LEMURS: Learning Distributed Multi-Robot Interactions

Eduardo Sebastián Thai Duong Nikolay Atanasov Eduardo Montijano Carlos Sagüés

***Abstract*— This paper presents LEMURS, an algorithm for learning scalable multi-robot control policies from cooperative task demonstrations. We propose a port-Hamiltonian description of the multi-robot system to exploit universal physical constraints in interconnected systems and achieve closed-loop stability. We represent a multi-robot control policy using an architecture that combines self-attention mechanisms and neural ordinary differential equations. The former handles time-varying communication in the robot team, while the latter respects the continuous-time robot dynamics. Our representation is distributed by construction, enabling the learned control policies to be deployed in robot teams of different sizes. We demonstrate that LEMURS can learn interactions and cooperative behaviors from demonstrations of multi-agent navigation and flocking tasks.**

I. Introduction

Multi-robot systems promise improved efficiency and reliability compared to a single robot in many applications, including exploration and mapping [1], [2], agriculture and herding [3]–[6], and search and rescue [7]. However, designing multi-robot control policies that achieve cooperative behaviors may be challenging. First, domain expertise may be required to specify the objective and constraints for a desired task in mathematical terms. Second, scaling the control policy to large teams may be computationally infeasible due to the increase of the joint state and control spaces. The first challenge motivates the use of machine learning techniques to learn reward functions or control policies from demonstration [8]–[23]. The second challenge motivates imposing a sparse structure in the control policy that respects the communication topology of the robot team and allows the complexity to scale with the number of neighbors [10]–[15], [24]–[26]. In this work, we develop **LEMURS** (**LE**arning distributed **MU**lti-**R**obot interaction**S**), a learning approach for distributed control synthesis from cooperative task demonstrations that generalizes to different tasks, scales favorably with the number of robots, and handles time-varying robot communication.

Recent works focus on learning control policies for optimal control or reinforcement learning problems [9], [18], [19]. Given a cost function, a recurrent neural network [22], or graph convolutions and multi-layer perceptrons [23] have been used to learn centralized control policies. Without a cost function, inverse reinforcement learning [27] may be used to learn centralized [20], [21] or distributed [16] policies from task demonstrations. While black-box neural networks are widely used for learning control policies, they do not encode energy conservation and kinematic constraints satisfied by physical robot systems, and failing to infer them from data may result in unstable behaviors. A key contribution of our work is to represent the robot team as a *port-Hamiltonian system* [28] and learn a distributed control policy from demonstration by modeling robot interactions as energy exchanges. The use of Hamiltonian mechanics has been explored for centralized control policies or fixed-time known topologies [15], [29], in which scalability is achieved in the absence of communication [17]. Meanwhile, our work achieves scalability with a time-varying topology by modeling robot interactions using *self-attention techniques* [30].

Learning and execution of control policies for multi-robot systems should scale favorably with an increasing numbers of robots. Learning a joint value or policy function is challenging due to the exponential growth of the state and action space [24]. Successful methods for multi-agent reinforcement learning factorize value functions according to the k-hop neighborhoods [24], [26] or using attention mechanisms [31]. Graph neural networks have been utilized as a scalable and communication-aware policy representation in coverage, exploration, and flocking problems [10]–[14]. Recently, Li et al. [32] combine graph-neural networks with self-attention to solve decentralized multi-robot path planning problems. Many of these techniques assume discrete robot dynamics, fixed or known communication topology, or prior knowledge about the task. In contrast, our approach learns from demonstrated robot trajectories with an unknown task objective and handles time-varying communication and team sizes. In order to handle continuous-time dynamics, we use *neural ordinary differential equation (ODE) networks* [33]. Closely related, Jiahao et al. [8] develop a neural ODE network that learns distributed controllers but enforces collision avoidance using an explicit potential field and assumes a fixed maximum number of neighbors. By using a port-Hamiltonian formulation and self-attention mechanism, we handle time-varying neighbors, do not constrain the size of the neighborhoods, and learn constraints such as collision avoidance from data.

In summary, we develop LEMURS, a novel algorithm for learning scalable multi-robot control policies from demonstration. Our *first contribution* is the use of port-Hamiltonian

E. Sebastián, E. Montijano and C. Sagüés are with the RoPeRt group, at DIIS - I3A, Universidad de Zaragoza, Spain (e-mails: {esebastian, emonti, csagues}@unizar.es).

T. Duong and N. Atanasov are with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093 USA (e-mails: {tduong, natanasov}@ucsd.edu).

This work has been supported by NSF CCF-2112665 (TILOS), by the ONR Global grant N62909-19-1-2027 and via Spanish projects PID2021-125514NB-I00, PID2021-124137OBI00 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 T45-20R, and Spanish grant FPU19-05700 and EST22/00253.

dynamics to restrict the family of learned policies to those that are stable and distributed. Our *second contribution* is a novel learning architecture that integrates concepts of self-attention and neural ODEs to handle continuous-time dynamics, time-varying communication, and large robot teams.

II. Problem Statement

Consider a team of robots, indexed by $\mathcal{V} = \{1, \dots, n\}$. Assume that the dynamics of each robot $i \in \mathcal{V}$ are *known*:

$$\dot{\mathbf{x}}_i(t) = \mathbf{f}_i(\mathbf{x}_i(t), \mathbf{u}_i(t)), \tag{1}$$

where $\mathbf{x}_i(t) \in \mathbb{R}^{n_x}$ and $\mathbf{u}_i(t) \in \mathbb{R}^{n_u}$ denote the state and control input of the robot at time $t \geq 0$. The robots interact in a distributed manner, described by a time-varying undirected graph $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t))$, where $\mathcal{E}(t) \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. An edge $(i, j) \in \mathcal{E}(t)$ exists when robots i and j interact at time t. Robot i can always interact with itself, i.e., $(i, i) \in \mathcal{E}(t)$ for all i, t. Let $\mathbf{A}(t) \in \{0, 1\}^{n \times n}$ be the weighted adjacency matrix associated to $\mathcal{G}(t)$, such that $[\mathbf{A}(t)]_{ij} \neq 0$ if and only if $(i, j) \in \mathcal{E}(t)$, and 0 otherwise. The set of k-hop neighbors of robot i at t is $\mathcal{N}_i^k(t) = \{j \in \mathcal{V} \mid [\mathbf{A}^k(t)]_{ij} \neq 0\}$. Each robot executes an *unknown* control policy:

$$\mathbf{u}_i(t) = \boldsymbol{\pi}_{\boldsymbol{\theta}} \left(\mathbf{x}_{\mathcal{N}_i^k}(t) \right), \tag{2}$$

where $\mathbf{x}_{\mathcal{N}_i^k}(t) = \{\mathbf{x}_j(t) \mid j \in \mathcal{N}_i^k(t)\}$ and $\boldsymbol{\theta}$ is the control policy parameters. Our objective is to use task demonstrations to learn $\boldsymbol{\theta}$, such that the multi-robot team, following the learned control policy, replicates the task.

Example 1. *Consider a flocking task [34] in which a robot team must achieve a formation with aligned velocities, while avoiding collisions. The robots follow double integrator dynamics $\dot{\mathbf{p}}_i(t) = \mathbf{v}_i(t)$ and $\dot{\mathbf{v}}_i(t) = \mathbf{u}_i(t)$, where $\mathbf{p}_i(t) \in \mathbb{R}^m$, $\mathbf{v}_i(t) \in \mathbb{R}^m$, and $\mathbf{u}_i(t) \in \mathbb{R}^m$ are the position, velocity, and input of robot i. A distributed control policy that achieves flocking was developed by Olfati-Saber [34]:*

$$\begin{aligned}\mathbf{u}_i(t) = &- c_1 \mathbf{p}_i(t) - c_2 \mathbf{v}_i(t) + \sum_{j \in \mathcal{N}_i^1(t)} \phi(||\Delta \mathbf{p}_{ij}(t)||_\sigma) \mathbf{n}_{ij}(t) \\ &+ \sum_{j \in \mathcal{N}_i^1(t)} \rho\left(||\Delta \mathbf{p}_{ij}(t)||_\sigma\right) \left(\mathbf{v}_j(t) - \mathbf{v}_i(t)\right)\end{aligned} \tag{3}$$

where $\|\cdot\|_\sigma$ is the σ-norm of a vector and $\Delta \mathbf{p}_{ij}(t) = \mathbf{p}_j(t) - \mathbf{p}_i(t)$. The first and second terms are a proportional controller with gains $c_1, c_2 > 0$ that prevents the formation of sub-flocks, where we have assumed $\mathbf{p}_i(t) = \mathbf{v}_i(t) = \mathbf{0}$ $\forall i$ as the desired flock configuration. The third term avoids robot collisions and induces the desired lattice formation, where $\phi(\cdot)$ is a potential function whose minima are located at the desired inter-robot distances, and $\mathbf{n}_{ij}(t)$ is a vector that points in the repulsion/coalition direction. The last term achieves velocity consensus using a distance scaling function $\rho(\cdot)$ that models the robot communication. Further details can be found in [34]. This paper aims to learn policies like (3) *from demonstrations.*

We assume that state trajectories from successful task executions are available as training data. Let $\mathbf{x}(t) = [\mathbf{x}_1^\top(t), \dots, \mathbf{x}_n^\top(t)]^\top$ and $\mathbf{u}(t) = [\mathbf{u}_1^\top(t), \dots, \mathbf{u}_n^\top(t)]^\top$ denote the joint state and control of the robot team. Given an initial state $\mathbf{x}(0) = \mathbf{x}_0$, let $\bar{\mathbf{x}}_{0:K} := [\bar{\mathbf{x}}(0), \dots, \bar{\mathbf{x}}(rT), \dots, \bar{\mathbf{x}}(KT)]$ and $\mathbf{x}_{0:K} := [\mathbf{x}(0), \dots, \mathbf{x}(rT), \dots, \mathbf{x}(KT)]$ be demonstrated and learned policy's trajectories, respectively, with K denoting the number of discrete samples r along the trajectories with sampling interval T. Let $\bar{\mathcal{D}} := \{\bar{\mathbf{x}}_{0:K}^l\}_{l=0}^{l=L}$ denote a dataset of $L > 0$ demonstrated trajectories. Let $\mathcal{D} := \{\mathbf{x}_{0:K}^l\}_{l=0}^{l=L}$ be the generated trajectories under policy $\pi_{\boldsymbol{\theta}}$. We aim to learn a control policy that minimizes the distance between the demonstrated and generated trajectories:

$$\mathcal{L}(\mathcal{D}, \bar{\mathcal{D}}) = \frac{1}{KL} \sum_{l=0}^{L} \sum_{r=0}^{K} ||\mathbf{x}^l(rT) - \bar{\mathbf{x}}^l(rT)||_2^2. \tag{4}$$

Formally, the problem we consider is:

$$\min_{\boldsymbol{\theta}} \ \mathcal{L}(\mathcal{D}, \bar{\mathcal{D}}) \tag{5a}$$

$$\text{s.t. } \dot{\mathbf{x}}_i^l(t) = \mathbf{f}_i(\mathbf{x}_i^l(t), \mathbf{u}_i^l(t)), \ \mathbf{x}_i^l(0) = \bar{\mathbf{x}}_i^l(0), \ \forall i, l, \tag{5b}$$

$$\mathbf{u}_i^l(t) = \boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{x}_{\mathcal{N}_i^k}^l(t)), \ \forall i, l, t. \tag{5c}$$

As specified in the formulation above, the learned control policy should also handle time-varying communication and should be adaptable to changes in the total number of robots or the number of neighbors for each robot.

III. LEMURS

In this section, we present a port-Hamiltonian formulation of the multi-robot dynamics and an energy-based distributed control design that can shape the interactions and Hamiltonian of the closed-loop system (Sec. III-A). Given task demonstrations, we employ self-attention and neural ordinary differential equations to learn the interactions and energy parameters of the control policy that minimize the distance between the demonstrated and generated trajectories (Sec. III-B). To simplify the notation, we omit the time dependence of the states $\mathbf{x}$ and controls $\mathbf{u}$ in the remainder of the paper.

A. Port-Hamiltonian Formulation of Multi-Robot Dynamics

Port-Hamiltonian mechanics are a general yet interpretable modeling approach for learning and control. On the one hand, many physical networked systems can be described as a port-Hamiltonian system [15] using the same formulation and with a modular and distributed interpretation. Meanwhile, the port-Hamiltonian description allows to derive general energy-based controllers with closed-loop stability guarantees. Since robots are physical systems that satisfy Hamiltonian mechanics, we model each robot's dynamics in (1) as a port-Hamiltonian system [28]:

$$\dot{\mathbf{x}}_i = \left(\mathbf{J}_s^{(i)}(\mathbf{x}_i) - \mathbf{R}_s^{(i)}(\mathbf{x}_i) \right) \frac{\partial H_s^{(i)}(\mathbf{x}_i)}{\partial \mathbf{x}_i} + \mathbf{F}_s^{(i)}(\mathbf{x}_i) \mathbf{u}_i, \tag{6}$$

where the skew-symmetric interconnection matrix $\mathbf{J}_s^{(i)}(\mathbf{x}_i)$ represents energy exchange *within a robot*, the positive-semidefinite dissipation matrix $\mathbf{R}_s^{(i)}(\mathbf{x}_i)$ represents energy

dissipation, the Hamiltonian $H_s^{(i)}(\mathbf{x}_i)$ represents the total energy, and the matrix $\mathbf{F}_s^{(i)}(\mathbf{x}_i)$ is the input gain. Then, the multi-robot system with joint state $\mathbf{x}$ also follows port-Hamiltonian dynamics:

$$\dot{\mathbf{x}} = (\mathbf{J}_s(\mathbf{x}) - \mathbf{R}_s(\mathbf{x})) \frac{\partial H_s(\mathbf{x})}{\partial \mathbf{x}} + \mathbf{F}_s(\mathbf{x})\mathbf{u}, \tag{7}$$

where $H_s(\mathbf{x}) = \sum_{i=1}^n H_s^{(i)}(\mathbf{x}_i)$ and

$$\begin{aligned} \mathbf{J}_s(\mathbf{x}) &= \text{diag}\left(\mathbf{J}_s^{(1)}(\mathbf{x}_1), \dots, \mathbf{J}_s^{(n)}(\mathbf{x}_n)\right), \\ \mathbf{R}_s(\mathbf{x}) &= \text{diag}\left(\mathbf{R}_s^{(1)}(\mathbf{x}_1), \dots, \mathbf{R}_s^{(n)}(\mathbf{x}_n)\right), \\ \mathbf{F}_s(\mathbf{x}) &= \text{diag}\left(\mathbf{F}_s^{(1)}(\mathbf{x}_1), \dots, \mathbf{F}_s^{(n)}(\mathbf{x}_n)\right). \end{aligned} \tag{8}$$

Without control, the trajectories of the open-loop system in (7) would not match the demonstrations in $\bar{\mathcal{D}}$. The dynamics need to be controlled by the policy in (2) in order to generate desired trajectories. We employ an interconnection and damping assignment passivity-based control (IDA-PBC) approach [28], which injects additional energy to the system through the control input $\mathbf{u}$ to achieve some closed-loop dynamics that replicate the demonstrated task:

$$\dot{\mathbf{x}} = (\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x}) - \mathbf{R}_{\boldsymbol{\theta}}(\mathbf{x})) \frac{\partial H_{\boldsymbol{\theta}}(\mathbf{x})}{\partial \mathbf{x}}, \tag{9}$$

with Hamiltonian $H_{\boldsymbol{\theta}}(\mathbf{x})$, skew-symmetric interconnection $\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x})$, and positive semidefinite dissipation $\mathbf{R}_{\boldsymbol{\theta}}(\mathbf{x})$. By matching the terms in (7) and (9), one obtains the policy:

$$\begin{aligned} \mathbf{u} = \mathbf{F}_s^{\dagger}(\mathbf{x}) \Bigg(&(\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x}) - \mathbf{R}_{\boldsymbol{\theta}}(\mathbf{x})) \frac{\partial H_{\boldsymbol{\theta}}(\mathbf{x})}{\partial \mathbf{x}} \\ &- (\mathbf{J}_s(\mathbf{x}) - \mathbf{R}_s(\mathbf{x})) \frac{\partial H_s(\mathbf{x})}{\partial \mathbf{x}} \Bigg), \end{aligned} \tag{10}$$

where $\mathbf{F}_s^{\dagger}(\mathbf{x}) = \left(\mathbf{F}_s^{\top}(\mathbf{x})\mathbf{F}_s(\mathbf{x})\right)^{-1}\mathbf{F}_s^{\top}(\mathbf{x})$ is the pseudo-inverse of $\mathbf{F}_s(\mathbf{x})$. If the robots are fully-actuated, i.e., $\mathbf{F}_s(\mathbf{x})$ is full-rank, the matching condition on the pseudo-inverse is always satisfied, achieving the desired closed-loop dynamics. For underactuated systems, satisfaction of the matching condition may not always be possible [35]. Being able to achieve zero error $\mathcal{L}(\mathcal{D}, \bar{\mathcal{D}})$ is, hence, related to whether the demonstrated trajectories $\bar{\mathcal{D}}$ are realizable by the class of control policies in (10). Even if the trajectories in $\bar{\mathcal{D}}$ are not realizable, the policy parameters $\boldsymbol{\theta}$ may still be optimized to achieve a behavior as similar as possible.

Let $[\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x})]_{ij}$ and $[\mathbf{R}_{\boldsymbol{\theta}}(\mathbf{x})]_{ij}$ denote the $n_x \times n_x$ blocks with index (i, j), representing the energy exchange between robot i and j and the energy dissipation of robot i caused by robot j, respectively. Since the input gain $\mathbf{F}_s(\mathbf{x})$ in (8) is block-diagonal, the individual control policy of robot i is:

$$\begin{aligned} \mathbf{u}_i = \mathbf{F}_s^{(i)\dagger}(\mathbf{x}_i) \Bigg(&\sum_{j \in \mathcal{V}} \left([\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x})]_{ij} - [\mathbf{R}_{\boldsymbol{\theta}}(\mathbf{x})]_{ij}\right) \frac{\partial H_{\boldsymbol{\theta}}(\mathbf{x})}{\partial \mathbf{x}_j} \\ &- \left(\mathbf{J}_s^{(i)}(\mathbf{x}_i) - \mathbf{R}_s^{(i)}(\mathbf{x}_i)\right) \frac{\partial H_s^{(i)}(\mathbf{x})}{\partial \mathbf{x}_i} \Bigg). \end{aligned} \tag{11}$$

The individual control policies in (11) do not necessarily respect the hops in the communication network as desired in (2) because this depends on the structure of $\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x})$, $\mathbf{R}_{\boldsymbol{\theta}}(\mathbf{x})$, and $H_{\boldsymbol{\theta}}(\mathbf{x})$. In Sec. III-B, we impose conditions on these terms to ensure that they respect the communication topology and are skew-symmetric, and positive semidefinite, respectively, as required for a valid port-Hamiltonian system.

Example 2. *By substituting the control policy* (3) *in the double integrator dynamics of the robots, the closed-loop dynamics for the flocking problem in Example 1 are:*

$$\begin{aligned} \dot{\mathbf{p}} = \mathbf{v}, \quad \dot{\mathbf{v}} = -\frac{\partial U_{\boldsymbol{\theta}}(\mathbf{p})}{\partial \mathbf{p}} - \mathbf{D}_{\boldsymbol{\theta}}(\mathbf{p})\mathbf{v}, \\ \textit{with } \frac{\partial U_{\boldsymbol{\theta}}(\mathbf{p})}{\partial \mathbf{p}_i} = c_1 \mathbf{p}_i + \sum_{j \in \mathcal{N}_i^1} \phi(||\mathbf{p}_j - \mathbf{p}_i||_\sigma)\mathbf{n}_{ij}, \\ [\mathbf{D}_{\boldsymbol{\theta}}(\mathbf{p})]_{ij} = (c_2[\mathbf{I}_n]_{ij} + \rho\left(||\mathbf{p}_j - \mathbf{p}_i||_\sigma/\tau\right))\mathbf{I}_m. \end{aligned} \tag{12}$$

In port-Hamiltonian terms, $H_s(\mathbf{x}) = \frac{1}{2}\mathbf{v}^\top\mathbf{v}$, $\mathbf{R}_s = \mathbf{0}$, $\mathbf{F}_s(\mathbf{x}) = [\mathbf{0}, \mathbf{I}_m]^\top$, $\mathbf{J}_s(\mathbf{x}) = \mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{pmatrix} \mathbf{0} & \mathbf{I}_n \\ -\mathbf{I}_n & \mathbf{0} \end{pmatrix}$, $\mathbf{R}_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{\boldsymbol{\theta}}(\mathbf{p}) \end{pmatrix}$, *and* $H_{\boldsymbol{\theta}}(\mathbf{x}) = U_{\boldsymbol{\theta}}(\mathbf{p}) + \frac{1}{2}\mathbf{v}^\top\mathbf{v}$.

B. Learning Distributed Multi-Robot Interactions

The analytical design of scalable cooperative control policies like the flocking controller of Example 1 is challenging when the complexity of the task increases. Instead, we seek to learn control policies that scale with the number of robots, handle time-varying communications and guarantee the port-Hamiltonian constraints. To do so, we first derive conditions on $\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x})$, $\mathbf{R}_{\boldsymbol{\theta}}(\mathbf{x})$ and $H_{\boldsymbol{\theta}}(\mathbf{x})$. Then, we develop a novel architecture based on self-attention and neural ordinary differential equations to ensure that the learned control policies guarantee these conditions.

We first impose $\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x})$ and $\mathbf{R}_{\boldsymbol{\theta}}(\mathbf{x})$ to be block-sparse,

$$[\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x})]_{ij} = [\mathbf{R}_{\boldsymbol{\theta}}(\mathbf{x})]_{ij} = \mathbf{0}, \quad \forall j \notin \mathcal{N}_i^k. \tag{13}$$

This is to satisfy the topology constraints of the multi-robot team. Moreover, we require that the desired Hamiltonian factorizes over k-hop neighborhoods:

$$H_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{i=0}^{n} H_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}_{\mathcal{N}_i^k}). \tag{14}$$

The factorization in (14) ensures that each robot i can calculate $\partial H_{\boldsymbol{\theta}}(\mathbf{x})/\partial \mathbf{x}_i = \sum_{j \in \mathcal{N}_i^k} \partial H_{\boldsymbol{\theta}}^{(j)}(\mathbf{x}_{\mathcal{N}_j^k})/\partial \mathbf{x}_i$ by gathering $\partial H_{\boldsymbol{\theta}}^{(j)}(\mathbf{x}_{\mathcal{N}_j^k})/\partial \mathbf{x}_i$ from its k-hop neighbors j.

Then, the control policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ of robot i becomes:

$$\begin{aligned} \mathbf{u}_i = \mathbf{F}_s^{(i)\dagger}(\mathbf{x}_i) \Bigg(&\sum_{j \in \mathcal{N}_i^k} \left([\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x})]_{ij} - [\mathbf{R}_{\boldsymbol{\theta}}(\mathbf{x})]_{ij}\right) \frac{\partial H_{\boldsymbol{\theta}}(\mathbf{x})}{\partial \mathbf{x}_j} \\ &- \left(\mathbf{J}_s^{(i)}(\mathbf{x}_i) - \mathbf{R}_s^{(i)}(\mathbf{x}_i)\right) \frac{\partial H_s^{(i)}(\mathbf{x})}{\partial \mathbf{x}_i} \Bigg). \end{aligned} \tag{15}$$

Imposing the requirements in (13)-(14) is a first step towards making the control policy in (15) distributed. Note that the terms $[\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x})]_{ij}$ and $[\mathbf{R}_{\boldsymbol{\theta}}(\mathbf{x})]_{ij}$ might still depend on the joint state $\mathbf{x}$ even if $j \in \mathcal{N}_i^k$. We discuss how to remove this dependence next and achieve a similar factorization as (14).

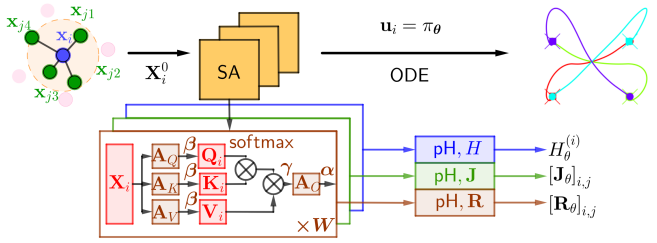


Fig. 1: Architecture of LEMURS: robot i receives information from its neighbors. Then, the self-attention module obtains the port-Hamiltonian terms. Finally, it computes the control policy through an ordinary differential equation solver.

1) Modeling robot interactions using self-attention: We model $[\mathbf{J}_{\boldsymbol{\theta}}]_{ij}, [\mathbf{R}_{\boldsymbol{\theta}}]_{ij}$, and $H_{\boldsymbol{\theta}}^{(i)}$ in Eq. (15) with the parameters $\boldsymbol{\theta}$ shared across the robots, so that the team can handle time-varying communication graphs. Specifically, we propose a novel architecture based on self-attention [30]. Self-attention consists of a sequence of operations (a layer) that extracts the relationships among the inputs of a sequence by calculating the importance associated to each input using an attention map. The length of the sequences can vary as the number of parameters of the self-attention is constant with the number of inputs. Our key idea is to consider the self and neighboring information as the sequence, where each neighbor's information is an input.

To learn $[\mathbf{R}_{\boldsymbol{\theta}}]_{ij}$, robot i will use, at instant t, the states $\mathbf{x}_j$ from all $k-$hop neighbors $j \in \mathcal{N}_i^k$, concatenated as follows:

$$\mathbf{X}_i^0 = [\mathbf{x}_i, \mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \dots, \mathbf{x}_{j_{|\mathcal{N}_i^k|}}] \in \mathbb{R}^{n_x \times (|\mathcal{N}_i^k|+1)}. \quad (16)$$

For each layer $w = 1, \dots, W$, we define:

$$\mathbf{Q}_i^w = \mathbf{A}_{Q,\mathbf{R}}^w \mathbf{X}_i^w, \ \mathbf{K}_i^w = \mathbf{A}_{K,\mathbf{R}}^w \mathbf{X}_i^w, \ \mathbf{V}_i^w = \mathbf{A}_{V,\mathbf{R}}^w \mathbf{X}_i^w, \quad (17)$$

$$\mathbf{Y}_i^w = \gamma\left(\mathsf{softmax}\left(\frac{\beta(\mathbf{Q}_i^w)\beta((\mathbf{K}_i^w)^\top)}{\sqrt{|\mathcal{N}_i|}}\right)\beta(\mathbf{V}_i^w)\right), \quad (18)$$

$$\mathbf{X}_i^w = \alpha(\mathbf{A}_{Z,\mathbf{R}}^w \mathbf{Y}_i^w), \quad (19)$$

where $\mathsf{softmax}$ stands for the softmax operation; W is the number of self-attention layers; $\mathbf{A}_{Q,\mathbf{R}}^w, \mathbf{A}_{K,\mathbf{R}}^w, \mathbf{A}_{V,\mathbf{R}}^w \in \mathbb{R}^{r_w \times h_w}$ and $\mathbf{A}_{Z,\mathbf{R}}^w \in \mathbb{R}^{d_w \times r_w}$ for $w = 1, \dots, W$ are matrices to be learned and shared across robots; and $h_w, r_w, d_w > 0$, with $d_W = n_x^2$ and $h_1 = n_x$ for valid matrix multiplications. The size of $\mathbf{A}_{Q,\mathbf{R}}^w, \mathbf{A}_{K,\mathbf{R}}^w, \mathbf{A}_{V,\mathbf{R}}^w$ does not depend on the number of robots, so robot i can deal with time-varying neighbors. Nonlinear activation functions $\beta(\cdot)$, $\gamma(\cdot)$ and $\alpha(\cdot)$ account for potential nonlinearities. The concatenation in (16) is valid since the self-attention equation (18) learns the relationship among all the elements of $\mathbf{X}_i^w$ via the inner matrix multiplication. Then, $[\mathbf{R}_{\boldsymbol{\theta}}]_{ij}$ is constructed as a weighted matrix that models the interactions of robot i with its neighbors, and a diagonal positive semidefinite matrix that accounts for the self-interactions:

$$\begin{aligned} \mathbf{Z}_{ij}^R &= \mathsf{vec}^{-1}(\mathbf{x}_{ij}^W) \\ [\mathbf{R}_{\boldsymbol{\theta}}]_{ij} &= -(\mathbf{Z}_{ij}^R + \mathbf{Z}_{ji}^R), \quad \forall j \in \mathcal{N}_i^k \backslash \{i\}, \\ [\mathbf{R}_{\boldsymbol{\theta}}]_{ii} &= \mathbf{Z}_{ii}^R + \sum_{j \in \mathcal{N}_i^k \backslash \{i\}} (\mathbf{Z}_{ij}^R + \mathbf{Z}_{ji}^R), \end{aligned} \quad (20)$$

where $\mathbf{x}_{ij}^W$ is the column $\mathbf{x}_j$ that corresponds to neighbor j in $\mathbf{X}_i^W$, $\mathsf{vec}^{-1}(\cdot)$ is the operator that reshapes the $d_W \times 1$ vector to a $n_x \times n_x$ matrix. This way, $[\mathbf{R}_{\boldsymbol{\theta}}]_{ij}$ is positive semidefinite by design.

To construct $[\mathbf{J}_{\boldsymbol{\theta}}]_{ij}$, we follow the same steps (17)-(19), with parameters $\mathbf{A}_{Q,\mathbf{J}}^w, \mathbf{A}_{K,\mathbf{J}}^w, \mathbf{A}_{V,\mathbf{J}}^w$ and $\mathbf{A}_{Z,\mathbf{J}}^w$, to obtain encodings $\mathbf{Z}_{ij}^J$. Due to the undirected communication between robots i and j, we enforce the skew-symmetry of $\mathbf{J}_{\boldsymbol{\theta}}$ by:

$$[\mathbf{J}_{\boldsymbol{\theta}}]_{ij} = \mathbf{Z}_{ij}^J - \mathbf{Z}_{ji}^J \quad \forall j \in \mathcal{N}_i^k. \quad (21)$$

For each robot i, we construct $H_{\boldsymbol{\theta}}^{(i)}$ as follows:

$$H_{\boldsymbol{\theta}}^{(i)} = (\mathsf{vec}(\mathbf{X}_i^0))^\top \mathbf{M}_{\boldsymbol{\theta}}^{(i)}(\mathbf{X}_i^0)(\mathsf{vec}(\mathbf{X}_i^0)) + U_{\boldsymbol{\theta}}^{(i)}(\mathbf{X}_i^0), \quad (22)$$

where the first term $(\mathsf{vec}(\mathbf{X}_i^0))^\top \mathbf{M}_{\boldsymbol{\theta}}^{(i)}(\mathbf{X}_i^0)(\mathsf{vec}(\mathbf{X}_i^0))$ is a kinetic-like energy function with $\mathbf{M}_{\boldsymbol{\theta}}^{(i)}(\mathbf{X}_i^0) = \mathrm{diag}(\mathbf{1}^\top \mathbf{Z}_i^M)$, and the second term $U_{\boldsymbol{\theta}}^{(i)}(\mathbf{X}_i^0)$ is a potential energy function with $U_{\boldsymbol{\theta}}^{(i)}(\mathbf{X}_i^0) = \mathbf{1}^\top \mathbf{Z}_i^U \mathbf{1}$. The encodings $\mathbf{Z}_i^M$ and $\mathbf{Z}_i^U$ are calculated using the same steps (17)-(19), with parameters $\mathbf{A}_{Q,\mathbf{M}}^w, \mathbf{A}_{K,\mathbf{M}}^w, \mathbf{A}_{V,\mathbf{M}}^w$ and $\mathbf{A}_{Z,\mathbf{M}}^w$, and $\mathbf{A}_{Q,U}^w, \mathbf{A}_{K,U}^w, \mathbf{A}_{V,U}^w$ and $\mathbf{A}_U^w$, respectively. With $H_{\boldsymbol{\theta}}^{(i)}$, we obtain $\frac{\partial H_{\boldsymbol{\theta}}^{(i)}}{\partial \mathbf{x}_j}$ $\forall j \in \mathcal{N}_i^k$ and compute $\frac{\partial H_{\boldsymbol{\theta}}}{\partial \mathbf{x}_i} = \sum_{j \in \mathcal{N}_i^k} \frac{\partial H^{(j)}}{\partial \mathbf{x}_i}$.

2) Learning distributed control policies using neural ODE networks: Let $\mathsf{SA}(\mathbf{X}_i^0, \boldsymbol{\theta})$ be the operations (17)-(22) with

$$\boldsymbol{\theta} = \{\{\mathbf{A}_{Q,k}^w, \mathbf{A}_{K,k}^w, \mathbf{A}_{V,k}^w, \mathbf{A}_{Z,k}^w\}_{w=1}^{w=W}\}_{k=\{\mathbf{R},\mathbf{J},\mathbf{M},U\}}.$$

To address Problem (5), we use a neural ODE network [33] whose structure respects the continuous-time dynamics in (6). To calculate the loss $\mathcal{L}(\mathcal{D}, \bar{\mathcal{D}})$ in (4), for each trajectory l of robot i, $\{\mathbf{x}_i^l(rT)\}_{r=0}^K$ in the data, we solve an ODE:

$$\dot{\mathbf{x}}_i^l = \mathbf{f}_i(\mathbf{x}_i^l, \boldsymbol{\pi}_{\boldsymbol{\theta}}; \boldsymbol{\theta}), \quad \mathbf{x}_i^l(0) = \bar{\mathbf{x}}_i^l(0), \quad (23)$$

using an ODE solver to obtain a predicted state $\mathbf{x}_i^l(rT)$ for $i \in \mathcal{V}, l = 0, \dots, L$:

$$\mathbf{x}_i^l = \mathsf{ODESolver}\left(\mathbf{x}_i^l(0), \mathbf{f}_i, rT; \boldsymbol{\theta}\right). \quad (24)$$

The parameters $\boldsymbol{\theta}$ are updated using gradient descent by back-propagating the loss through the neural ODE solver using adjoint states $\mathbf{y}_i = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_i}$ [33]. We form an augmented state $\mathbf{z}_i = \left(\mathbf{x}_i, \mathbf{y}_i, \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}\right)$ that satisfies $\dot{\mathbf{z}}_i = \mathbf{f}_{\mathbf{z}} = \left(\mathbf{f}_i, -\mathbf{y}_i^\top \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i}, -\mathbf{y}_i^\top \frac{\partial \mathbf{f}_i}{\partial \boldsymbol{\theta}}\right)$. The gradients $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$ are obtained by solving a reverse-time ODE starting from $\mathbf{z}_i(rT) = \bar{\mathbf{z}}_i(rT)$:

$$(\mathbf{x}_i(0), \mathbf{y}_i(0), \partial \mathcal{L}/\partial \boldsymbol{\theta}) = \mathsf{ODESolver}(\mathbf{z}_i(rT), \mathbf{f}_z, rT). \quad (25)$$

We refer the reader to [33] for more details.

3) Deploying LEMURS: To deploy the control policy (15), we design a message $\mathbf{m}_{ij}(t)$, encoding information that robot i needs from robot j at time t to calculate $[\mathbf{J}_{\boldsymbol{\theta}}]_{ij}, [\mathbf{R}_{\boldsymbol{\theta}}]_{ij}$, and $H_{\boldsymbol{\theta}}^{(i)}$.

Each robot i receives a message $\mathbf{m}_{ij} = [\mathbf{m}_{ij}^{(1)}, \mathbf{m}_{ij}^{(2)}, \mathbf{m}_{ij}^{(3)}]$ in 3 communication rounds: 1) robot i receives $\mathbf{m}_{ij}^{(1)} = \mathbf{x}_j$ $\forall j \in \mathcal{N}_i^k$ and calculates $\mathbf{Z}_{ij}^J$, $\mathbf{Z}_{ij}^R$, $H_{\boldsymbol{\theta}}^{(i)}$, and $\partial H_{\boldsymbol{\theta}}^{(i)}/\partial \mathbf{x}_j$; 2) robot i receives $\mathbf{m}_{ij}^{(2)} = \partial H_{\boldsymbol{\theta}}^{(j)}/\partial \mathbf{x}_i, \mathbf{Z}_{ji}^J, \mathbf{Z}_{ij}^R$ $\forall j \in \mathcal{N}_i^k$, and calculates $\partial H_{\boldsymbol{\theta}}/\partial \mathbf{x}_i$, $[\mathbf{J}_{\boldsymbol{\theta}}]_{ij}, [\mathbf{R}_{\boldsymbol{\theta}}]_{ij}$; and 3) each robot

i receives $\mathbf{m}_{ij}^{(3)} = \partial H_{\boldsymbol{\theta}}/\partial \mathbf{x}_j \ \forall j \in \mathcal{N}_i^k$ and calculates the control input $\mathbf{u}_i$. We assume negligible delays between communication rounds. If the delay is large, Wang et al. [26] suggest to learn a function that predicts quantities such as $\partial H_{\boldsymbol{\theta}}(\mathbf{x})/\partial \mathbf{x}_j$, $\mathbf{Z}_{ji}^J$, $\mathbf{Z}_{ji}^R$, leading to one communication round. We leave this for future work. If the Hamiltonian changes slowly over sampling interval T, at time $t = rT$, robot i can use its previous neighbor states $\mathbf{x}_j((r-1)T)$ to approximate $\mathbf{m}_{ij}^{(2)}(rT)$ and $\mathbf{m}_{ij}^{(3)}(rT)$.

Example 3. *In the flocking of Examples 1-2, LEMURS can be directly applied to learn $\mathbf{D}_{\boldsymbol{\theta}}(\mathbf{p})$ and $U_{\boldsymbol{\theta}}(\mathbf{p})$. Another option is to learn $\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{R}_{\boldsymbol{\theta}}(\mathbf{x})$ and $H_{\boldsymbol{\theta}}(\mathbf{x})$, obtaining extra degrees of freedom for the control policy.*

IV. Results

In this section we evaluate LEMURS in three multi-robot tasks with simulated point robots, illustrated in Fig. 2:

1) *Fixed swapping* [15]: Robots are initialized in two columns and navigate to the diagonally opposite position in the other column while avoiding collisions (Fig. 2a). The communication graph is a fixed ring such that robot i communicates with robots $(i \pm 1) \mod n$. We use the same parameters as [15]. We generate demonstrations from the following expert controller:

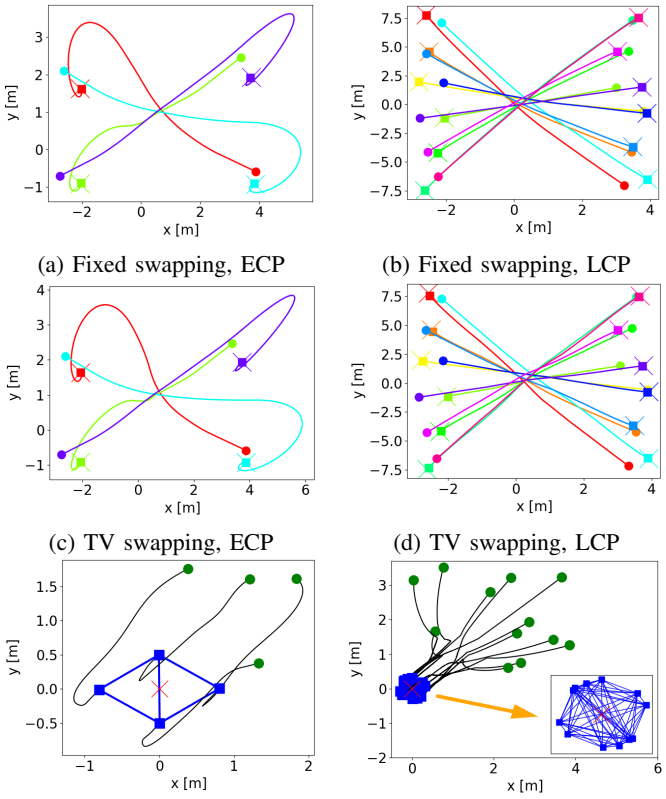
$$\mathbf{u}_i = -c_1 \mathbf{p}_i - c_2 \mathbf{v}_i - \sum_{j \in \mathcal{N}_i^1} \frac{\mathbf{p}_i - \mathbf{p}_j}{\sqrt{1 + \sigma ||\mathbf{p}_i - \mathbf{p}_j||_2^2}}, \quad (26)$$

with $c_1 = 0.8$, $c_2 = 1.0$, $\sigma = 0.1$.

2) *Time-varying swapping*: We consider the *fixed swapping* task but with a time-varying communication graph (Fig. 2c), where $\mathbf{A}(t)$ is such that $[\mathbf{A}(t)]_{ij} = \text{sigmoid}(\lambda(||\mathbf{p}_i(t) - \mathbf{p}_j(t)|| - l/2))$ if $||\mathbf{p}_i(t) - \mathbf{p}_j(t)|| < l$ and 0 otherwise, with $l = 2.4$m and $\lambda = 2.0$. We use the controller in (26) with time-varying neighbors to generate demonstrations.
3) *Flocking*: We consider the flocking task described in Examples 1-3 with parameters from [34] (Fig. 2e). We use the controller (3) to generate demonstrations.

The training and evaluation datasets for each task have $L = 400$ trajectories of $K = 250$ samples with sampling interval $T = 0.04$s. The number of demonstrating robots is $n = 4$, and the trajectories are split in sub-trajectories of 5 samples for training. We train for 10000 epochs with learning rate 0.001, and new batches of 200 samples every 100 epochs. The ODEsolver is the Euler numerical method [36]. We consider $k = 1$ as the number of hops.

The learned control policies are stable and scalable for all the tasks, converging to the desired goal with a larger number of robots as seen in Figs. 2. We plot trajectories from the expert control policy (left) and learned control policies for 12 robots (right), three times larger than the team size in training. Similar results with up to 64 robots can be found on our website[1]. For all three tasks, LEMURS achieves similar performance compared to analytical policies, which were used to generate training trajectories. LEMURS successfully captures behaviors that are not encoded a priori in the architecture of the networks nor in the cost function, such as the collision avoidance or the flock formation in *flocking*. Collision avoidance among the point robots is verified in all tasks by checking the distance between each pair of robots. In the *swapping* problems, as the training dataset is formed by sub-trajectories of 5 samples, which resemble a straight line in general, LEMURS infers that the motion to the goals should be a straight line as well. On the other hand, the minimum distance among robots is 0.005m, avoiding collisions even in the center of the stage. In flocking task, since we train LEMURS for *flocking* with only 4 robots in Fig. 2e, LEMURS infers that it is desired to have 4 groups of robots with equal distances between the groups (Fig. 2f), prioritizing formation to safety. In this sense, evaluation with 4 robots yields to a minimum distance among robots of 0.05m, while with 12 robots the minimum distance among robots is 0.001m. To improve generalization, we suggest increasing the number of robots during training, but we leave this for future work.



(a) Fixed swapping, ECP (b) Fixed swapping, LCP

(c) TV swapping, ECP (d) TV swapping, LCP

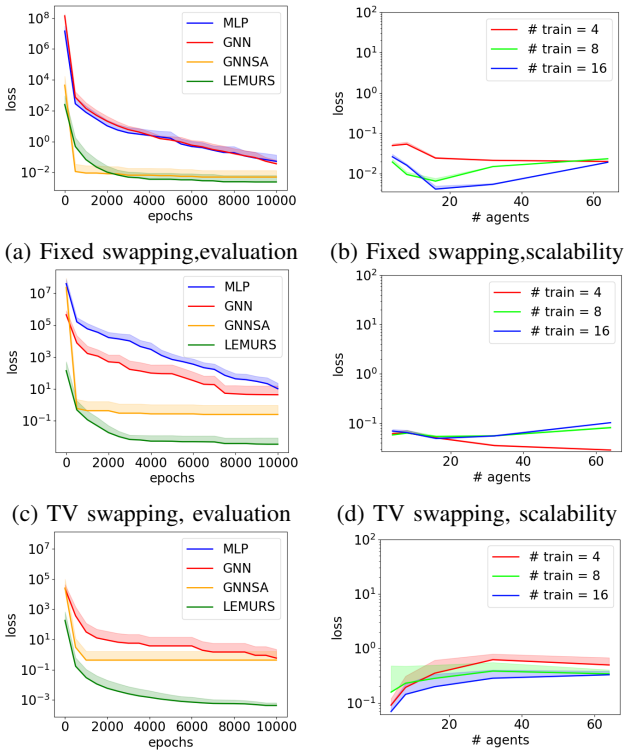
(e) Flocking, ECP (f) Flocking, LCP

Fig. 2: Demonstration of expert and learned control policies: (left) expert control policies (ECP) for training with 4 robots, (right) learned control policies (LCP) with 12 robots for 3 tasks.

We compare LEMURS with three other learning methods: 1) Multi-Layer Perceptron (MLP), inspired by [15]; 2) Graph Neural Network (GNN) from [10], [13]; and 3) Self-Attention based Graph Neural Network (GNNSA) [32], which uses graph neural networks preceded by a self-attention layer to model communication channels. These

[1] https://eduardosebastianrodriguez.github.io/LEMURS/

learning models substitute the self-attention layers in our Hamiltonian-based neural ODE networks. We keep the port-Hamiltonian neural ODE architecture for a fair comparison with the other discrete-time and/or black-box policies, leaving the complete adaptation of the other papers to our setting for future work. The size of the layers/filters in the MLP, GNN and GNNSA depends on the number of robots, so scalability is not directly achievable unlike in our approach. LEMURS has 2208 parameters while MLP, GNN and GNNSA have 1 layer/filter with $4448, 4448$ and 4672 parameters, respectively. The details are in the Appendix I.



(a) Fixed swapping,evaluation (b) Fixed swapping,scalability

(c) TV swapping, evaluation (d) TV swapping, scalability

(e) Flocking, evaluation (f) Flocking, scalability

Fig. 3: Comparison of LEMURS with a multi-layer perceptron (MLP) [15], a graph neural network (GNN) [10], [13] and a self-attention-based graph neural network (GNNSA) [32] in learning robot interactions: (left) evaluation loss over 10000 epochs, (right) LEMURS training scalability with $4, 8, 16$ robots for 3 tasks.

Fig. 3 (left) plots the evaluation loss of the 4 models and the 3 tasks, from 3 runs using 3 randomized seeds. Our self-attention architecture surpasses the other three methods in capturing interactions for all tasks with half of the number of parameters, illustrating the benefits of combining self-attention networks and Hamiltonian architecture in LEMURS. In our experiments, self-attention is shown to learn more complex aggregation patterns compared to graph neural networks, potentially because in graph neural networks the data is aggregated via a pre-multiplication of a linear graph shift operator, while SA aggregates data through Eq. (18). LEMURS achieves training loss two orders of magnitudes better than that of MLP, GNN and GNNSA in tasks with time-varying topologies. Meanwhile, the MLP training did not converge with data from the *flocking* task. For the *fixed swapping* task, LEMURS's evaluation loss is slightly better than that of GNNSA, and two orders of magnitudes better than that of MLP and GNN.

We also validate scalability. The policies are simulated over a time horizon $KT = 10$s. We train LEMURS with datasets of $n = \{4, 8, 16\}$, for 5 runs using randomized seeds, and test the learned control policies with $n = \{4, 8, 16, 32, 64\}$. The mean and standard deviation of the test loss (Eq. (4)) is normalized by n and plotted in Fig. 3 (right). LEMURS obtains similar test loss with respect to the number of training robots. In the case of *fixed swapping*, increasing the number of robots in training improves the controller performance since the larger number of robots is, the more data is available to learn about a fixed communication topology. For the *time-varying swapping* task, a small number of robots in training performs slightly better, potentially because the time-varying topology is more complex with more robots. Meanwhile, for *flocking* task, increasing the number of training robots slightly improves the performance, even though the topology is also time-varying. This is because the robots form a flocking formation in the training trajectories, leading to a fixed topology in a large portion of the dataset, similar to *fixed swapping*.

V. Conclusions

This work presented LEMURS, an algorithm that learns robot interactions from trajectory demonstrations using self-attention and Hamiltonian-based neural ODE networks. LEMURS advances the state of the art by learning control policies that generalize to increasing numbers of robots and time-varying communications. Our evaluation shows that LEMURS learns behaviors such as collision avoidance and flocking formation from state-only trajectories of few robots, and successfully replicates the tasks in larger robot teams.

Appendix I
Network and Experiment Parameters

The SA architecture is parameterized as follows:

- $[\mathbf{R}_{\boldsymbol{\theta}}]_{ij}$: $W = 3$, $h_w = [4, 8, 8]$, $r_w = [8, 8, 8]$, $d_w = [8, 8, 16]$; functions $\beta = \mathsf{sigmoid}$, $\gamma = \alpha = \mathsf{swish}$ [37].
- $[\mathbf{J}_{\boldsymbol{\theta}}]_{ij}$: $W = 3$, $h_w = [4, 8, 8]$, $r_w = [8, 8, 8]$, $d_w = [8, 8, 1]$; functions $\beta = \mathsf{sigmoid}$, $\gamma = \alpha = \mathsf{swish}$ [37]; and $[\mathbf{J}_{\boldsymbol{\theta}}]_{ij} = \mathbf{0}\ \forall i \neq j$.
- $H^i_{\boldsymbol{\theta}}$: $W = 3$ layers, $h_w = [6, 8, 8]$, $r_w = [8, 8, 8]$, $d_w = [8, 8, 25]$; functions $\beta = \mathsf{sigmoid}$, $\gamma = \alpha = \mathsf{swish}$ [37].

The network input $\mathbf{X}^0_i$ is an offset version of (16) as follows:

- $[\mathbf{R}_{\boldsymbol{\theta}}]_{ij}$ and $[\mathbf{J}_{\boldsymbol{\theta}}]_{ij}$: $\mathbf{X}^0_i = \left[\Delta\mathbf{x}_{ii}, \{\Delta\mathbf{x}_{ij}\}_{j\in\mathcal{N}^k_i\setminus\{i\}}\right]$.
- $H^i_{\boldsymbol{\theta}}$:

$$\mathbf{X}^0_i = \left[\{\Delta\mathbf{x}_{ii}, 0, 0\}, \{\Delta\mathbf{x}_{ij}, \|\Delta\mathbf{x}_{ij}\|_2^{\frac{1}{4}}, \|\Delta\mathbf{x}_{ij}\|_2\}_{j\in\mathcal{N}^k_i\setminus\{i\}}\right],$$

where $\Delta\mathbf{x}_{ii} = \mathbf{x}_i - \bar{\mathbf{x}}_i(KT)$ and $\Delta\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$.

The other networks are as follows. For the MLP and GNN, $W = 1$ and $(4\times n)\times(4\times 4\times n)$ parameters from 1 unbaised layer/filter; for H, $W = 1$ and $(6\times n)\times(5\times 5\times n)$ parameters from 1 unbaised layer/filter. The GNNSA has $3\times(4\times 8)+(8\times 16)$ additional parameters from three self-attention matrices and one self-attention vector.

REFERENCES

[1] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot SLAM," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 4775–4782.

[2] Y. Tian, Y. Chang, F. H. Arias, C. Nieto-Granda, J. P. How, and L. Carlone, "Kimera-multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems," *IEEE Transactions on Robotics*, 2022.

[3] X. Kan, T. C. Thayer, S. Carpin, and K. Karydis, "Task planning on stochastic aisle graphs for precision agriculture," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3287–3294, 2021.

[4] A. Pierson and M. Schwager, "Bio-inspired non-cooperative multi-robot herding," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 1843–1849.

[5] E. Sebastián and E. Montijano, "Multi-robot implicit control of herds," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 1601–1607.

[6] E. Sebastián, E. Montijano, and C. Sagüés, "Adaptive multirobot implicit control of heterogeneous herds," *IEEE Transactions on Robotics*, 2022.

[7] L. Heintzman, A. Hashimoto, N. Abaid, and R. K. Williams, "Anticipatory planning and dynamic lost person models for human-robot search and rescue," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 8252–8258.

[8] T. Z. Jiahao, L. Pan, and M. A. Hsieh, "Learning to swarm with knowledge-based neural ordinary differential equations," in *IEEE International Conference on Robotics and Automation*, 2022, pp. 6912–6918.

[9] D. Bloembergen, K. Tuyls, D. Hennes, and M. Kaisers, "Evolutionary dynamics of multi-agent learning: A survey," *Journal of Artificial Intelligence Research*, vol. 53, pp. 659–697, 2015.

[10] A. Khan, E. Tolstaya, A. Ribeiro, and V. Kumar, "Graph policy gradients for large scale robot control," in *Conference on Robot Learning*, 2020, pp. 823–834.

[11] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, "Learning decentralized controllers for robot swarms with graph neural networks," in *Conference on Robot Learning*, 2020, pp. 671–682.

[12] E. Tolstaya, J. Paulos, V. Kumar, and A. Ribeiro, "Multi-robot coverage and exploration using spatial graph neural networks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021, pp. 8944–8950.

[13] F. Yang and N. Matni, "Communication topology co-design in graph recurrent neural network based distributed control," in *IEEE Conference on Decision and Control*, 2021, pp. 3619–3626.

[14] F. Gama, Q. Li, E. Tolstaya, A. Prorok, and A. Ribeiro, "Synthesizing decentralized controllers with graph neural networks and imitation learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1932–1946, 2022.

[15] L. Furieri, C. L. Galimberti, M. Zakwan, and G. Ferrari-Trecate, "Distributed neural network control with dependability guarantees: a compositional port-hamiltonian approach," in *Learning for Dynamics and Control Conference*, 2022, pp. 571–583.

[16] R. Han, S. Chen, and Q. Hao, "Cooperative multi-robot navigation in dynamic environment with deep reinforcement learning," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 448–454.

[17] G. Shi, W. Hönig, Y. Yue, and S.-J. Chung, "Neural-swarm: Decentralized close-proximity multirotor control using learned interactions," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 3241–3247.

[18] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 6252–6259.

[19] S. H. Semnani, H. Liu, M. Everett, A. De Ruiter, and J. P. How, "Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3221–3226, 2020.

[20] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn, "Robonet: Large-scale multi-robot learning," in *Conference on Robot Learning*, 2020, pp. 885–897.

[21] K. Bogert and P. Doshi, "Multi-robot inverse reinforcement learning under occlusion with estimation of state transitions," *Artificial Intelligence*, vol. 263, pp. 46–73, 2018.

[22] H. Zhu, F. M. Claramunt, B. Brito, and J. Alonso-Mora, "Learning interaction-aware trajectory predictions for decentralized multi-robot motion planning in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2256–2263, 2021.

[23] S. Zhou, M. J. Phielipp, J. A. Sefair, S. I. Walker, and H. B. Amor, "Clone swarms: Learning to predict and control multi-robot systems by imitation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 4092–4099.

[24] G. Qu, A. Wierman, and N. Li, "Scalable reinforcement learning of localized policies for multi-agent networked systems," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 256–266.

[25] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5571–5580.

[26] 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.

[27] A. Y. Ng, S. Russell, *et al.*, "Algorithms for inverse reinforcement learning," in *International Conference on Machine Learning*, vol. 1, 2000, p. 2.

[28] A. Van Der Schaft and D. Jeltsema, "Port-Hamiltonian systems theory: An introductory overview," *Foundations and Trends in Systems and Control*, vol. 1, no. 2-3, pp. 173–378, 2014.

[29] C. L. Galimberti, L. Furieri, L. Xu, and G. Ferrari-Trecate, "Hamiltonian deep neural networks guaranteeing non-vanishing gradients by design," *arXiv preprint arXiv:2105.13205*, 2021.

[30] 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.

[31] Q. Long, Z. Zhou, A. Gupta, F. Fang, Y. Wu, and X. Wang, "Evolutionary population curriculum for scaling multi-agent reinforcement learning," in *International Conference on Learning Representations*, 2020.

[32] Q. Li, W. Lin, Z. Liu, and A. Prorok, "Message-aware graph attention networks for large-scale multi-robot path planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5533–5540, 2021.

[33] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[34] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[35] G. Blankenstein, R. Ortega, and A. J. Van Der Schaft, "The matching conditions of controlled lagrangians and ida-passivity based control," *International Journal of Control*, vol. 75, no. 9, pp. 645–665, 2002.

[36] J. C. Butcher, *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[37] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.