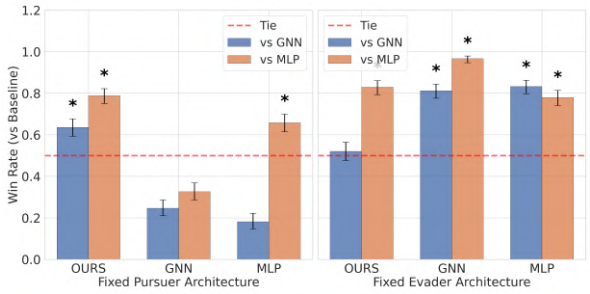


Fig. 6: Nonparametric binomial paired test over 500 episodes for policies trained with MAPPO. We report the win rates and Wilson confidence intervals (95%). Asterisk denotes when the null hypothesis has been rejected with a p-value lower than 0.05.

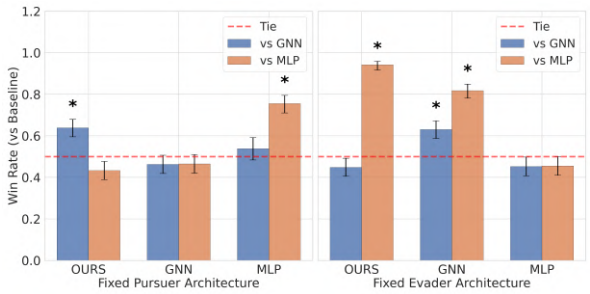


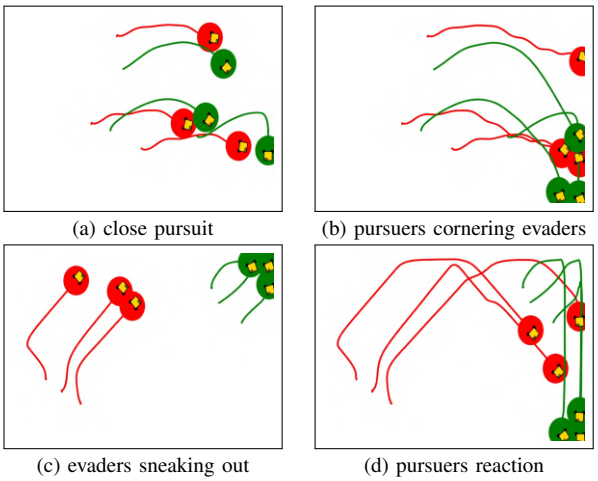
Fig. 7: Nonparametric binomial paired test over 500 episodes for policies trained with MADDPG. We report the win rates and Wilson confidence intervals (95%). Asterisk denotes when the null hypothesis has been rejected with a p-value lower than 0.05.

baseline in off-policy MADDPG settings, although with a significantly higher number of parameters. Interestingly, the defensive capabilities of Ours as an evader are more sensitive to opponent strategies. However, even in cases where the null hypothesis cannot be rejected, Ours outperforms (Fig. 7, fixed pursuer MLP) or ties (Fig. 6, fixed evader Ours) versus the GNN baseline. Ultimately, the synergy between our attention-based policy parameterization and MAPPO's centralized advantage estimation provides a superior method for policy learning in multi-team games. In terms of scalability, physical constraints of the scenario limit the number of agents that can be used for evaluation, as can be observed in the qualitative results on our webpage[1].

D. Pursuit and evasion with real robots

Finally, we test our policy parameterization in a real-robot deployment. We zero-shot transfer the policies trained in Sec. IV-C to the Georgia Tech Robotarium [36], a remotely accessible, multi-robot research facility that allows for fast integration and deployment of control policies. The arena is 3.2×2.0 m^2 and has a fleet of 20 non-holonomic robots with radius of 0.11 m. Due to space constraints, we limit the deployment to $\mathsf{M}_i = 3$ agents per team, as in the previous section. Robots include a projection method to match the holonomic commands from the learned policies to the their non-holonomic dynamics, and control barrier functions that ensure safety during robot operation. These are imposed by the Robotarium platform and cannot be modified.

[1]https://sites.google.com/view/learning-distributed-games



(a) close pursuit (b) pursuers cornering evaders

(c) evaders sneaking out (d) pursuers reaction

Fig. 8: Simulated Robotarium environment modeling the physical interactions and restrictions of the real robots. (top) When both teams are initialized close to each other, the evaders (green) try to escape from the pursuers (red), but the pursuers corner them. Due to the safety filters, the game ends because none of the robots can navigate through others in tight spaces. (bottom) When the teams are initialized far from each other, the evaders go to the most distant corner to minimize the chances of being caught. When the pursuers approach, the evaders sneak out to the other corner, where they finally get trapped due to the safety and space constraints.

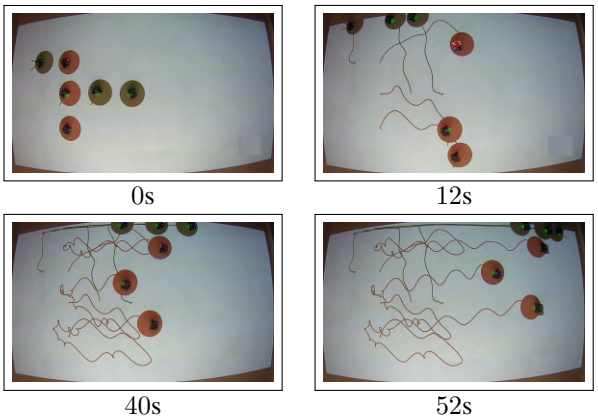


Fig. 9: Real Robotarium deployment. Initially, the evaders (green) can deceive the pursuers (red) and escape to the top-left corner of the arena. However, the pursuers react and try to trap the evaders. The evaders are able to sneak out through the topmost border and escape to the top-right corner being pursued by the other team. The pursuers eventually trap the evaders, finishing the game.

Fig. 8 presents snapshots of the Robotarium simulator, which is designed to be as close to the real setting as possible. Despite the gap in realism between the gym-based environment and the real-robot setting, the policies display complex pursuit-and-evasion behaviors. For instance, the evaders deceive the pursuers and react in advance to their efforts to catch them, whereas the pursuers counteract and finally corner the evaders. These complex behaviors also happen in real deployments (Fig. 9), where the pursuers firstly trap the evaders in the top-left corner of the arena but are then tricked by the evaders, who sneak out to the top-right corner of the arena, to finally be trapped again. Videos and further material can be found on the website.

V. Conclusions

This paper developed a novel method for nonlinear multi-agent dynamic games, focused on model-free multi-team games with time-varying configurations. Inspired by optimal control policies in linear and nonlinear setups, we proposed a novel distributed feedback gain that learns team-level strategies from cost signals. The gains are parameterized with a self-attention structure that handles the dynamic nature of the connectivity of the agents' graph and the diversity in roles/identities of the agents. Experiments showed that our method finds optimal policies under topological constraints without transition and cost models, and that the policies can be deployed in real multi-robot teams. The main limitation of our evaluation is the lack of multi-team robotic benchmarks to comprehensively assess performance across different tasks with different numbers of teams. Existing benchmarks consider either teams of 1 agent each or only 2 teams. However, training policies for real applications, like perimeter defense in Fig. 1, require multi-team, multi-agent benchmarks beyond the currently available resources. In future work, it will also be important to investigate the performance of the learned policies under real communication failures and changes in the number of teammates during the game.

References

[1] R. Spica, E. Cristofalo, Z. Wang, E. Montijano, and M. Schwager, "A real-time game theoretic planner for autonomous two-player drone racing," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1389–1403, 2020.

[2] S. Park, Y. D. Zhong, and N. E. Leonard, "Multi-robot task allocation games in dynamically changing environments," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 8678–8684.

[3] E. Sebastián, E. Montijano, and C. Sagüés, "Adaptive multirobot implicit control of heterogeneous herds," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3622–3635, 2022.

[4] E. Sebastián, T. Duong, N. Atanasov, E. Montijano, and C. Sagüés, "Physics-informed multi-agent reinforcement learning for distributed multi-robot problems," *IEEE Transactions on Robotics*, 2025.

[5] D. Shishika and V. Kumar, "A review of multi agent perimeter defense games," in *International Conference on Decision and Game Theory for Security*, 2020, pp. 472–485.

[6] J. Chen, Z. Tang, and M. Guo, "Accelerated K-serial stable coalition for dynamic capture and resource defense," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 443–450, 2023.

[7] F. Parise and A. Ozdaglar, "Analysis and interventions in large network games," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 455–486, 2021.

[8] B. Lian, W. Xue, F. L. Lewis, and A. Davoudi, "Nash-minmax strategy for multiplayer multiagent graphical games with reinforcement learning," *IEEE Transactions on Control of Network Systems*, 2024.

[9] D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. J. Tomlin, "Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 1475–1481.

[10] Y. Zheng, Y. Sun, M. Fazel, and N. Li, "Escaping high-order saddles in policy optimization for linear quadratic gaussian (LQG) control," in *IEEE Conference on Decision and Control*, 2022, pp. 5329–5334.

[11] S. Williams and J. Deshmukh, "Potential games on cubic splines for self-interested multi-agent motion planning," *IEEE Control Systems Letters*, vol. 7, pp. 2487–2492, 2024.

[12] M. Bhatt, Y. Jia, and N. Mehr, "Strategic decision-making in multi-agent domains: A weighted constrained potential dynamic game approach," *IEEE Transactions on Robotics*, vol. 41, pp. 2749–2764, 2025.

[13] T. Kavuncu, A. Yaraneri, and N. Mehr, "Potential iLQR: A potential-minimizing controller for planning multi-agent interactive trajectories," in *Robotics: Science and Systems*, 2021.

[14] R. Zhang, J. Shamma, and N. Li, "Equilibrium selection for multi-agent reinforcement learning: A unified framework," *arXiv preprint arXiv:2406.08844*, 2024.

[15] L. Furieri, Y. Zheng, and M. Kamgarpour, "Learning the globally optimal distributed LQ regulator," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 287–297.

[16] D. T. A. Nguyen, M. Bianchi, F. Dörfler, D. T. Nguyen, and A. Nedić, "Constrained multi-cluster game: Distributed nash equilibrium seeking over directed graphs," in *IEEE Conference on Decision and Control*, 2024, pp. 4712–4719.

[17] G. Carnevale, N. Mimmo, and G. Notarstefano, "A unifying system theory framework for distributed optimization and games," *IEEE Transactions on Automatic Control*, 2025.

[18] Y. Yang and J. Wang, "An overview of multi-agent reinforcement learning from game theoretical perspective," *arXiv preprint arXiv:2011.00583*, 2020.

[19] A. Giannou, K. Lotidis, P. Mertikopoulos, and E.-V. Vlatakis-Gkaragkounis, "On the convergence of policy gradient methods to Nash equilibria in general stochastic games," *Advances in Neural Information Processing Systems*, vol. 35, pp. 7128–7141, 2022.

[20] S. Aydin and C. Eksin, "Policy gradient play with networked agents in markov potential games," in *Learning for Dynamics and Control Conference*. PMLR, 2023, pp. 184–195.

[21] Z. Huang, Z. Yang, R. Krupani, B. Şenbaşlar, S. Batra, and G. S. Sukhatme, "Collision avoidance and navigation for a quadrotor swarm using end-to-end deep reinforcement learning," in *IEEE International Conference on Robotics and Automation*, 2024, pp. 300–306.

[22] V. Kučera, "The discrete Riccati equation of optimal control," *Kybernetika*, vol. 8, no. 5, pp. 430–447, 1972.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[25] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[26] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 21, no. 178, pp. 1–51, 2020.

[27] B. Wang, J. Xie, and N. Atanasov, "DARL1N: Distributed multi-agent reinforcement learning with one-hop neighbors," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022.

[28] Z. Williams, J. Chen, and N. Mehr, "Distributed potential iLQR: Scalable game-theoretic trajectory planning for multi-agent interactions," in *IEEE International Conference on Robotics and Automation*, 2023.

[29] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in Neural Information Processing Systems*, 2017.

[30] M. Bettini, R. Kortvelesy, J. Blumenkamp, and A. Prorok, "VMAS: A vectorized multi-agent simulator for collective robot learning," *International Symposium on Distributed Autonomous Robotic Systems*, 2022.

[31] M. Bettini, A. Prorok, and V. Moens, "BenchMARL: Benchmarking multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 25, no. 217, pp. 1–10, 2024.

[32] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[33] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative multi-agent games," *Advances in neural information processing systems*, vol. 35, pp. 24 611–24 624, 2022.

[34] N. Smeeton, N. H. Spencer, and P. Sprent, *Applied nonparametric statistical methods*. Chapman and Hall/CRC, 2025.

[35] D. Snyder, A. J. Hancock, A. Badithela, E. Dixon, P. Miller, R. A. Ambrus, A. Majumdar, M. Itkina, and H. Nishimura, "Is your imitation learning policy better than mine? policy comparison with near-optimal stopping," in *Robotics: Science & Systems*, 2025.

[36] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The Robotarium: A remotely accessible swarm robotics research testbed," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1699–1706.