Cooperative Control of Mobile Robots with Stackelberg Learning

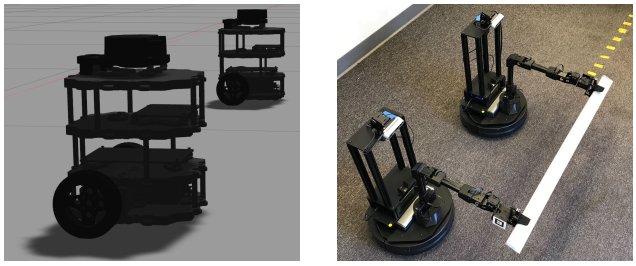
Joewie J. Koh*, Guohui Ding*, Christoffer Heckman, Lijun Chen, Alessandro Roncone

***Abstract*— Multi-robot cooperation requires agents to make decisions that are consistent with the shared goal without disregarding action-specific preferences that might arise from asymmetry in capabilities and individual objectives. To accomplish this goal, we propose a method named SLiCC: Stackelberg Learning in Cooperative Control. SLiCC models the problem as a partially observable stochastic game composed of Stackelberg bimatrix games, and uses deep reinforcement learning to obtain the payoff matrices associated with these games. Appropriate cooperative actions are then selected with the derived Stackelberg equilibria. Using a bi-robot cooperative object transportation problem, we validate the performance of SLiCC against centralized multi-agent Q-learning and demonstrate that SLiCC achieves better combined utility.**

I. Introduction

Robotics at large has been improving at a rapid pace, and this has resulted in increased demand in applications ranging from manufacturing [1], to warehousing [2], to human-populated environments [3]. However, despite the clear potential for distributed controllers that leverage cooperation between multiple mobile robots (e.g., the cooperative transport of large or heavy objects, see Fig. 1b), the vast majority of existing techniques are either limited to single-robot operation or require that each robot perceive the complete state of the environment [4]. Interestingly enough, model-free learning-based methods present a promising alternative to traditional model-based control, in that they are less reliant on domain knowledge such as kinematic and dynamic modeling of the system, and that they scale more naturally with the number of agents.

Notably, reinforcement learning (RL) shows promise for controlling robotic systems in unstructured environments as it enables robots to discover useful behaviors simply by interacting with the environment [5]. However, research in the field is still at an early stage, and state-of-the-art methods are often limited by issues such as partial observability, particularly in multi-agent settings [6]. Learning cooperative control of multi-robot systems to achieve common objectives can be difficult in practice due to agents having asymmetric embodiments and capabilities; for example, robot participants can be limited by perceptual [7], communicative [8], locomotive [9], or computational [10] constraints to different extents. Additionally, agent-specific preferences in a cooperative setting can inadvertently interfere with the successful accomplishment of the common objective, leading to unstable learning in naïve multi-agent adaptations of single-agent RL methods. To help agents make appropriate decisions in these circumstances, it is critical to utilize well-assigned cooperation responsibilities without disregarding agent-specific preferences—this motivates the use of negotiated decision-making processes.



(a) Two TurtleBot3 Burger mobile robots in the Gazebo simulator.

(b) Two LoCoBot mobile robots cooperatively transport an object.

Fig. 1: SLiCC is a novel method designed for cooperative control of bi-robot systems. In this work, we evaluate the performance of SLiCC using a pair of simulated robots and another pair of real robots. The two robots were selected for their different physical dynamics.

In this paper, we introduce SLiCC: Stackelberg Learning in Cooperative Control, which uses a novel prosocial–introspective framework to achieve a common goal in a bi-agent system. The proposed framework allows for agents to have different observation scopes, with prosocial and introspective behaviors assigned to agents based on the completeness of their state perception.

The main idea behind the prosocial–introspective framework is to provide a link between perception asymmetry and agent utility, in the context of cooperative control. More specifically, in order to successfully achieve a shared goal (e.g., two robots tasked with jointly carrying an object), the efficient exploitation of inter-agent interaction dynamics and consequent allocation of learning responsibilities can compensate for partial observability that arises as a consequence of perception asymmetry.

Deriving these interaction dynamics requires additional perceptual capabilities, either in terms of perceiving the complete state (e.g., the positions of both robots) or interaction forces (e.g., tension or compression on jointly-carried objects). Therefore, the utility of the agent with such additional perceptual capabilities should be dependent on both agents' decisions; we refer to this agent as being *prosocial*—its

This material is based upon work supported by the National Science Foundation under Grant No. 1646556. The authors also acknowledge support from a "PyRobot: Democratizing Robotics" Research Award from Facebook Research. *(Joewie J. Koh and Guohui Ding contributed equally to this work.)*

All authors are with the Department of Computer Science, University of Colorado Boulder. {firstname.lastname}@colorado.edu

behavior is largely motivated by concerns about obeying constraints of the common goal. On the other hand, the agent with only partial state perception has utility that is only dependent on its own decisions; we therefore refer to this as the *introspective* agent—its attention is exclusively on its own state and utility.

More specifically, SLiCC models the prosocial–introspective cooperation problem as a partially observable stochastic game (POSG) composed of Stackelberg bimatrix games. Decomposing a POSG into Stackelberg bimatrix games allows the use of the Stackelberg equilibrium at each decision step to approximate the POSG equilibrium. With the insight that an agent's payoff function is equivalent to their Q-value function, we use deep RL to learn payoff matrices, capitalizing on the function approximation capabilities of deep learning to do so even in settings with continuous state spaces. The Stackelberg equilibrium can then be derived from the agents' payoff matrices and used to inform the agents' actions, serving as a mechanism for negotiated decision-making.

Our contributions are as follows:

1) Introduce SLiCC, a method for cooperative control of bi-agent systems in partially observable settings based on an asymmetric prosocial–introspective cooperation framework that links state perception with agents' decision-making strategies.
2) Provide an improved Stackelberg game–based architecture to enhance the agents' policy learning capabilities under POSG settings.
3) Demonstrate that a SLiCC policy learned in simulation can be used to control real robots without additional training.

The rest of the paper has the following structure. To begin with, Section II presents an overview of the relevant literature. Following a description of the problem setting in Section III, we elucidate the proposed SLiCC method in Section IV: we begin with a brief treatment of POSGs, and then explain how we bridge the POSG setting and our prosocial–introspective framework using the Stackelberg equilibrium. Next, Section V discusses key differences of SLiCC compared with other learning paradigms. The advantages of SLiCC detailed in this section are consequential in real-world applications of cooperative control. Section VI follows with our experimental setup and results. Besides evaluating SLiCC in simulation, we also validate our proposed method with real robots. Finally, Section VII concludes the paper and provides an inventory of future directions.

II. Related Work

Multi-robot cooperative control has traditionally been pursued from the perspective of control theory [11]–[13]. Indeed, multi-agent scenarios present peculiar challenges and constraints for RL-based solutions [6], which has limited their relevance for the field despite recent successes of deep RL. Methods such as policy search enable robots to learn complex real-world skills in single-agent settings (e.g., door opening [14] and map-less navigation [15]). Unfortunately, it is difficult to extend these methods to multi-robot applications. Doing so with a centralized learning paradigm leads to exponentially increasing state and action spaces [16], and consequently, infeasible computational needs. Conversely, agent decentralization–based RL approaches are not guaranteed to result in stable policies [17]. However, there is a promising line of research with game-theoretic RL: policy updates can be based on stochastic game equilibria, with the goal of improving state-value function estimates and reducing learning instability. This has been demonstrated as an effective solution that incorporates agent–agent interaction in multi-agent RL [18]–[20], and these approaches are seeing increasing interest across different communities [21]–[23].

Nonetheless, this direction is not without limitations. Stochastic games typically assume that all agents have complete observations [18]–[20], which is a limiting constraint for some settings—and in particular for multi-agent applications. This point provides an opportune segue into partially observable stochastic games (POSGs), which arise naturally from asymmetry in agent-specific preferences, cooperation responsibilities, or observation scopes. In previous work, the POSG learning process has been approximated with a series of Bayesian games and common payoffs, using belief spaces to compensate for missing observations [24]. Methods like Joint Equilibrium–based Search for Policies (JESP) [25] have also been used to alternately optimize the agents' objective. Improved flexibility in the decentralized learning process has furthermore been achieved via modeling an opponent's policy through recursive reasoning [26].

These works motivate us to further consider how to coordinate asymmetric agent roles engendered by robots with different perceptual capabilities. Previous research in this area attempted to do so by incorporating Stackelberg games into RL techniques [27]–[29]. Similar approaches have been applied in human–robot interaction [21] and smart microgrids [30], but they dictate a common state space for all agents. Inspired by recent work applying Stackelberg games to partially observable scenarios [31], we present a novel improvement for robot learning by integrating Stackelberg games with POSGs and reducing the prerequisite of complete state observations through our prosocial–introspective framework. In order to optimize the performance of game-theoretic RL approaches, we investigate agent roles in a multi-robot cooperation problem with asymmetric information. In the rest of the paper, we demonstrate the potential of game-theoretic RL approaches with SLiCC as an exemplar.

III. Problem Setting and Generalizability

For convenience, in this work we consider a bi-agent system composed of two mobile robots—although the proposed method can generally be applied to a variety of multi-robot scenarios. The state of each robot is denoted with a tuple $(x^k, y^k, \theta^k, v^k)$, where x^k and y^k are the robot's coordinates in the 2-dimensional plane, and θ^k is the robot's orientation about the z-axis. Furthermore, the magnitude of the agent's linear velocity v^k in the 2-dimensional plane is given by the L_2-norm of the time derivatives of x^k and y^k.

Accordingly, with $a_t^{\omega,k}$ and $a_t^{v,k}$ being the actionable increments for angular velocity and linear velocity respectively, the robots' dynamics in discrete space (which is the same as that which is used by the RL agents) is given by:

$$x_{t+1}^k = x_t^k + v_t^k \cos(\theta_t^k)\Delta t + \epsilon_{x^k}, \quad (1)$$
$$y_{t+1}^k = y_t^k + v_t^k \sin(\theta_t^k)\Delta t + \epsilon_{y^k}, \quad (2)$$
$$\theta_{t+1}^k = \theta_t^k + a_t^{\omega,k}\Delta t + \epsilon_{\theta^k}, \quad (3)$$
$$v_{t+1}^k = v_t^k + a_t^{v,k} + \epsilon_{v^k}, \quad (4)$$

where $\epsilon_{(\cdot)}$ denotes environment-induced noise (e.g., wheel-spin) and errors incurred from the discretization of the dynamics. The robots are given the objective of cooperatively transporting an object that is beyond the capabilities of a single robot to carry (see Fig. 1b).

We selected this particular task because it introduces dynamical constraints that are not explicitly known by all agents, and it is furthermore a convenient setting to investigate perception asymmetry. In our setting, the first robot (i.e., the prosocial agent) can perceive both robots' state tuples, but the second robot (i.e., the introspective agent) can only perceive its own state tuple. We also assume that the prosocial agent can receive information from the introspective agent at each decision step.

We elected to demonstrate the merits of our prosocial–introspective framework with this simple bi-agent system as it is representative of imbalanced agent roles in a multi-agent setting. However, it is worthwhile to note that SLiCC is applicable too in systems with more than two agents: we can easily add more introspective agents without loss of generality.

IV. Stackelberg Learning in Cooperative Control

In this section, we present the main components of SLiCC (see Fig. 2 for an outline of the method). As previously mentioned, we use the POSG model to characterize the policy learning process due to the difference in observation scopes between the two agents. The prosocial–introspective framework allows us to relate this difference to agent utility, while step-wise Stackelberg equilibria are used to approximate the POSG equilibrium. Finally, we describe how deep Q-networks can be used to estimate the agents' Q-values.

A. Partially Observable Stochastic Game (POSG)

A POSG can be represented by a tuple $\mathcal{G}$, where

$$\mathcal{G} = \left\{\mathcal{N}, S, \left(O^k\right)_{k\in\mathcal{N}}, \left(A^k\right)_{k\in\mathcal{N}}, T, \left(r^k\right)_{k\in\mathcal{N}}, \left(\pi^k\right)_{k\in\mathcal{N}}\right\}.$$

The elements of $\mathcal{G}$ are defined as follows:

- $\mathcal{N} = \{1, \ldots, n\}$ is the set of agents;
- S is the set of states, common to all agents;
- $O^k \subseteq S$ is the observation space for agent k;
- A^k is the set of actions available to agent k;
- $T\colon S \times A^1 \times \cdots \times A^n \to PD(S)$ is the state transition function, with $PD(\cdot)$ being a probability distribution, and with the transition dynamics common to all agents;

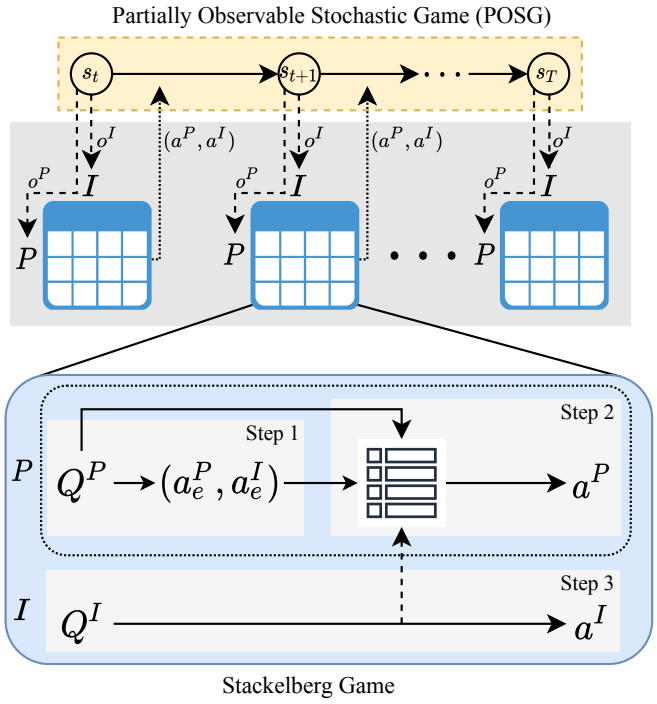


Fig. 2: Outline of the SLiCC algorithm. (P: Prosocial Agent, I: Introspective Agent.) Each decision step t of the POSG corresponds to a Stackelberg game (see Section IV-C). The Q-table of each agent is estimated based on its observations. Step 1: Agent P determines the expected action pair (a_e^P, a_e^I) based on its own Q-table; Step 2: Agent P derives the actual action a^P; Step 3: Agent I determines its actual action a^I.

- $r^k\colon O^k \times A^1 \times \cdots \times A^n \to \mathbb{R}$ is the reward function for agent k; and
- $\pi^k\colon O^k \to PD(A^k)$ is the policy of agent k.

As detailed in Section III, in this work we consider a bi-agent system consisting of a prosocial agent and an introspective agent, i.e., $\mathcal{N} = \{P, I\}$. Therefore, $\forall o \in O^k$, the Stackelberg equilibrium (π^{P^*}, π^{I^*}) [27] satisfies Eqs. (5) and (6), for all π^P and π^I respectively:

$$V^P\left(o\middle|\pi^{P^*}, \pi^{I^*}\right) \geq V^P\left(o\middle|\pi^P, BR(V^I, \pi^P)\right), \quad (5)$$
$$V^I\left(o\middle|\pi^{P^*}, \pi^{I^*}\right) \geq V^I\left(o\middle|\pi^{P^*}, \pi^I\right). \quad (6)$$

Here, $BR(V^I, \pi^P) = \arg\max_{u^I \in A^I} V^I\left(o\middle|\pi^P, u^I\right)$ represents the best response of Agent I given Agent P's policy. For $k \in \mathcal{N}$, $V^k(\cdot)$ refers to the discounted episode reward with observation o at time t,

$$V^k(o) = \mathbf{E}\left[\sum_{i=0}^{T-t} \gamma^i r_{t+i}^k \,\middle|\, o_t = o, (\pi^P, \pi^I)\right]. \quad (7)$$

Eqs. (5) and (6) essentially state that, when agent k follows an equilibrium policy π^{k^*}, the discounted episode reward attained is guaranteed to be greater than or equal to that attained with any other policy. Agent P's decision is also related to the best response of Agent I. In our problem, we only consider stationary policies $\left(\pi^k = \pi^k(s^1), \ldots\right)$.

B. Prosocial–Introspective Framework

Under the POSG setting, each agent makes decisions based on its observations. The prosocial agent (Agent P) has complete observation of the system (i.e. perfect perception), but the introspective agent (Agent I) only observes its own state. Specifically for our problem setting, the state of agent k is $s^k = (x^k, y^k, \theta^k, v^k)$. Thus, the observations of the two agents are $o^P = (s^P, s^I)$ and $o^I = s^I$. The reward function of each agent depends on its observation and action spaces: $r^k : O^k \times A^k \to \mathbb{R}$.

C. Stackelberg Game

We use a Stackelberg game formulation as the intermediary between our prosocial–introspective framework and the POSG model, as shown in Fig. 2. We express each decision step of the POSG as a partially observed bimatrix game $\mathcal{G}^{SG}$, where

$$\mathcal{G}^{SG}(s) = \left\{\mathcal{N}, (Q^k(\cdot))_{k\in\mathcal{N}}, (\pi^k)_{k\in\mathcal{N}}, s\right\}. \tag{8}$$

Here, Q^k refers to the payoff matrix of Agent k. Note that an agent's payoff matrix is also its Q-table: respectively, $Q^P\left(o^P, a^P, a^I\right)$ and $Q^I\left(o^I, a^I\right)$ are the Q-values for the Agents P and I when observing o^P and o^I [28]. To derive a Stackelberg equilibrium solution and actions at each decision step t, the agents use the following steps:

1) Agent P, using its own Q^P, determines the expected action pair that maximizes Q^P:
$$(a_e^P, a_e^I) = \underset{\substack{u^P\in A^P\\ u^I\in A^I}}{\arg\max}\, Q^P(o_t^P, u^P, u^I). \tag{9}$$
2) Agent P receives Q^I from Agent I, and derives an actual action a^P by minimizing the difference between the Q-values obtained from the actual and expected action pairs:
$$a^P = \underset{u^P\in A^P}{\arg\min} |Q^P(o_t^P, u^P, \underset{u^I\in A^I}{\arg\max}\, Q^I(o_t^I, u^I)) - Q^P(o_t^P, a_e^P, a_e^I)|. \tag{10}$$
3) Agent I, using its own Q^I, determines its actual action a^I that maximizes Q^I:
$$a^I = \underset{u^I\in A^I}{\arg\max}\, Q^I(o^I, u^I). \tag{11}$$

D. Algorithm Overview

The SLiCC method is described in Algorithm 1. First, both Agent P and Agent I initialize their respective neural networks to approximate Q^P and Q^I respectively. Then, during each episode, the agents use the Stackelberg equilibrium to guide their action choices, i.e. using Eqs. (10) and (11). Finally, the agents update their Q-value functions:

$$Q_{t+1}^P\left(o_t^P, a^P, a^I\right) = (1-\alpha_t)\cdot Q_t^P\left(o_t^P, a^P, a^I\right) + \alpha_t\left[r_t^P + \gamma \max_{u^P\in A^P} Q_t^P\left(o_{t+1}^P, u^P, F_t^I(o_{t+1}^I)\right)\right], \tag{12}$$

$$Q_{t+1}^I\left(o_t^I, a^I\right) = (1-\alpha_t)\cdot Q_t^I\left(o_t^I, a^I\right) + \alpha_t\left[r_t^I + \gamma \max_{u^I\in A^I} Q_t^I\left(o_{t+1}^I, u^I\right)\right]. \tag{13}$$

For conciseness of notation in Eq. (12), we have defined:

$$F_t^I(x) = \underset{u^I\in A^I}{\arg\max}\, Q_t^I(x, u^I). \tag{14}$$

From Eq. (13), we see that the Agent I's policy learning does not depend on Agent P's actions. Conversely, Eq. (12) shows that Agent P learns to adapt to Agent I during its Q-value updates. This reinforces our interpretation of the prosocial–introspective framework: the introspective agent only focuses on its own state and utility, while the prosocial agent ensures the common objective is fulfilled by making adaptive decisions and reconciling actions taken by the introspective agent. Ultimately, the approach allows us to mediate the agents' perception asymmetry by means of appropriately assigning cooperation responsibilities.

V. Comparison with Other Learning Paradigms

We now discuss several key aspects in which SLiCC differs from other learning paradigms. We address them with regard to two main themes: perceptual or communicative requirements, and time-varying agent-specific preferences. These have important implications when considering the design specification and adaptability of a multi-robot system.

A. Perceptual or Communicative Requirements

SLiCC places less emphasis on the capabilities of individual agents compared to other frameworks. As previously mentioned, SLiCC does not require all agents to have complete state observations. This means that we can avoid equipping introspective agents with the full suite of sensors typically required for the task. Alternatively, we can reduce the required communication bandwidth as agents no longer have to be in constant two-way communications with each other to share state information. Furthermore, SLiCC operates with full effectiveness using one-way communications: introspective agents do not need to receive information from any other agent.

B. Time-Varying Agent-Specific Preferences

We consider here a situation that can further clarify some advantages of the proposed SLiCC method. Imagine a scenario where an agent suffers a mechanical malfunction, therefore impeding its ability to carry out a certain action as was originally determined at the design step of the problem. In such a scenario, it would be beneficial for the agent to modify its policy to accommodate its diminished capabilities.

1) Independent Learning: With independent learning, each agent disregards the existence of the other agents, subsuming them within the environment. This approach requires no inter-agent communication, and agents seek to maximize their individual utilities. While independent learning might allow to learn satisfactory policies in some multi-agent settings, it is expected to fail in this scenario given the additional agent–agent dynamics in the environment.

Algorithm 1 Stackelberg Learning in Cooperative Control (SLiCC)

Initialize neural networks $Q^k(\cdot;\theta^k)$ with corresponding output size ($P : |A^P| \times |A^I|$, $I : |A^I|$) and replay buffer $\mathcal{D}$.
for episode $1 : M$ **do**
  Initialize state s_0 for all agents.
  for $t = 1, \ldots, T$ **do**
    P derives expected actions $(a_e^P, a_e^I) = \arg\max_{u^P, u^I} Q^P(o_t^P, u^P, u^I; \theta^P)$.
    P and I simultaneously ϵ-greedily execute a^P and a^I respectively:

$$\begin{cases} a^P = \arg\min_{u^P} |Q^P(o_t^P, u^P, \arg\max_{u^I} Q^I(o_t^I, u^I)) - Q^P(o_t^P, a_e^P, a_e^I)|, \\ a^I = \arg\max_{u^I} Q^I(o^I, u^I). \end{cases}$$

    Both agents add their corresponding transition tuple $\tau_t^k = (o_t^k, a_t^k, r_t^k, o_{t+1}^k)$ to the replay buffer $\mathcal{D}$.
    for $(\tau^P, \tau^I)_i$ in the mini-batch sample from $\mathcal{D}$ **do**
      P calculates target as follows: $\ell_i^P = \begin{cases} r^P, & \text{if episode terminates;} \\ r^P + \gamma \max_{u^P} Q^P(o^{P'}, u^P, F_t^I(o^{I'}); \theta^P), & \text{otherwise (see Eq. (14)).} \end{cases}$
      I calculates target as follows: $\ell_i^I = \begin{cases} r^I, & \text{if episode terminates;} \\ r^I + \gamma \max_{u^I} Q^I(o^{I'}, u^I; \theta^I), & \text{otherwise.} \end{cases}$
      Update θ^k by minimizing the cost $(\ell_i^k - Q^k((o^k, a^k)_i; \theta^k))^2$.
    end for
  end for
end for

2) Centralized Learning: Centralized learning utilizes a single policy to jointly control all agents, managing the multi-agent system as if it were a single complex agent. This approach is, in theory, capable of adapting to this scenario in two ways. First, the learning process can be designed a priori to accommodate such an eventuality, by introducing agent-specific preferences explicitly in the state space; however, doing so for all agents quickly leads to combinatorial explosion. Second, the centralized policy can adapt to the malfunctioning robot's modified behavior and action space through learning. Unfortunately, since the centralized policy's action space is the Cartesian product of all agents' action spaces, it will take a long time to converge to the new policy.

3) SLiCC: Introspective agents in SLiCC are able to implicitly convey their updated agent-specific preferences in the form of Q^I. Following a change in its agent-specific preferences, an introspective agent can quickly update its Q^I since the dimensionality of Q^I is simply equal to the dimensionality of its action space. This stands in direct contrast to the update situation with centralized learning described above. Upon receiving an updated Q^I, Agent P can immediately accommodate the changes in Agent I's agent-specific preferences (see Eq. (10)).

VI. Experiments and Evaluations

In this section, we report on the results of our evaluation of SLiCC with two different reward prototypes. We use SLiCC to learn a policy for the cooperative transport task in the Gazebo robotics simulator with a pair of TurtleBot3 Burger mobile robots (Fig. 6a). As a baseline for comparison, we also learn a centralized policy (see Section V-B.2) with a deep Q-network [32], referred to hereinafter as centralized Q-learning. Our code is publicly available on GitHub[1], and a summary video of our experiments can be accessed at https://youtu.be/NnuhFeVTcOw.

A. Reward Structure

In our problem, the different characteristics of the agents provide an advantageous scenario to design reward functions to cover multiple learning responsibilities. Specifically, one of the main advantages for our game-theoretic RL approach is the inclusion of agent-specific preferences [19], denoted as r_{ap}. This component characterizes agent-based concerns, preferences, or desires, which are not affected by the other agents or the environment. In addition, a multi-agent system also needs to carefully balance agent–agent coordination with the achievement of the common goal—which we hereinafter refer to as r_{int} and r_{goal} respectively.

Considering that the prosocial agent has a broader observation scope, it can respond to the introspective agent's possibly unexpected behaviors, without disregarding its agent-specific preferences or the common goal. We have the following reward prototypes:

1) $\mathbf{RP}_\alpha$: The prosocial agent incorporates the global goal, agent–agent interaction, and its agent-specific preferences. The introspective agent focuses on the global goal and its agent-specific preferences.

$$r_t^P(o_t^P, a_t^P) = r_{goal}^P + r_{int} + r_{ap}^P, \tag{15}$$
$$r_t^I(o_t^I, a_t^I) = r_{goal}^I + r_{ap}^I. \tag{16}$$

2) $\mathbf{RP}_\beta$: The prosocial agent incorporates agent–agent interaction and its agent-specific preferences. The introspective agent focuses on the global goal and its

[1] https://github.com/HIRO-group/SLiCC

agent-specific preferences.

$$r_t^P(o_t^P, a_t^P) = r_{int} + r_{ap}^P, \tag{17}$$

$$r_t^I(o_t^I, a_t^I) = r_{goal}^I + r_{ap}^I. \tag{18}$$

In these reward prototypes, r_{int} evaluates the interaction between P and I: a higher value implies that the two robots are staying in formation, as is required for cooperative transport. This term does not contain information about the global target. Meanwhile, r_{goal} quantifies the discrepancy between the current state and the expected goal state. Since the two agents cooperatively complete one task, the global goal can be defined based on individual states. Finally, r_{ap}^k represents the agents' preference for smooth consecutive actions: drastic changes in robot commands might result in undesirable behaviors (e.g., loss of traction).

$\mathbf{RP}_\alpha$ and $\mathbf{RP}_\beta$ present two different cooperation plans of the prosocial agent. Compared to $\mathbf{RP}_\alpha$, $\mathbf{RP}_\beta$ reduces the prosocial agent's duty towards accomplishing the common goal and allows it to focus on how to negotiate its agent-specific preferences and the agent–agent interactions, while trusting that the limited information available to the introspective agent will be enough to accomplish the task. Although the consideration of r_{ap}^k introduces additional complexity in the reward landscape and thus complicates the policy learning process, the equilibrium-based policy updates of SLiCC provide flexibility to make appropriate decisions.

In our experiment, the target state is $s^{target} = (x^{target}, y^{target}, v^{target}, \theta^{target})$. Based on the problem setting and our testbed, we define $r_{ap} = (r_{ap}^P, r_{ap}^I)$, r_{int}, and $r_{goal} = (r_{goal}^P, r_{goal}^I)$ as follows:

$$r_{int} = -\|\|(x_t^P, y_t^P) - (x_t^I, y_t^I)\| - \sigma\|, \tag{19}$$

$$r_{goal}^k = -\|(v, \theta)_t^k - (v, \theta)^{target}\|, \tag{20}$$

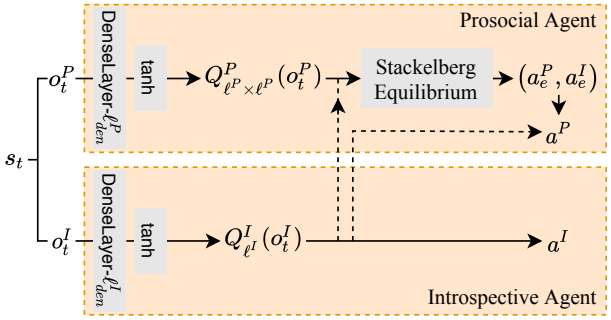
$$r_{ap}^k = \begin{cases} \mu^{upper}, & \text{if } \|a_t^{v,k} - a_{t-1}^{v,k}\| \le \zeta, \\ -\mu^{lower}, & \text{else.} \end{cases} \tag{21}$$

Here $\mu^{upper} > 0$, $\mu^{lower} > 0$, and $\zeta > 0$. The desired distance between the two agents is denoted as σ.

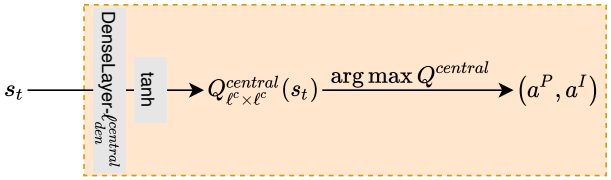
B. Training Architecture

The neural network structure for Algorithm 1 is shown in Fig. 3a. Each RL agent has a fully connected neural network with a single hidden dense layer containing ℓ_{den}^k neurons. The input layer of each network has the same dimensionality as its corresponding agent's observation o^k. The output size of each neural network has the same dimensionality as the corresponding agent's Q-table ($P : \ell^P \times \ell^P, I : \ell^I$). Based on the requirements of our problem setting and testbed, $\ell^P = \ell^I = 9$ and $\ell_{den}^k = 1024$. Relevant hyperparameters used in learning are: $\gamma = 0.95$, and $|\mathcal{D}_{\min}| = 64$. The parameters used in the reward are: $\mu^{upper} = 0.05$, $\mu^{lower} = 0.02$, and $\zeta = 0.03$. These values were selected via hyperparameter search, but our empirical experience suggests that SLiCC is relatively robust to reasonable changes in hyperparameters.

The action space is $\mathbf{A}^{type} = \{0, -2\Delta a^v, -\Delta a^v, \Delta a^v, 2\Delta a^v, -2\Delta a^\theta, -\Delta a^\theta, \Delta a^\theta, 2\Delta a^\theta\}$. For each agent, the actual action at each decision step is an angular velocity change ($a_t^{\omega,k} \in \mathbf{A}^{type}$) or a linear velocity increment ($a_t^{v,k} \in \mathbf{A}^{type}$). In our simulated environment, $\Delta a^v = 0.02$ and $\Delta a^\theta = 0.2$. As alluded to in Fig. 3a, it is relatively easy to increase the number of agents in the system—we can simply add new introspective agents in parallel.



(a) SLiCC: The prosocial and introspective agents use one neural network each to approximate their Q-function. Each network has a single hidden dense layer of size $\ell_{den}^P = 1024$ with a $tanh$ activation function. The networks have different output dimensionality: $\ell^P \times \ell^P$ for agent P, and ℓ^I for agent I. In our experiments, $\ell^P = \ell^I = 9$. The actions are selected as per Eqs. (9) to (11).



(b) Centralized Q-learning: The centralized learning network has a single hidden dense layer of size $\ell_{den}^{central} = 1024$ with a $tanh$ activation function. The output dimensionality is $\ell^c \times \ell^c$ is the number of output size, with $\ell^c = 9$. The action pair (a^P, a^I) is that which maximizes the Q-value.

Fig. 3: The neural network structures for our experiments.

Our experiments include two scenarios: $\mathbf{A}^{type}$ + $\mathbf{RP}_\alpha$ and $\mathbf{A}^{type}$ + $\mathbf{RP}_\beta$. Correspondingly, we use centralized Q-learning with reward $r^g = r_{int} + r_{goal}^1 + r_{goal}^2 + r_{ap}^1 + r_{ap}^2$ as the baseline. The neural network structure is shown in Fig. 3b. There is also a single hidden dense layer with $\ell_{den}^{central}$ neurons. The output is the approximated Q-table which is a matrix with size $\ell^c \times \ell^c$, where $\ell^c = 9$.

C. Training Performance

Fig. 4 shows the average reward per episode attained with SLiCC and centralized Q-learning. Firstly, SLiCC shows good convergence behavior in our experiments. This validates our prosocial–introspective cooperation framework, which allows SLiCC to accomplish the common objective with stabilized learned policies. Despite the system being constrained by partial observations, SLiCC still demonstrates good performance. Secondly, after around 1,500 episodes, SLiCC has achieved better average reward and maintains this trend until the end of the training process. We can see that SLiCC takes less time than centralized Q-learning in reaching the same average reward level. One of the critical

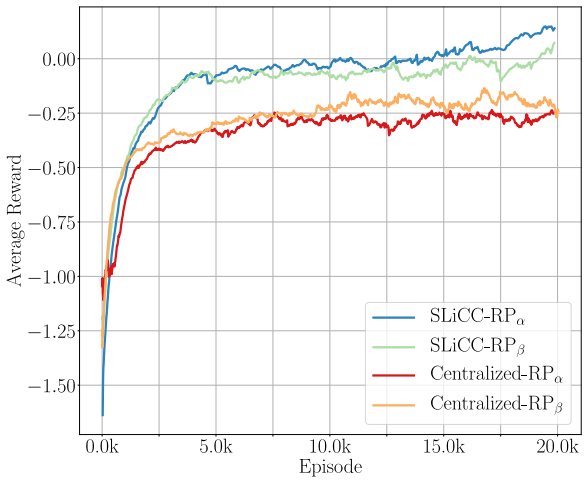


Fig. 4: Average reward per episode for centralized Q-learning and SLiCC with $\mathbf{A}^{type}$. Rewards for SLiCC are expressed as the sum of the rewards of both Agents P and I.

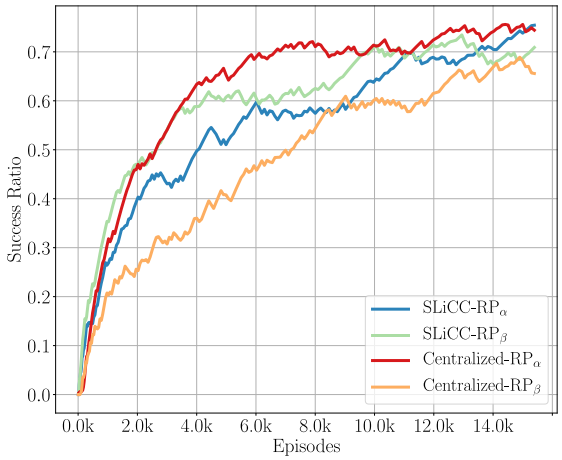


Fig. 5: Success ratio using SLiCC and centralized Q-learning.

reasons that explains why centralized Q-learning has lower average reward is the mutual interference between the three reward constituents: r_{int}, r_{goal}, and r_{ap}.

Fig. 5 shows the success ratio across episodes with SLiCC and centralized Q-learning, for both $\mathbf{RP}_\alpha$ and $\mathbf{RP}_\beta$. An episode is considered to be successful if three conditions are jointly satisfied:

1) the inter-agent distance is close to the desired distance σ at all times,
2) agent velocities are close to the target velocity $(v, \theta)^{target}$ at the end of an episode; and
3) the episode length exceeds a minimum duration.

As can be seen, SLiCC achieves a good success ratio which is equivalent to that of centralized Q-learning. While a centralized approach might appear to have an advantage in this experimental scenario because it can command the same action for each robot, this strategy does not work in practice due to noise in transition dynamics.

(a) TurtleBot3 Burger mobile robot (CC BY 4.0).

(b) Yujin Robot Kobuki mobile base.

Fig. 6: We learn a cooperative control policy in the Gazebo simulation environment with a pair of TurtleBot3 Burger mobile robots (Fig. 6a). To validate the real-world applicability of the learned policy, we run it on a pair of LoCoBot mobile robots, built on a Yujin Robot Kobuki mobile base (Fig. 6b). This introduces different physical dynamics, which confirms the generalizability of learned SLiCC policies. The real robots are able to successfully carry out the object transportation task (Fig. 7).

D. Real Robot Validation

To validate the real-world applicability of SLiCC, we used the policies learned in simulation to control a pair of real LoCoBot mobile robots. Notably, the LoCoBot is built on a Yujin Robot Kobuki mobile base (Fig. 6b), which has significantly different physical dynamics than the TurtleBot3 (Fig. 6a). After scaling the state and action spaces to approximate the spaces used in simulation, the learned policy was able to successfully perform the cooperative transport task without additional training (Fig. 7). This demonstrates the adaptability of SLiCC to different robotic platforms as long as appropriate considerations are made to account for differences in robot dynamics. These considerations include using generalized dynamics (Eqs. (1) to (4)) and compensating for any magnitude differences in the state and action spaces.

VII. Discussion and Conclusion

In this work, we introduce SLiCC: Stackelberg Learning in Cooperative Control, a novel technique to solve the multi-robot cooperation problem in partially observable scenarios. Our method links state perception with the agents' decision-making strategies through a Stackelberg game–based architecture, which we integrate with partially observable stochastic games (POSG). Payoff matrices are approximated via deep reinforcement learning, and the proposed framework is evaluated using both simulated and real robots.

In Section V, we detailed a number of advantages that SLiCC has over other learning paradigms. In future work, we intend to empirically demonstrate these advantages; some of the potential of the method is still untapped, and we will investigate other scenarios in which we expect SLiCC to perform better (e.g., limited communication bandwidth). In a separate direction, we also plan to explore alternative training frameworks (e.g., sequential training of the introspective and prosocial agents) and extend SLiCC to continuous action spaces. Finally, there are several improvements to SLiCC that

Fig. 7: The pair of real LoCoBot mobile robots carry out the object transportation task using the policies learned on TurtleBot3 Burger mobile robots in simulation. The photo on the far left shows the initial state of the the LoCoBot mobile robots. From left to right, the two robots cooperatively transport the object.

hold promise for applications in more complex environments (e.g., navigating around obstacles); these will also be pursued in future work.

ACKNOWLEDGMENT

We thank the anonymous reviewers and our colleagues for their insightful comments and suggestions which helped improve this manuscript.

REFERENCES

[1] M. Schneier and R. Bostelman, "Literature review of mobile robots for manufacturing," National Institute of Standards and Technology, Tech. Rep. NISTIR 8022, 2015.

[2] R. Bogue, "Growth in e-commerce boosts innovation in the warehouse robot market," *Industrial Robot*, vol. 43, no. 6, pp. 583–587, 2016.

[3] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human–robot collaboration," *Autonomous Robots*, vol. 42, no. 5, pp. 957–975, 2018.

[4] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Transactions on Cybernetics*, 2020.

[5] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[6] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 750–797, 2019.

[7] G. A. S. Pereira, A. K. Das, V. Kumar, and M. F. M. Campos, "Decentralized motion planning for multiple robots subject to sensing and communication constraints," in *2003 International Workshop on Multi-Robot Systems*, 2003, pp. 267–278.

[8] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[9] T. Miki, P. Khrapchenkov, and K. Hori, "UAV/UGV autonomous cooperation: UAV assists UGV to climb a cliff by attaching a tether," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8041–8047.

[10] J. Ichnowski, J. Prins, and R. Alterovitz, "The economic case for cloud-based computation for robot motion planning," in *18th International Symposium of Robotics Research (ISRR)*, 2020, pp. 59–65.

[11] O. Khatib, K. Yokoi, K.-S. Chang, D. Ruspini, R. Holmberg, and A. Casal, "Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation," in *1996 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1996, pp. 546–553.

[12] P. Ögren, M. Egerstedt, and X. Hu, "A control Lyapunov function approach to multiagent coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 847–851, 2002.

[13] J. T. Feddema, C. Lewis, and D. A. Schoenwald, "Decentralized control of cooperative robotic vehicles: Theory and application," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 852–864, 2002.

[14] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine, "Collective robot reinforcement learning with distributed asynchronous guided policy search," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 79–86.

[15] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 31–36.

[16] R. Fitch and Z. Butler, "Million module march: Scalable locomotion for large self-reconfiguring robots," *The International Journal of Robotics Research*, vol. 27, no. 3–4, pp. 331–343, 2008.

[17] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Tenth International Conference on Machine Learning (ICML)*, 1993, pp. 330–337.

[18] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Eleventh International Conference on Machine Learning (ICML)*, vol. 157, 1994, pp. 157–163.

[19] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *Journal of Machine Learning Research*, vol. 4, 2003.

[20] G. Ding, J. J. Koh, K. Merckaert, B. Vanderborght, M. M. Nicotra, C. Heckman, A. Roncone, and L. Chen, "Distributed reinforcement learning for cooperative multi-robot object manipulation," in *19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2020, pp. 1831–1833.

[21] S. Nikolaidis, S. Nath, A. D. Procaccia, and S. Srinivasa, "Game-theoretic modeling of human adaptation in human-robot collaboration," in *2017 ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2017, pp. 323–331.

[22] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Demand response management in smart grid networks: A two-stage game-theoretic learning-based approach," *Mobile Networks and Applications*, 2018.

[23] G. Ding, S. Aghli, C. Heckman, and L. Chen, "Game-theoretic cooperative lane changing using data-driven models," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3640–3647.

[24] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun, "Approximate solutions for partially observable stochastic games with common payoffs," in *Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2004.

[25] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella, "Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings," in *Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 2003, pp. 705–711.

[26] Y. Wen, Y. Yang, R. Luo, J. Wang, and W. Pan, "Probabilistic recursive reasoning for multi-agent reinforcement learning," in *7th International Conference on Learning Representations*, 2019.

[27] V. Könönen, "Asymmetric multiagent reinforcement learning," *Web Intelligence and Agent Systems: An International Journal*, vol. 2, no. 2, pp. 105–121, 2004.

[28] J. Laumônier and B. Chaib-draa, "Multiagent Q-learning: Preliminary study on dominance between the Nash and Stackelberg equilibriums," in *2005 AAAI Workshop on Multiagent Learning*, 2005, pp. 41–45.

[29] Z. Shi, R. Yu, X. Wang, R. Wang, Y. Zhang, H. Lai, and B. An, "Learning expensive coordination: An event-based deep RL approach," in *8th International Conference on Learning Representations*, 2020.

[30] H. Wang, T. Huang, X. Liao, H. Abu-Rub, and G. Chen, "Reinforcement learning in energy trading game among smart microgrids," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 8, 2016.

[31] G. Alcantara-Jiménez and J. B. Clempner, "Repeated Stackelberg security games: Learning with incomplete state information," *Reliability Engineering & System Safety*, vol. 195, 2020.

[32] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.