Near-Optimal Adversarial Policy Switching for Decentralized Asynchronous Multi-Agent Systems

Trong Nghia Hoang[1,†], Yuchen Xiao[2,†], Kavinayan Sivakumar[3], Christopher Amato[2], and Jonathan P. How[1]

Abstract— **A key challenge in multi-robot and multi-agent systems is generating solutions that are robust to other self-interested or even adversarial parties who actively try to prevent the agents from achieving their goals. The practicality of existing works addressing this challenge is limited to only small-scale synchronous decision-making scenarios or a single agent planning its best response against a single adversary with fixed, procedurally characterized strategies. In contrast this paper considers a more realistic class of problems where a team of asynchronous agents with limited observation and communication capabilities need to compete against multiple strategic adversaries with changing strategies. This problem necessitates agents that can coordinate to detect changes in adversary strategies and plan the best response accordingly. Our approach first optimizes a set of *stratagems* that represent these best responses. These optimized stratagems are then integrated into a unified policy that can detect and respond when the adversaries change their strategies. The near-optimality of the proposed framework is established theoretically as well as demonstrated empirically in simulation and hardware.**

I. INTRODUCTION

Multi-robot systems is a widely studied field, but the research is typically focused on a single team of cooperative, or self-interested, robots [1]–[3]. In contrast, many real-world domains consist of a team of robots that must complete tasks while competing against other adversarial robots. For instance, consider a team of UAVs tasked with surveying a scene or locating a secret base as well as an opposing team of UAVs tasked with preventing the secret base from being found. These adversarial scenarios require reasoning about not only completing the tasks designated to the team, but also considering what the adversarial robots may do to prevent their completion. In this paper, we study the general multi-robot decision-making problem with uncertainty in outcomes, sensors and communication, while incorporating multiple adversarial robots into this problem. Communication uncertainty and limitations further necessitates the design of decentralized agents that can coordinate with their teammates while anticipating changes in the adversary strategies using only their partial views of the world. This is for the first time all these forms of uncertainty as well as adversarial behavior have been considered in the same decision-theoretic planning framework[1]. Furthermore, the size of the spaces for possible set of actions and coordination strategies for both the teammates and adversaries scale exponentially in the number of agents [6] and are typically much too large for an agent to reason about directly. Hence, a successful agent usually requires some form of high-level abstraction to reduce its effective planning space [7]–[9].

One approach is therefore to create a set of basic stratagems, which are best-responses to particular forms of adversarial behavior. The reasoning problem is then reduced to choosing among these basic stratagems in a given situation, thus significantly improving the scalability of planning. This approach can be achieved by anticipating in advance a small set of high-level tactics from which the adversaries can choose in any situations, that capture the diversity of their intentions. The task of the high-level planner then is to choose a response to the adversaries' current tactics and follow it until it is determined that they have changed their tactics and a new response is needed. This fits particularly well in asynchronous robotic planning scenarios: since each stratagem has different execution time, agent decision making is no longer synchronized as assumed in existing non-cooperative multi-agent frameworks [6], [10], [11].

The main contribution of this paper therefore focuses on the design of such a high-level planner, which can be decoupled into two separate tasks. The first task involves generating a set of basic *stratagems* for a team of decentralized agents, each of which is optimized to work best against a particular tactic of the adversaries. This is formulated as a set of Macro-Action Decentralized Partially Observable Markov Decision Processes (MacDec-POMDPs) [9], [12] that each characterize a cooperative scenario where a team of decentralized agents collaborate to maximize the team's expected performance while operating in a stationary environment simulated by a single tactic of the adversaries (Section III). The stratagems can therefore be acquired by solving for a set of probabilistic policy controllers that maximize the expected total reward generated by the corresponding MacDec-POMDPs. Then, the second task is to integrate these specialized policy controllers into a unified policy controller that works best on average against the adversaries' switching tactics. This again can be achieved by optimizing the unified controller with respect to a series of MacDec-POMDPs (Section IV) so that it can detect situation changes and switch opportunistically between these stratagems to

[1] Trong Nghia Hoang and Jonathan How are with the Laboratory for Information and Decision Systems (LIDS), MIT, Cambridge, USA {nghiaht,jhow}@mit.edu

[2] Yuchen Xiao and Christopher Amato are with the College of Computer and Information Science (CCIS), Northeastern University, Boston, USA {xiao.yuch@husky.neu,c.amato@northeastern}.edu

[3] Kavinayan Sivakumar is with the Department of Electrical Engineering, Princeton University, Princeton, USA ks16@princeton.edu

[†] Trong Nghia Hoang and Yuchen Xiao contribute equally to this work.

[1] Previous works [4], [5] in the literature addressing this challenge have mostly focused on reactive frameworks or do not consider multiple adversarial robots into their frameworks.

respond effectively to the adversaries' new tactical choice. Interestingly, it can be shown that under a certain mild assumption, the result of this stratagem integration/fusion scheme appears to be near optimal with high probability as shown in Section V. Finally, to empirically demonstrate the effectiveness of the proposed framework, experiments conducted for a robotic scenarios are presented in Section VI, which show consistent results with our theoretical analysis.

II. BACKGROUND AND NOTATIONS

This section provides a short overview of MacDec-POMDPs [9], [12] for decentralized multi-agent decision-making under uncertainty. Formally, a MacDec-POMDP is defined as a Dec-POMDP [13], [14] tuple $(\mathbb{I},\mathbb{S},\{\mathbb{A}_i\}_{i=1}^{n},\{\mathbb{O}_i\}_{i=1}^{n}\mathbb{T},\mathbb{Z},\mathbb{R},\gamma,b)$ augmented with a finite set of macro-actions, $\mathbb{M}_i$, for each agent, i, with $\mathbb{M} \triangleq \mathbb{M}_1 \cup \ldots \cup \mathbb{M}_n$ denote the set of joint macro-actions. Each macro-action is defined as a tuple $m_i = (\beta_{m_i}, I_{m_i}, \rho_{m_i})$ where $\beta_{m_i} : \mathbb{S} \to \{0,1\}$ and $I_{m_i} \in \mathbb{S}_i$ are sets of rules that decide, respectively, the termination and eligibility to initiate of the corresponding macro-action m_i, while $\rho_{m_i} : \Theta_i \to \mathbb{A}_i$ denotes a low-level policy that maps agent i's local histories $\theta_i \in \Theta_i$ to primitive actions $a_i \in \mathbb{A}_i$. Each agent will follow a chosen macro-action m_i until its termination condition β_{m_i} is met. Its stream of observations collected during the execution of m_i is jointly defined as a macro-observation η_i. As such, each individual high-level policy $\pi_i : \zeta_i \to m_i$ of agent i can then be characterized as a mapping from its history of macro-actions and -observations $\zeta_i \triangleq \{(m_{t-1}^i, \eta_t^i)\}_{t\geq 1}$ to the next macro-action. Planning in Dec-POMDP therefore involves maximizing the following total expected reward with respect to the joint high-level policy $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$:

$$\pi^* \quad = \quad \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{+\infty} \gamma^t \mathbb{R}(s_t, a_t) | b, \mathbb{M}, \pi\right] \qquad (1)$$

Unlike Dec-POMDP's, the MacDec-POMDP formalism is naturally suitable for asynchronous multi-robot planning scenarios since it is not necessary for the macro-actions $m_i = (\beta_{m_i}, I_{m_i}, \rho_{m_i})$ to share the same execution time. In fact, from the perspective of an individual agent, the outcome of its selected macro-action (e.g., when it terminates) is non-deterministic as its termination rule may depend on the global state of the environment as well as the movements of the other parties, which are not observable to the agent. This makes optimizing π via (1) using traditional model-based dynamic programming techniques [9], [15]–[20] possible only if the probability distribution over the stochastic outcome of β_{m_i}, e.g., $p(\beta_{m_i}(s) = 0 \mid s)$, is explicitly characterized. This is not trivial and does not scale well in complex decision problems with long planning horizon, vast state and action spaces. Alternatively, to sidestep this difficulty, it is also possible to parameterize and optimize π directly via interaction with a black-box simulator[2] that implicitly encodes the probabilistic models of transition $\mathbb{T}$, observation $\mathbb{Z}$, reward $\mathbb{R}$ and termination rule β_{m_i} [8]. This interestingly allows us to avoid modeling these probabilistic models directly and improve the scalability of solving MacDec-POMDPs. The specifics of this model-free approach are detailed in Section III which serves as the building block of our adversarial multi-agent planning paradigm in Section IV.

III. GENERATING BASIC STRATAGEMS

This section assumes we have access to a set of black-box simulators preset by the domain expert to simulate accurately the adversaries' basic tactics, upon which more advanced strategies might be built. For example, in popular real-time strategy (RTS) games, a player can often anticipate in advance a small set of effective basic tactics from which the other competitors might choose in any situations. The decision making process of a player therefore comprises two parts. The first part focuses on formulating fundamental stratagems to counter the anticipated tactics of the adversaries and is addressed in the remaining of this section. The second part then is to integrate the resulting stratagems into a unified strategy that can detect changes in the adversaries' tactical choice and switch opportunistically between them in response to those changes (see Section IV).

In particular, formulating a stratagem to counter a specific tactic of the adversaries can be posed as solving a MacDec-POMDP which characterizes a cooperative scenario where a team of decentralized agents collaborate to maximize their total expected reward while operating in an artificial environment driven by the corresponding tactic simulator. The stratagem can then be optimized via simulation as detailed next. Formally, we represent a stratagem of a team of agents as a set of decentralized finite-state-automata (FSA) policy controllers, $\mathscr{C}^s = \{\mathscr{C}_k^s\}_{k=1}^n$, each of which characterizes a single agent's corresponding part of the stratagem, $\mathscr{C}_k^s$. Each individual controller $\mathscr{C}_k^s$ has p nodes $\{q_k^i\}_{i=1}^p$ and there are two probabilistic functions associated with each node q_k^i: (a) an output function $\lambda(m_k^j | q_k^i) \triangleq \lambda_{ij}^k$ which decides the probability λ_{ij}^k that macro-action $m_k^j \in \mathbb{M}_k$ is selected by agent k; and (b) a transition function $\delta(q_k^t | q_k^i, m_k^j) \triangleq \delta_{ij}^k(t)$ which determines the probability to transit from q_k^i to q_k^t following the execution of the selected macro-action m_k^j. The weights $\mathbf{w} \triangleq \{\{\lambda_{ij}^k\}, \{\delta_{ij}^k(t)\}\}$ can then be optimized via simulation using the graph-based direct cross-entropy (G-DICE) optimization method described in [8] (see Figure 1).

In essence, G-DICE iteratively samples $\mathbf{w}$ from a distribution $q(\mathbf{w};\theta)$ parameterized by θ and simulates the induced policy (with respect to $\mathbf{w}$) with the opponent's tactic $\mathscr{E}^s \triangleq \{\mathscr{E}_k^s\}_{k=1}^n$ using its black-box simulator to acquire a performance estimate $\mathbb{L}(\mathscr{C}^s(\mathbf{w}), \mathscr{E}^s)$. At each iteration, a subset of samples with top performance estimates is used to update θ via maximum likelihood estimation (MLE). This process has been demonstrated empirically in [21] to converge towards a uniform distribution over optimal values of $\mathbf{w}$. In practice, this optimization paradigm is very well-fitted to multi-robot planning scenarios since it allows us to bypass the explicit

[2] In many real-world scenarios, it is often easier to hand-code a simulator that captures the interaction rules between agents than learning probability models of their outcomes.

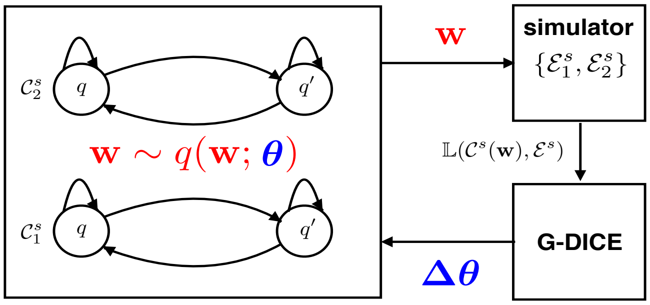

$\mathcal{C}_2^s$
q
q'
$\mathbf{w} \sim q(\mathbf{w};\boldsymbol{\theta})$
$\mathcal{C}_1^s$
q
q'
$\mathbf{w}$
simulator
$\{\mathcal{E}_1^s, \mathcal{E}_2^s\}$
$\mathbb{L}(\mathcal{C}^s(\mathbf{w}), \mathcal{E}^s)$
G-DICE
$\boldsymbol{\Delta\theta}$

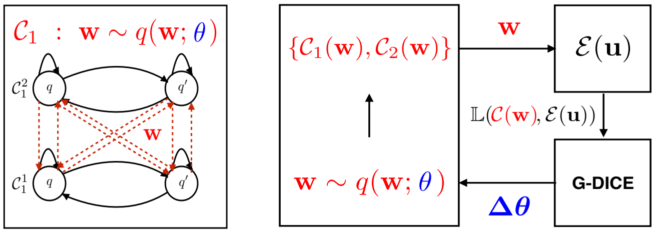
Fig. 1: A team of two collaborative agents is represented by two decentralized controllers $\mathscr{C}^s \triangleq \{\mathscr{C}_1^s, \mathscr{C}_2^s\}$ characterizing their stratagem against a basic tactic $\mathscr{E}^s \triangleq \{\mathscr{E}_1^s, \mathscr{E}_2^s\}$ of the adversaries. The sampling distribution of $\mathscr{C}^s$'s parametric weights $\mathbf{w}$ is then optimized using performance feedback $\mathbb{L}(\mathscr{C}^s(\mathbf{w}), \mathscr{E}^s)$ from interacting with $\mathscr{E}^s$'s black-box simulator.

probabilistic modeling of opponent's tactic which is usually fraught with the curses of dimensionality and histories, especially in complex problem domains with large number of agents, vast action and observation spaces [8]. This method will also serve as the building block for our stratagem fusion scheme detailed in Section IV below.

IV. STRATEGEM FUSION

This section introduces the stratagem fusion scheme that integrates all basic stratagems (see Section III) into a set of unified policies for a team of agents to collaborate effectively against the adversaries' high-level switching policies that switch opportunistically among a set of basic tactics. The task of stratagem fusion is then to formulate a high-level policy that can automatically detect situation changes and choose which response to follow at any point of decision to adapt effectively to new situations (e.g., the adversaries decide to switch to a different tactic) and consequently, maximize its expected performance. To achieve this, we model the team's high-level policy as a set of unified controllers, each of which characterizes a single agent's high-level individual policy that results from connecting its low-level controllers via inter-controller transitions (see Figure 2). This essentially allows the agents to change their strategic choices during real-time execution by transiting between different nodes of different controllers. The weights associated with these transitions therefore regulate the switching decision of the high-level controller and need to be optimized. If we know exactly how the adversaries change their tactics (i.e., their black-box simulators) in response to our strategic choices, these weights can be optimized using the same approach described in Section III (see Figure 2b).

In practice, however, the adversaries' switching mechanism is often unknown or highly non-trivial to characterize, especially in decentralized settings where their strategic choices are largely influenced by their limited observation and communication capacities, which are also unknown. Existing works [6], [10], [22], [23] that attempt to reason explicitly about the adversaries' strategic rationalities are therefore impractical and less robust in situations where irrational choices arise due to limited cognitive abilities and lack of communication. This motivates us to consider a more reasonable approach to formulate a robust policy that works well on average when tested against all possible high-level strategies of the adversaries. To achieve this, the adversaries' switching policies are similarly modeled as high-level controllers that connect low-level controllers representing their basic tactics using inter-controller transitions as illustrated in Figure 2a. The weights of these inter-controller transitions (that regulate switching decisions) are then treated as random variables distributed by a known distribution. Thus, instead of optimizing our agents' switching weights with respect to a single realization of the adversaries' inter-controller transitions, we optimize them with respect to the distribution of these switching weights to embrace their uncertainty.


$\mathcal{C}_1 \ : \ \mathbf{w} \sim q(\mathbf{w};\theta)$
$\mathcal{C}_1^2$
q
q'
$\mathbf{w}$
$\mathcal{C}_1^1$
q
q'
$\{\mathcal{C}_1(\mathbf{w}), \mathcal{C}_2(\mathbf{w})\}$
$\mathbf{w} \sim q(\mathbf{w};\theta)$
$\mathbf{w}$
$\mathcal{E}(\mathbf{u})$
$\mathbb{L}(\mathcal{C}(\mathbf{w}), \mathcal{E}(\mathbf{u}))$
G-DICE
$\boldsymbol{\Delta\theta}$

(a)

(b)

Fig. 2: (a) An agent's unified controller $\mathscr{C}_1(\mathbf{w})$ that connects its low-level controllers (i.e., stratagems) $\{\mathscr{C}_1^1, \mathscr{C}_1^2\}$ via inter-controller transitions (i.e., denote by the red, dash lines) whose weights $\mathbf{w}$ are to be optimized; and (b) a team of two agents optimizes their high-level joint controller $\{\mathscr{C}_1(\mathbf{w}), \mathscr{C}_2(\mathbf{w})\}$ via interaction with the black-box simulator of the adversaries' switching strategy $\mathscr{E}(\mathbf{u})$ given the switching weights $\mathbf{u}$.

Formally, let $\mathscr{C} = \{\mathscr{C}_k\}_{k=1}^n$ and $\mathscr{E} = \{\mathscr{E}_k\}_{k=1}^n$ denote the sets of high-level controllers for the teams of collaborative agents and adversaries, respectively, where $\mathscr{C}_k = \{\mathscr{C}_k^s\}_s$ ($\mathscr{E}_k = \{\mathscr{E}_k^s\}_s$) denotes a single agent's (adversary's) individual switching policy. Let $\mathbf{w} = \{\mathbf{w}_k\}_{k=1}^n$ and $\pi(\mathbf{u})$ denote the weights associated with inter-controller transitions of $\mathscr{C} = \{\mathscr{C}_k\}_{k=1}^n$ and the distribution over random weights $\mathbf{u} = \{\mathbf{u}_k\}_{k=1}^n$ that regulates the switching decision of $\mathscr{E} = \{\mathscr{E}_k\}_{k=1}^n$, respectively. Our approach proposes to optimize $\mathbf{w}$ such that the expected performance of the induced high-level controller $\mathscr{C}(\mathbf{w})$ when tested against a random adversary $\mathscr{E}(\mathbf{u})$ distributed by $\pi(\mathbf{u})$ is maximized:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left(\mathbb{L}(\mathbf{w}) \triangleq \mathbb{E}_{\mathbf{u}\sim\pi(\mathbf{u})}\Big[\mathbb{L}\left(\mathscr{C}(\mathbf{w}), \mathscr{E}(\mathbf{u})\right)\Big]\right) , \quad (2)$$

where $\mathbb{L}(\mathscr{C}(\mathbf{w}), \mathscr{E}(\mathbf{u}))$ denotes the simulated performance of $\mathscr{C}(\mathbf{w})$ against $\mathscr{E}(\mathbf{u})$. However, since we can only access the value of $\mathbb{L}(\mathscr{C}(\mathbf{w}), \mathscr{E}(\mathbf{u}))$ via simulation, solving (2) requires simulating $\mathscr{C}(\mathbf{w})$ against infinitely many candidates of $\mathscr{E}(\mathbf{u})$ and is therefore intractable. To sidestep this intractability, we instead exploit the following surrogate objective function,

$$\widehat{\mathbf{w}} = \arg\max_{\mathbf{w}} \left(\widehat{\mathbb{L}}(\mathbf{w}) \triangleq \frac{1}{m}\sum_{i=1}^{m} \mathbb{L}\left(\mathscr{C}(\mathbf{w}), \mathscr{E}(\mathbf{u}^{(i)})\right)\right) , \quad (3)$$

where $\{\mathbf{u}^{(i)}\}_{i=1}^m$ are i.i.d samples drawn from $\pi(\mathbf{u})$. Intuitively, these are the potential candidates for the adversaries' switching weights that can be identified in advance using the domain expert's knowledge. We can now solve (3) using G-DICE [8] (see Section III) with a meta black-box that aggregates the feedback of each black-box $\mathscr{E}(\mathbf{u}^{(i)})$.

V. THEORETICAL ANALYSIS

This section derives performance guarantees for the above stratagem fusion scheme (Section IV) which depend on the solution quality of the graph-based direct cross-entropy (G-DICE) optimization method described in [8]. To enable the analysis, we put forward the following assumption:

Assumption 1. Let $\widehat{\mathbb{L}}(\mathbf{w})$ denote an arbitrary black-box function being optimized via simulation with G-DICE using (3). Let $\mathbb{U} \triangleq \{\mathbf{w} \mid \widehat{\mathbb{L}}(\mathbf{w}) = \max_{\mathbf{w}'} \widehat{\mathbb{L}}(\mathbf{w}')\}$ denotes the set of optimal solutions to (3). Then, let $p(\mathbf{w})$ and $q(\mathbf{w};\theta)$ denote the uniform distribution over $\mathbb{U}$ and the sampling distribution of G-DICE parameterized by θ (see Section III). For any $\delta \in (0,1)$, there exists a non-decreasing sequence $\{\varepsilon_{n,\delta}\}_{n=1}^{\infty}$ for which:

$$\Pr\left(\mathbb{D}_{\mathrm{KL}}\Big(q\left(\mathbf{w};\theta^*\right) \| p(\mathbf{w})\Big) \ \leq\ \varepsilon_{n,\delta}\right) \quad \geq \quad 1-\delta \ ,$$

where n and θ^* denote the size of $\mathbf{w}$ and the optimal parameterization of $q(\mathbf{w};\theta)$ found by G-DICE, respectively.

This is a reasonable assumption to make since it has been previously demonstrated that the underlying cross-entropy optimization process of G-DICE empirically causes $q(\mathbf{w};\theta)$ to converge towards the uniform distribution $p(\mathbf{w})$ over optimal values of $\mathbf{w}$ [8], [21]. Then, let $\mathbb{L}(q) \triangleq \mathbb{E}_{\mathbf{w}}[\mathbb{L}(\mathbf{w})]$ (with $\mathbb{L}(\mathbf{w})$ defined in (2)) denote the expected performance of $\mathscr{C}(\mathbf{w})$ when $\mathbf{w}$ is drawn randomly from $q(\mathbf{w};\theta^*)$, we are interested in the gap between $\mathbb{L}(q)$ and $\mathbb{L}(\mathbf{w}^*)$ (see Eq (2)), the latter of which is the best performance that can be achieved. Thus, this gap essentially characterizes the near-optimality of $q(\mathbf{w};\theta^*)$, which are bounded below. To do this, we first establish the following results in Lemmas 1 and 2 that bound the difference between the generalized performance of q (i.e., $\mathbb{L}(q)$) and its empirical version (i.e., the average performance $\widehat{\mathbb{L}}(q)$ when tested against a finite set of adversary candidates). Lemma 3 is then established to bridge the gap between the $\widehat{\mathbb{L}}(q)$ and $\mathbb{L}(\mathbf{w}^*)$. The main result that bounds the performance gap between $\mathbb{L}(q)$ and $\mathbb{L}(\mathbf{w}^*)$ is then derived in Theorem 1 as a direct consequence of the previous Lemmas.

Lemma 1. For any sampling distribution $q(\mathbf{w};\theta)$, let $\widehat{\mathbb{L}}(q) \triangleq \mathbb{E}_{\mathbf{w}}\left[\widehat{\mathbb{L}}(\mathbf{w})\right]$, with $\widehat{\mathbb{L}}(\mathbf{w})$ defined in (3), denotes the empirical performance of $\mathscr{C}(\mathbf{w})$ where $\mathbf{w}$ is randomly drawn from $q(\mathbf{w};\theta)$. Then, it follows that with probability at least $1-\delta$ over the choice of candidates $\{\mathbf{u}^{(i)}\}_{i=1}^{m}$ for the adversaries' switching weights,

$$\mathbb{L}(q) \quad \leq \quad \widehat{\mathbb{L}}(q) + \left(\frac{\mathbb{D}_{\mathrm{KL}}\left(q(\mathbf{w};\theta) \| p(\mathbf{w})\right) + \log\frac{4m}{\delta}}{2m-1}\right)^{\frac{1}{2}} \tag{4}$$

holds universally for all possible $q(\mathbf{w};\theta)$ where $p(\mathbf{w})$ denotes the uniform distribution over the set $\mathbb{U}$ of optimal choice of $\mathbf{w}$ for (3), i.e., $\mathbb{U} \triangleq \{\mathbf{w} \mid \widehat{\mathbb{L}}(\mathbf{w}) = \max_{\mathbf{w}'} \widehat{\mathbb{L}}(\mathbf{w}')\}$.

Exploiting the result of Lemma 1, we can further derive a tighter and domain specific bound on the difference between the generalized and empirical performance of our stratagem fusion scheme (see Section IV) that incorporates the empirical optimality of G-DICE (see Assumption 1):

Lemma 2. Let $q \ \triangleq \ q(\mathbf{w};\theta^*)$ denotes the optimal sampling distribution found by G-DICE [8]. Let r denotes the number of stratagems of each agent (Section III) and let k denotes the number of nodes in each agent's individual specialized controller $\mathscr{C}_k^s$. It then follows that with probability at least $1-\delta$,

$$\mathbb{L}(q) \quad \leq \quad \widehat{\mathbb{L}}(q) \ + \ \left(\frac{\varepsilon_{h,\frac{\delta}{2}} + \log\frac{8m}{\delta}}{2m-1}\right)^{\frac{1}{2}} \ , \tag{5}$$

where $\mathbb{L}(q)$ and $\widehat{\mathbb{L}}(q)$ are defined in Lemma 1, $h = O\left(nr(r-1)k^2\right)$ and $\delta \in (0,1)$.

Lemmas 1 and 2 thus bound the performance gap between $\mathbb{L}(q)$ and $\widehat{\mathbb{L}}(q)$. To relate $\mathbb{L}(q)$ to $\mathbb{L}(\mathbf{w}^*)$, we need to bound the gap between $\widehat{\mathbb{L}}(q)$ and $\mathbb{L}(\mathbf{w}^*)$, which is detailed below.

Lemma 3. Let $q \ \triangleq \ q(\mathbf{w};\theta^*)$ denotes the optimal sampling distribution found by G-DICE [8] and $\delta \in (0,1)$, then with probability at least $1-\delta$,

$$\widehat{\mathbb{L}}(q) \quad \leq \quad \mathbb{L}(\mathbf{w}^*) \ + \ \left(\frac{\log\frac{1}{\delta}}{2m}\right)^{\frac{1}{2}} \ , \tag{6}$$

where $\widehat{\mathbb{L}}(q)$ is defined in Lemma 1.

Using these results, the key result can be stated and proven:

Theorem 1. Let $q \ \triangleq \ q(\mathbf{w};\theta^*)$ denotes the optimal sampling distribution found by G-DICE [8] and $\mathbf{w}^*$ denotes the optimal solution to (2). $\mathbb{L}(\mathbf{w}^*)$ thus represents the best possible performance and with probability at least $1-\delta$,

$$\mathbb{L}(q) \quad \leq \quad \mathbb{L}(\mathbf{w}^*) \ + \ 2\left(\frac{\varepsilon_{h,\frac{\delta}{4}} + \log\frac{16m}{\delta}}{2m-1}\right)^{\frac{1}{2}} \ , \tag{7}$$

where $\mathbb{L}(q)$ is defined in Lemma 1, $h = O\left(nr(r-1)k^2\right)$. Due to the limited space, all proofs of the above results are deferred to the appendix of the extended version of this paper at `https://arxiv.org/pdf/1710.06525.pdf`.

VI. EXPERIMENTS

This section presents an adversarial, multi-robot Capture-The-Flag (CTF) domain adapted from its original domain in [4] to demonstrate the effectiveness of our stratagem fusion framework in Section IV. The specific domain setup for our CTF variant is detailed in Section VI-A below.

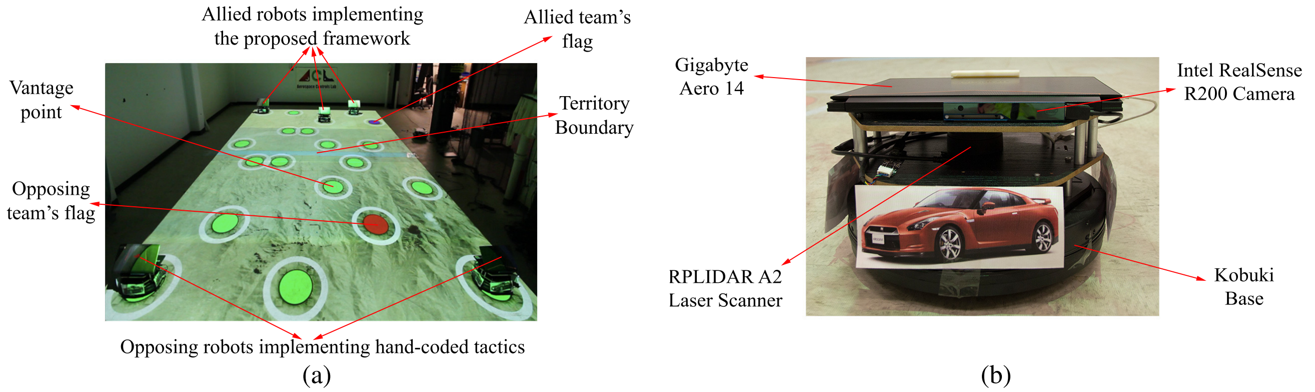


Fig. 3: Figures of (a) Capture-The-Flag (CTF) domain setup; and (b) hardware configuration of the experimented robots: each robot is built from the Kobuki base of the TurtleBot 2 with on-board processing unit (Gigabyte Aero 14 laptop with Intel Core i7-7700HQ quad-core CPU and NVIDIA GTX 1060 GPU with 6GB RAM) as well as sensory devices including (1) Intel RealSense Camera (R200) Developer Kit (130mm x 20mm x 7mm) with Depth/IR: Up to 640×480 resolution at 60 FPS & RGB: 1080p at 30 FPS; and (2) Omnidirectional RPLIDAR A2 with 4000 samples/s (10Hz) and 8/16m range.

A. *Capture-The-Flag Domain*

The domain settings for Capture-The-Flag are shown in Fig. 3a, which depicts a competitive scenario between two teams of decentralized, collaborative robots. Each team has $2-3$ robots and the two teams divide the environment into two parts separated by a horizontal boundary (the cyan line in Fig. 3a), each of which belongs to one team (e.g., red & blue). There are 10 vantage points within each team's territory. One of which contains the flag of the team (e.g., the red and blue circles in Fig .3a denote the locations of the flags for the red and blue team, respectively). Each team, however, only knows the location of its own flag, thus making observations necessary to correctly detect the enemy's flag. The rule of the game is for each team to defend its own flag while seeking to capture the flag of the opposing team without getting caught. The game ends when one team successfully captures the flag of the opposing team. To achieve this, each team of agents need to coordinate their movements between vantage points to reach the opposing team's flag and at the same time, avoid being seen by opposing agents. If an agent engages an opposing agent on foreign territory, its team will be charged with a penalty. The particular macro-actions and -observations available for each robot are detailed below, which feature a wide range of interesting observations and patterns of collaborative attack and defend for the opposing robots:

Macro-Actions. There are 4 classes of macro-actions available to each robot at any decision time: (a) **Move**(p) which invokes a collision avoidance navigation procedure that directs the robot to vantage point p from its current location; (b) **Sentry**(p_1, p_2, p_3) which directs the robot to vantage point p_1 and then lets it stay in a closed-loop moving from p_1 to p_2 to p_3 and back to p_1. There are 5 predefined instances for each team; (c) **Pincer**(p_1, p_2, p_3, p_4) which directs the robot to vantage point p_i (with i being its role index in the team) and then p_4. This creates an effective pincer attack when 2 or 3 robots choose the same **Pincer** instance. There are 3 different **Pincer** macros predefined for each team; and (d) **Tag** which allows a robot to catch an opposing robot on its own territory provided the opposing robot is within a predefined tagging range.

Macro-Observations. There are in total 128 macro-observations for each robot, which are generated by first collecting raw observations the environment using the robot's on-board visual recognition/detection modules and then summarizing the raw information into a 6-dimensional observation vector. Each observation is represented as a 6-dimensional binary vector whose components correspond to yes/no $(1/0)$ answers to the following questions: (a) Is the robot residing in its own or the opposition territory? (b) does the enemy flag appear in sight? (c) is there an opposition robot in close proximity? (d) is there an opposition robot further away? (e) is there an allied robot in the vicinity? and (f) is there an observed pincer signal emitted from allied robots? The answers to these questions can be generated from the raw visual processing unit on-board each robot.

Rewards. Finally, in order to encourage each team to discover and capture the opposition's flag as soon as possible while avoid getting tagged, a reward mechanism is implemented which issues (a) a negative reward of -1 to each robot at each time step; (b) a positive reward of 10 to a team if one of its member successfully tags an enemy; (c) a negative reward of -10 for the entire team if one of the team member gets caught; and (d) a large award of 500 is issued to a team when it successfully captures the opposition's flag. Conversely, this implies a large penalty of -500 issued to the other team who loses the flag.

Black-box Simulators. In addition to the domain specification above, the allied robots also have access to a set of black-box simulators of the opposition's fundamental tactics upon which more advanced strategies might be built. In our experiments, these are constructed as tuples of individual hand-coded tactics (see Table I below) that include: (a) **DL** and **DR** which script the robot to play defensively on left and right flank of its territory using **Sentry** and **Move** macro-actions, respectively; (b) **DC** which scripts the robot to play defensively on the middle-front of the allied territory; (c)

TABLE I: The opposition's team tactics $\mathscr{E}^s$ represented as combinations of individual's tactics $\{\mathscr{E}^s_k\}_k$ of 3 robots **R1**, **R2** and **R3**.

	R1 $\left(\{\mathscr{E}^s_1\}^4_{s=1}\right)$	**R2** $\left(\{\mathscr{E}^s_2\}^4_{s=1}\right)$	**R3** $\left(\{\mathscr{E}^s_3\}^4_{s=1}\right)$
$\mathscr{E}^1 \triangleq (\mathscr{E}^1_1, \mathscr{E}^1_2, \mathscr{E}^1_3)$	$\mathscr{E}^1_1 \triangleq$ **DL**	$\mathscr{E}^1_2 \triangleq$ **DC**	$\mathscr{E}^1_3 \triangleq$ **DR**
$\mathscr{E}^2 \triangleq (\mathscr{E}^2_1, \mathscr{E}^2_2, \mathscr{E}^2_3)$	$\mathscr{E}^2_1 \triangleq$ **DL**	$\mathscr{E}^2_2 \triangleq$ **AS**	$\mathscr{E}^2_3 \triangleq$ **DR**
$\mathscr{E}^3 \triangleq (\mathscr{E}^3_1, \mathscr{E}^3_2, \mathscr{E}^3_3)$	$\mathscr{E}^3_1 \triangleq$ **AA**	$\mathscr{E}^3_2 \triangleq$ **DC**	$\mathscr{E}^3_3 \triangleq$ **AS**
$\mathscr{E}^4 \triangleq (\mathscr{E}^4_1, \mathscr{E}^4_2, \mathscr{E}^4_3)$	$\mathscr{E}^4_1 \triangleq$ **AS**	$\mathscr{E}^4_2 \triangleq$ **AA**	$\mathscr{E}^4_3 \triangleq$ **AS**

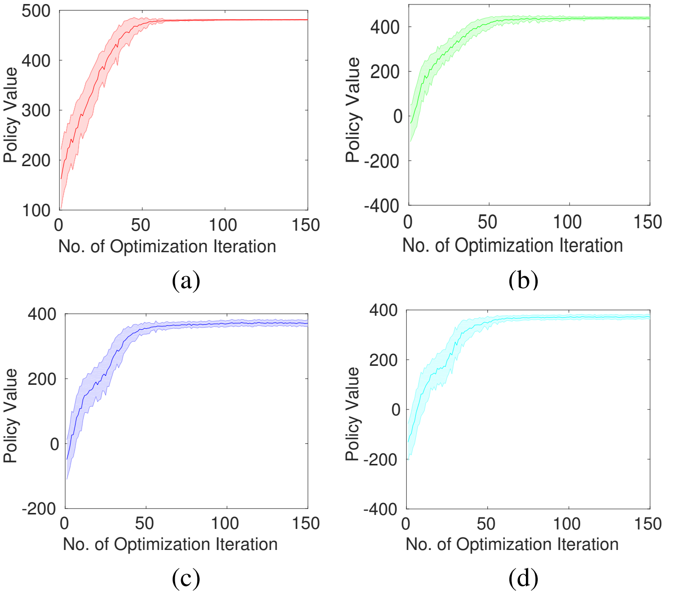
AS which leads the robot to a vantage point inside the opposition's territory to get an observation. Depending on the collected observation, the robot either moves to another vantage point or launch a pincer attack to a vantage point estimated to contain the opposition's flag; and (d) **AA** which is similar to **AS** except that it enables the robot to retreat to a safe place within the allied territory to gather extra observations if it observes that there is an opposing robot in close proximity. The team of allied robots however do not have access to these details and can only interact with them via a black-box interface that gives feedback on how well their strategies fare against the opposition's.

B. Experiment: Generating Basic Stratagems

To learn the fundamental stratagems to counter the opposition's basic tactics as described in Section VI-A, we construct separate MacDec-POMDPs (see Section II) that encapsulates the opposition's corresponding tactic simulator $\mathscr{E}^s \triangleq \{\mathscr{E}^s_k\}_k$. The corresponding stratagem can then be formulated and computed as decentralized FSA controllers $\mathscr{C}^s \triangleq \{\mathscr{C}^s_k\}_k$ (Section III) that optimizes these MacDec-POMDPs. This is achieved via a recently developed graph-based direct cross-entropy (G-DICE) stochastic optimization method of [8]. Fig 4 shows that the empirical performance of each stratagem $\mathscr{C}^s$ when tested against the corresponding opposition's tactic $\mathscr{E}^s$ increases and converges rapidly to the optimal performance when we increase the number of optimization iterations. Table II then reports the averaged performance (with standard errors) of each stratagem when tested against all other opposition's tactics over 1000 independent simulations. The results interestingly show that the quality of each stratagem $\mathscr{C}^s$ decreases significantly when tested against other opposition's tactics $\{\mathscr{E}^{s'}\}_{s' \neq s}$ that it was not optimized to interact with (see Table II's first 4 rows and columns). This implies a performance risk when applying a single stratagem against non-stationary opponent with switching tactics: The applied stratagem might no longer be optimal when the opponent switches to a new tactic. This necessitates design of agents which can detect and respond aptly when the opponents change their tactics which constitutes the main contribution of our work (Section IV). Its effectiveness is demonstrated next in Section VI-C.

C. Experiment: Stratagem Fusion

This section empirically demonstrates the effectiveness of our stratagem fusion framework (Section IV) against more sophisticated and non-stationary/strategic opponents. In particular, we first evaluate the performance of the optimized stratagems in the previous experiments (Section VI-B) against a team of opponents with switching tactic: each opponent independently switches its tactic based on a set of probability weights $\mathbf{u}$ (as previously described in Section IV). The results (averaged over 1000 independent runs) are reported in the last columns of Table II, which show significant decreases in the performance of each stratagem when tested against an opponent that keeps switching between tactics. This corroborates our observations earlier that a single stratagem is generally ineffective against opponents with unexpected behaviors. This can be remedied using our stratagem fusion scheme (see Fig. 2) to integrate all single stratagems into a unified (switching) policy which can perform effectively against the switching tactic of the opponents (assuming the switching weights $\mathbf{u}$ are known). The reported results in the last row of Table II in fact show that among all policies, the optimized switching policy performs best against the tactic-switching opponents and near-optimal against each stationary opponent: Its performance is, in most cases, only second (and very close) to the corresponding stratagem specifically designed to counter the opponent's tactic.



(a) (b)

(c) (d)

Fig. 4: Graphs of each stratagem's increasing performance quality in the no. of optimization iterations when optimized against the opposition's corresponding tactic of (a) (**DL**, **DC**, **DR**), (b) (**DL**, **AS**, **DR**), (c) (**AA**, **DC**, **AS**) and (d) (**AS**, **AA**, **AS**) (see Table I). The shaded area represents the confidence interval of the average performance.

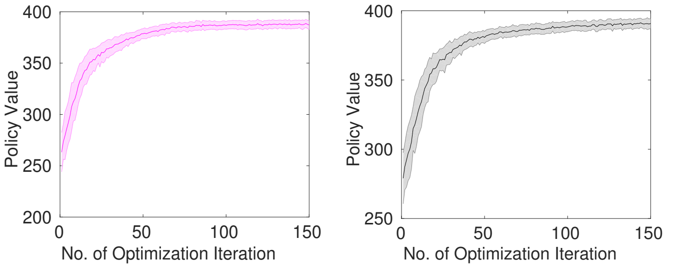
In practice, however, since the switching weights of the opponents are usually not known a priori, a similar problem arises when the actual weights $\mathbf{u}$ used by the opposition's switching tactic are different from those used to optimize the switching policy of the allied robots. To resolve this, our stratagem fusion scheme further treated the switching weights $\mathbf{u}$ of the opposition as random variables whose samples are either given in advance or can be drawn directly from a blackbox distribution $\pi(\mathbf{u})$. A *good-for-all* switching policy $\mathscr{C}(\widehat{\mathbf{w}})$ can thus be computed using our sampling method in Section IV (specifically, see Eq. (3))

TABLE II: Average performance (with standard errors) of the robots' basic stratagems $\mathscr{C}^1,\mathscr{C}^2,\mathscr{C}^3,\mathscr{C}^4$ and switching policy $\mathscr{C}(\mathbf{w})$ when tested against the opposition's basic tactics $\mathscr{E}^1,\mathscr{E}^2,\mathscr{E}^3,\mathscr{E}^4$ (see Table I) and switching tactic $\mathscr{E}(\mathbf{u})$ with switching weights $\mathbf{u}$. The switching policy $\mathscr{C}(\mathbf{w})$ is learned assuming access to a blackbox simulator of $\mathscr{E}(\mathbf{u})$ (see Section III).

	$\mathscr{E}^1 = (\mathbf{DL},\mathbf{DC},\mathbf{DR})$	$\mathscr{E}^2 = (\mathbf{DL},\mathbf{AS},\mathbf{DR})$	$\mathscr{E}^3 = (\mathbf{AA},\mathbf{DC},\mathbf{AS})$	$\mathscr{E}^4 = (\mathbf{AS},\mathbf{AA},\mathbf{AS})$	$\mathscr{E}(\mathbf{u})$
$\mathscr{C}^1$	**481.235 ± 0.119**	405.737 ± 0.933	184.081 ± 1.453	126.665 ± 1.277	329.598 ± 1.137
$\mathscr{C}^2$	450.004 ± 1.217	**439.339 ± 0.699**	191.609 ± 1.279	97.129 ± 1.392	295.037 ± 1.302
$\mathscr{C}^3$	296.436 ± 2.616	139.034 ± 1.573	**374.477 ± 1.049**	190.408 ± 1.285	263.993 ± 1.343
$\mathscr{C}^4$	323.481 ± 2.463	218.717 ± 1.432	352.924 ± 0.924	**375.485 ± 0.893**	309.696 ± 1.229
$\mathscr{C}(\mathbf{w})$	469.007 ± 0.774	399.731 ± 0.949	332.819 ± 1.006	301.353 ± 1.095	**386.831 ± 0.992**

TABLE III: Average performance (with standard errors) of the allied robots' *good-for-one* $\mathscr{C}(\mathbf{w})$ (optimized against a particular $\mathscr{E}(\mathbf{u})$) and *good-for-all* $\mathscr{C}(\widehat{\mathbf{w}})$ (optimized against the entire distribution of $\mathbf{u}$ – see Section IV) switching policies when tested against unseen switching policies $\mathscr{E}(\mathbf{u}^{(1)}),\mathscr{E}(\mathbf{u}^{(2)}),\ldots,\mathscr{E}(\mathbf{u}^{(6)})$ of the opposition.

	$\mathscr{E}(\mathbf{u}^{(1)})$	$\mathscr{E}(\mathbf{u}^{(2)})$	$\mathscr{E}(\mathbf{u}^{(3)})$	$\mathscr{E}(\mathbf{u}^{(4)})$	$\mathscr{E}(\mathbf{u}^{(5)})$	$\mathscr{E}(\mathbf{u}^{(6)})$
$\mathscr{C}(\mathbf{w})$	383.836 ± 1.006	385.482 ± 0.892	388.875 ± 0.875	388.361 ± 0.876	389.545 ± 0.871	389.824 ± 0.869
$\mathscr{C}(\widehat{\mathbf{w}})$	**385.811 ± 1.013**	**390.403 ± 0.882**	**393.296 ± 0.865**	**391.793 ± 0.870**	**394.021 ± 0.861**	**391.698 ± 0.873**



(a) (b)

Fig. 5: Graphs of the switching policy's increasing performance in the no. of optimization iterations when optimized against a switching tactic of the opposition with (a) fixed switching weights; and (b) random switching weights. The shaded area represents the confidence interval surrounding the average performance.

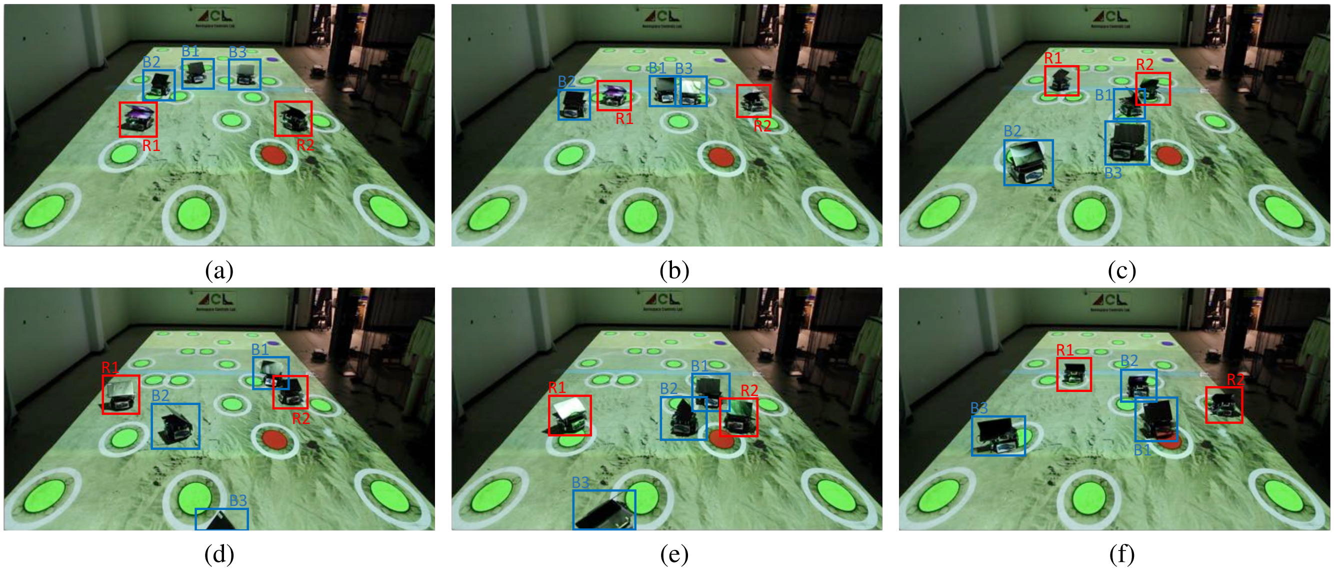
which is guaranteed, with high probability, to produce near-optimal performance against unseen switching weights of the opposition. This is empirically demonstrated in Table III which shows the superior performance of the *good-for-all* policy $\mathscr{C}(\widehat{\mathbf{w}})$ to that of the *good-for-one* policy $\mathscr{C}(\mathbf{w})$ when tested against opponents with unseen tactic-switching weights $\mathscr{E}(\mathbf{u}^{(1)}),\mathscr{E}(\mathbf{u}^{(2)}),\ldots,\mathscr{E}(\mathbf{u}^{(6)})$. Also, similar to the case of basic stratagem in Section III, the quality of those switching policies increases and converges rapidly to the optimal value when we increase the number of optimization iterations (Fig. 5) in our stratagem fusion framework, which demonstrates the its stability.

VII. Hardware Experiments

In addition to the simulated experiments, we also conduct real-time experiments with real robots to showcase the robustness of our proposed framework in practical RTS scenarios. The specifics of our robot configuration and domain setup are shown in Fig. 3. Each robot is built with the Kobuki base of TurtleBot 2 and configured with on-board processing unit (Gigabyte Aero 14 laptop with Intel Core i7-7700HQ quad-core CPU and NVIDIA GTX 1060 GPU with 6GB RAM) as well as sensory devices including (1) Intel RealSense Camera (R200) Developer Kit (130mm x 20mm x 7mm) with Depth/IR: Up to 640×480 resolution at 60 FPS & RGB: 1080p at 30 FPS; and (2) Omnidirectional RPLIDAR A2 (4000 samples/sec (10Hz) and 8/16m range). The information provided by the LIDAR sensor is directed to each robot's on-board collision-avoidance navigation procedure [24] to helps it localize and move around without colliding with other robots and obstacles in the environment. The visual feed from RealSense camera is passed through the Single Shot MultiBox Detector [25] implemented on each allied robot's processing unit to detect its surrounding objects (e.g., the opposing robots, other allied robots and flags). The processed information is then used to generate the high-level macro-observations (Section VI-A) for the robot's on-board policy controller. Fig. 6 shows a visual excerpt from our video demo featuring a CTF scenario of 3 allied robots which implement the optimized policy produced by our framework to compete against an opposing team of 2 adversary robots implementing the hand-coded tactics in Section VI-A. The excerpt shows interesting teamwork between all allied robots in capturing the opposing team's flag despite their partial, decentralized views of the world (see detailed narration in Fig. 6's caption), which further demonstrates the robustness of our proposed framework in practical robotic applications. Interested readers are referred to our video at `https://youtu.be/EGLprD6MMGE` for a complete visual demonstration.

VIII. Conclusion

This paper introduces a novel near-optimal adversarial policy switching algorithm for decentralized, non-cooperative multi-agent systems. Unlike the existing works in literature which are mostly limited to simple decision-making scenarios where a single agent plans its best response against an adversary whose strategy is specified a priori under reasonable assumptions, we investigate instead a class of multi-agent scenarios where multiple robots need to operate independently in collaboration with their teammates to act effectively against adversaries with changing strategies. To achieve this, we first optimize a set of basic stratagems that each is tuned to respond optimally to a pre-identified basic tactic of the adversaries. The stratagems are then integrated into a unified policy which performs near-optimally against



(a) (b) (c)

(d) (e) (f)

Fig. 6: Image excerpts from a video demo showing (1) a team of 3 allied (blue) robots (**B1**,**B2** and **B3**) that implement the optimized stratagem produced by our framework (Section III) to compete against (2) an opposing team of 2 opponent (red) robots (**R1** and **R2**) which implement the hand-coded tactics **DL** and **DR** (see Section VI-A), respectively: (a) **B1**,**B2** and **B3** decide to invade the opposition territory; (b) **B1** and **B3** decide to attack the center while **B2** decides to take the left flank of the opposition; (c) **B2** passes through **R1**'s defense while **B1** takes an interesting position to block **R2** so that **B3** can pass through its defense; (d) **B1** and **B2** detect the flag and mount a pincer attack; (e) **R2** arrives to defend the flag and **B2** retreats to avoid getting tagged; and (f) without noticing **B1** from behind, **R2** continues its **DR** patrol, thus losing the flag to **B1**.

any high-level strategies of the adversaries that switches between their basic tactics. The near-optimality of our proposed framework can be established in both theoretical and empirical settings with interesting and consistent results. We believe this is a significant step towards bridging the gap between theory and practice in multi-agent research.

Acknowledgements. This research is funded in part by the ONR under MURI program award #N000141110688 and BRC award #N000141712072 and Lincoln Lab.

References

[1] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

[2] Justin Werfel, Kirstin Petersen, and Radhika Nagpal. Designing collective behavior in a termite-inspired robot construction team. *Science*, 343(6172):754–758, 2014.

[3] Shayegan Omidshafiei, Ali-Akbar Agha-Mohammadi, Yu Fan Chen, Nazim Kemal Ure, Shih-Yuan Liu, Brett T. Lopez, Rajeev Surati, Jonathan P. How, and John Vian. Measurable augmented reality for prototyping cyberphysical systems: A robotics platform to aid the hardware prototyping and performance testing of algorithms. *IEEE Control Systems*, 36:65–87, 2016.

[4] Raffaello D'Andrea and Richard M. Murray. The roboflag competition. In *Proc. ACC*, 2003.

[5] Peter Stone. *Intelligent Autonomous Robotics: A Robot Soccer Case Study*. Morgan and Claypool Publishers, 2007.

[6] P. J. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multi-agent settings. *JAIR*, 24:49–79, 2005.

[7] S. Omidshafiei, A. akbar Agha-mohammadi, C. Amato, and J. P. How. Decentralized control of partially observable markov decision processes using belief space macro-actions. In *Proc. ICRA*, 2015.

[8] S. Omidshafiei, A. akbar Agha-mohammadi, C. Amato, Shih-Yuan Liu, J. P. How, and J. Vian. Graph-based cross entropy method for solving multi-robot decentralized POMDPs. In *Proc. ICRA*, 2016.

[9] Christopher Amato, George D. Konidaris, and Leslie P. Kaelbling. Planning with macro-actions in decentralized POMDPs. In *Proc. AAMAS*, 2014.

[10] T. N. Hoang and K. H. Low. Interactive POMDP Lite: Towards practical planning to predict and exploit intentions for interacting with self-interested agents. In *Proc. IJCAI*, 2013.

[11] T. N. Hoang and K. H. Low. A general framework for interacting Bayes-optimally with self-interested agents using arbitrary parametric model and model prior. In *Proc. IJCAI*, pages 1394–1400, 2013.

[12] Christopher Amato, George D. Konidaris, Ariel Anders, Gabriel Cruz, Jonathan P. How, and Leslie P. Kaelbling. Policy search for multi-robot coordination under uncertainty. *The International Journal of Robotics Research*, 2017.

[13] Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of markov decision processes. In *Proc. of the 16th Conference on Uncertainty in Artificial Intelligence*, June 2000.

[14] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

[15] D. Szer, F. Charpillet, and S. Zilberstein. Maa*: A heuristic search algorithm for solving decentralized POMDPs. In *Proc. of Uncertainty in Artificial Intelligence*, 2005.

[16] S. Seuken and S. Zilberstein. Improved memory-bounded dynamic programming for decentralized POMDPs. In *Proc. of Uncertainty in Artificial Intelligence*, July 2007.

[17] S. Seuken and S. Zilberstein. Memory-bounded dynamic programming for DEC-POMDPs. In *Proc. IJCAI*, pages 2009–2015, 2007.

[18] A. Boularias and B. Chaib-draa. Exact dynamic programming for decentralized POMDPs with lossless policy compression. In *Proc. of Int. Conference on Automated Planning and Scheduling*, 2008.

[19] M. T. J. Spaan, F. A. Oliehoek, and N. Vlassis. Multiagent planning under uncertainty with stochastic communication delays. In *Proc. ICAPS*, 2008.

[20] M. T. J. Spaan, F. A. Oliehoek, and C. Amato. Scaling up optimal heuristic search in DEC-POMDPs via incremental expansion. In *Proc. IJCAI*, pages 2027–2032, 2011.

[21] Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A Unified Approach to Monte Carlo Simulation, Randomized Optimization and Machine Learning*. 2004.

[22] P. Doshi and D. Perez. Generalized point based value iteration for interactive POMDPs. In *Proc. AAAI*, pages 63–68, 2008.

[23] P. Doshi, Xia Qu, Adam Goodie, and Diana Young. Modeling recursive reasoning by humans using empirically informed interactive POMDPs. In *Proc. AAMAS*, pages 1223–1230, 2010.

[24] Y. Chen, S. Liu, M. Liu, J. Miller, and J. How. Motion planning with diffusion maps. In *Proc. IROS*, 2016.

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. arXiv:1512.02325, 2016.

[26] David McAllester. PAC-Bayesian model averaging. In *COLT*, 1999.