2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Learning Coordinated Maneuver in Adversarial Environments

Zechen Hu, Manshi Limbu, Daigo Shishika, Xuesu Xiao, and Xuan Wang

***Abstract*— This paper aims to solve the coordination of a team of robots traversing a route in the presence of adversaries with random positions. Our goal is to minimize the overall cost of the team, which is determined by (i) the accumulated risk when robots stay in adversary-impacted zones and (ii) the mission completion time. During traversal, robots can reduce their speed and act as a 'guard' (the slower, the better), which will decrease the risks certain adversary incurs. This leads to a trade-off between the robots' guarding behaviors and their travel speeds. The formulated problem is highly non-convex and cannot be efficiently solved by existing algorithms. We employ reinforcement learning techniques by developing new encoding and policy-generating methods. Simulations demonstrate that our learning methods can efficiently produce team coordination behaviors. We discuss the reasoning behind these behaviors and explain why they reduce the overall team cost.**

I. Introduction

Coordination of multi-robot systems has been studied under various contexts [2], including cooperative path planning [3], resource sharing, and task allocation [4], and geometric formation maintenance [5]. Complementary to the challenges addressed in these works, this paper introduces a new problem centered around generating coordinated team behaviors to reduce risks caused by adversaries. Considering a graph-based representation, team coordination problems have been studied in [6]–[8]. In this work, as shown in Fig. 1, we consider a route-based version of the problem, fine-grinding the movements of robots in continuous space. The environment features adversaries whose positions are randomly initialized from a set. Robots accumulate 'risks' when traveling through adversary-controlled zones, and such risks can be reduced by robots if they slow down and 'guard' a certain adversary. We define the team cost as a combination of mission completion time and accumulated risk. Therefore, minimizing this cost requires robots' coordination, trading off between their speed and the adoption of guarding behaviors. This adds a novel dimension of complexity and strategic decision-making, which is unattended in conventional multi-robot coordination tasks.

The described scenario has direct applications in many domains. For instance, when multiple rescue robots need to pass a fire-engulfed corridor [9], some robots might deploy countermeasures as *guard* to quell flames, ensuring safer passage for their peers; On battlefields, when multiple vehicles need to traverse enemy-controlled territories [10], suppressive fire from allies can guard and mitigate threats posed by the enemy. The formulated multi-robot coordination problem is challenging due to the combinatorial nature

Work supported by Army Research Office (W911NF-22-2-0242) and NSF (2332210). George Mason University. {zhu3, klimbu2, dshishik, xiao, xwang64}@gmu.edu. Extra experiments of this work appear in [1].

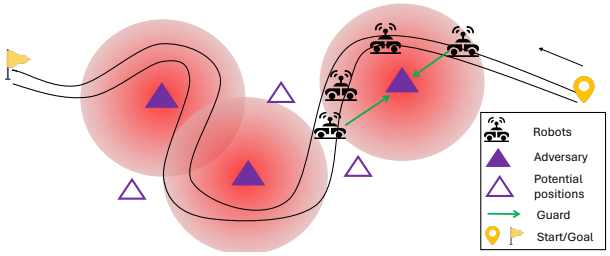


Fig. 1: A team of robots traversing an environment with risk. Adversary positions are randomly initialized from a set.

of robots' states, their hybrid actions (speed and guard), and multiple constraints embedded through robot-adversary interactions.

To solve this problem, one feasible approach from the optimization literature is Mixed Integer Programming (MIP). MIP has been applied to various multi-robot coordination problems [8], [11], [12]. Solving MIP problems is NP-complete, making traditional MIP solvers sensitive to the number of variables [13] and primarily applicable to small-scale problems. However, for our problem of interest, the coordination of robots occurs at every time step, depending on their coupled actions and states. Such constraints and task specifications can be encoded through logic formulation and piece-wise nonlinear functions [14]. Yet, considering the entire trajectories of all robots as variables can quickly render MINLP (Mixed Integral NonLinear Programming) with coupled constraints intractable. To address this issue, the advances in MIP such as interior point [15], branch and bound [6] methods, and heuristic approaches [16], can drastically improve computation scalability by leveraging convexity properties or the decomposability of the problem. While these properties may not inherently exist for general applications, one can introduce special cost function design and relaxation [6] to promote convexity; or by simplifying explicit collaboration between robots [16] to enhance problem decomposability. However, since MIP for multi-robot coordination generally provides an *open-loop* solution for the full trajectories of all robots, it faces scalability challenges as the number of robots increases, and requires repeated re-planning if the adversary position or any environmental features change.

While optimization techniques suffer from computational complexity, Reinforcement Learning (RL) methods enable robots to employ trial and error to efficiently find empirical solutions for complex problems. Moreover, the learned policy is closed-loop and can adapt in real-time to new adversary positions. For centralized problem solving, Deep Q-Networks (DQN) [17], a value-based RL method suitable for discrete actions, has been applied to learn team formations

in battle games [18]. For complex tasks with continuous actions such as traffic optimization, Advantage Actor-Critic (A2C) methods [19] can achieve faster convergence and better exploration by utilizing a policy-based model as an actor. Building on A2C, there are generalizations such as Proximal Policy Optimization (PPO) [20] and Deep Deterministic Policy Gradient (DDPG) [21] with improved stability or data efficiency. For systems with hybrid action space, there exist hybrid-PPO [22] methods that can output discrete actions simultaneously with continuous actions. In addition, considering the agent-based nature of our problem, the mentioned algorithms also have multi-agent variants such as Deep Coordination Graph [23], Multi-agent PPO [24], and Multi-agent DDPG [25].

In our setup, although we follow an idea similar to existing centralized hybrid-PPO methods, we face challenges in training efficiency. This has motivated us to apply special treatment to the state encoding of robots and adversaries, as well as reward reshaping, to address these issues.

The *main contributions* are as follows: (i) We rigorously formulate a new multi-robot coordination problem that incorporates guard behaviors among team members to mitigate risks from adversaries. (ii) We investigate the conversion of the problem into Markov Decision Processes (MDPs) with hybrid move and guard actions. (iii) We introduce a Hybrid Proximal Policy Optimization algorithm tailored to our problem, featuring special treatment of reward reshaping and a unique multi-weighted hot encoding for representing robots' states and adversary positions. (iv) We perform extensive simulated experiments to validate the effectiveness of the proposed method and compare it with MIP methods. We also elucidate the rationale behind observed behaviors and their effectiveness in reducing the collective team cost.

II. Problem Formulation

We will first formulate the *task* of the multi-robot team, then introduce the notions of *risk* and *guard*. Based on these, we quantify the *team cost* and describe our *problem of interest*. Throughout the following definitions, adversaries are considered heterogeneous while robots are homogeneous.

Robots: Consider a number of n robots traversing a route of length L. The position of the i-th robot along the route at time t is represented by $s_i^t \in [0, L]$. Let $v_i^t \in [0, v_{\max}]$ denote the speed of the i-th traveling robot at time t. Thus, the position update for each traveling robot is given as

$$s_i^{t+1} = s_i^t + v_i^t \Delta t. \tag{1}$$

where Δt is the time interval. Before robot i arrives at the destination, each time step will produce a time penalty, denoted by P_i^t. We define

$$P_i^t = \begin{cases} p & \text{if } s_i^t < L, \\ 0 & \text{if } s_i^t = L. \end{cases} \tag{2}$$

Adversaries and Risk: Let m denote the number of adversaries. The position of adversary j is represented as

$$z_j \in \mathcal{D}_j \subset (0, L), \tag{3}$$

which is randomly chosen from a set $\mathcal{D}_j$ for each trial of experiment. Around z_j is the impact zone of the adversary, denoted by $\mathcal{M}_j$. If a robot is in this region, i.e., $s_i^t \in \mathcal{M}_j$, a cost $r_{i,j}^t$ will be incurred,

$$r_{i,j}^t = \begin{cases} f_j(s_i^t, z_j) \geq 0 & \text{if } s_i^t \in \mathcal{M}_j, \\ 0 & \text{if } s_i^t \notin \mathcal{M}_j, \end{cases} \tag{4}$$

which depends on the relative positions of s_i^t and z_j^t.

Guard: During traversal, if $s_k^t \in \mathcal{M}_j$, robot $k \in \{1, \cdots, n\}$ can reduce its speed and counteract adversary j as a 'guard'. Specifically, let $g_k^t \in \{1, 2, \cdots, m\}$ denote the index of the adversary that robot k is guarding against at time t. Then, the risks that adversary j incurs to robots i, $\forall s_i^t \in \mathcal{M}_j$ are discounted to $\alpha_{k,j}^t r_{i,j}^t$, where

$$\alpha_{k,j}^t = \begin{cases} 1 - \beta \dfrac{|v_{\max} - v_k^t|}{v_{\max}} & \text{if } g_k^t = j, \\ 1 & \text{otherwise}, \end{cases} \tag{5}$$

is the discount factor with $\beta \in (0, 1)$. When $v_k^t = 0$, robot k achieves best guarding performance $\alpha_{k,j}^t = 1 - \beta$, while as $v_k^t \to v_{\max}$, the guarding effect vanishes. Furthermore, we assume the guarding effects stack with each other, thus, considering all robots in the system guarding an adversary j, the risk it incurs to robot i is discounted by all guards as $\prod_{k=1}^n \alpha_{k,j}^t r_{i,j}^t$.

Team Cost: Let T be the total time for all robots in the team to traverse the route. Based on the above definitions, the team cost considers the risks and time penalties accumulated by all robots,

$$\boldsymbol{J} = \sum_{i=1}^{n} (\boldsymbol{R}_i + \boldsymbol{P}_i), \tag{6}$$

where $\boldsymbol{R}_i = \sum_{t=1}^T R_i^t \Delta t$ and $\boldsymbol{P}_i = \sum_{t=1}^T P_i^t \Delta t$, with

$$R_i^t = \sum_{j=1}^{m} \prod_{k=1}^{n} \alpha_{k,j}^t r_{i,j}^t. \tag{7}$$

taking into account the guarding effect.

Problem of Interest: In each time step, robot i's action is composed of traveling speed v_i^t and guard target g_i^t. Assume the robots can observe adversaries' positions z_j but do not know $\mathcal{D}_j$. To strategically design all robots' behaviors, let $\boldsymbol{v}^t = \{v_1^t, \ldots, v_n^t\}$, and $\boldsymbol{g}^t = \{g_1^t, \ldots, g_n^t\}$. The problem is to minimize the team cost, i.e., $t \in \{1, \cdots, T\}$

$$\min_{\{\boldsymbol{v}^t, \boldsymbol{g}^t\}} \boldsymbol{J}. \tag{8}$$

Note that the moving and guarding behaviors of robots lead to team coordination, as one robot can decrease its speed to benefit all traveling robots within the influence region of the guarded adversary.

III. Method

In this section, we present methods for solving the formulated problem. We introduce proximal policy optimization (PPO) based RL algorithms with a special multi-weighted hot

state encoding mechanism and reward reshaping to improve training efficiency.

The Markov Decision Process (MDP) formulation of our problem is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, R)$, including state, action, state transition, discount factor, and reward. Here, we propose a centralized MDP to address the multi-robot coordination problem formulated in Sec. II. Specifically, let $\boldsymbol{s}^t = \{s_1^t, \ldots, s_n^t, z_1, \ldots, z_m\} \in \mathcal{S}$ be the state set at time t, where $s_i^t, z_j \in [0, L]$ are associated with robots positions and adversaries positions, respectively. Since adversaries may appear at random positions, their information needs to be encoded into the system state so that the learned policy can generate different strategies corresponding to the adversary positions. Following this, the state space is defined as:

$$\mathcal{S} \coloneqq [0, L]^n \times [0, L]^m.$$

For the actions of robots, we have $a_i^t = (v_i^t, g_i^t)$, which is a hybrid combination of continuous speed $v_i^t \in [-v_{\max}, v_{\max}]$ and discrete guard behavior $g_i^t \in \{1, 2, \ldots, m\}$. The action space for all robots is

$$\mathcal{A} \coloneqq \left([-v_{\max}, v_{\max}] \times \{1, 2, \ldots, m\}\right)^n. \tag{9}$$

Based on $\mathcal{S}$ and $\mathcal{A}$, the state transition is $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. The robot states follow motion dynamics (1), which is deterministic, and the adversary positions are static and do not depend on $\mathcal{A}$. The $R(\boldsymbol{s}^t, \boldsymbol{a}^t)$ is the immediate reward of action $\boldsymbol{a}^t \in \mathcal{A}$ with state $\boldsymbol{s}^t \in \mathcal{S}$, defined as the negative team cost

$$R(\boldsymbol{s}^t, \boldsymbol{a}^t) \coloneqq -\sum_{i=1}^{n} (R_i^t + P_i^t). \tag{10}$$

We complete our MDP formulation by choosing a discounter factor $\gamma = 0.995$. The goal of RL is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ to maximize the expected cumulative reward for the whole team over the task horizon T, i.e.,

$$\max_{\pi} \quad \mathop{\mathbb{E}}_{\boldsymbol{a}^t \sim \pi(\cdot|\boldsymbol{s}^t)} \left[\sum_{t=0}^{T} (\gamma)^t R^t \right]. \tag{11}$$

Multi-weighted hot state encoding. To employ RL methods to solve our MDP problem, we can directly feed a scalar representation of each robot's and adversary's state (position) to the model. However, we observe that the dimension of our state space is much smaller than that of the action space due to the joint speed and guard behaviors. This discrepancy hinders the neural network's reasoning capabilities, which cannot efficiently learn parameters [26]. Inspired by the one-hot encoding, we seek to expand the state space using a similar mechanism. However, one-hot encoding is typically suited for discrete variables, whereas our state space is continuous. To address this, for robot positions, we introduce a new *weighted hot encoding* mechanism, which represents a continuous variable as the weighted average of two neighboring one-hot vectors. Specifically, let $h(s) \in [0, 1]^{L+1}$ denote the weighted hot encoding for a state $s \in [0, L]$, and $s = s_{\text{int}} + s_{\text{dec}}$, which has both integer and decimal parts. Let

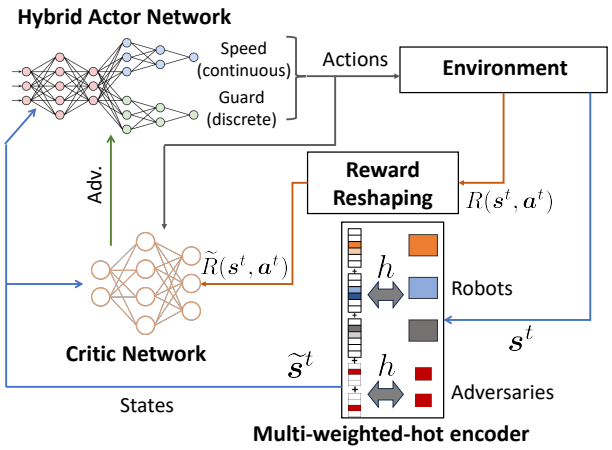


Fig. 2: RL Implementation: H-PPO with multi-weighted hot encoding and reward reshaping

$h(s)[k], k \in \mathbb{Z}$ denote the kth element of vector $h(s)$. Then $h(s)$ is a vector with two nonzero entries:

$$\begin{cases} h(s)[s_{\text{int}} + 1] = 1 - s_{\text{dec}} \\ h(s)[s_{\text{int}} + 2] = s_{\text{dec}} \end{cases} \tag{12}$$

As a simple example, if $L = 4$ and $s = 3.2$. Since $3.2 = 0.8 \times 3 + 0.2 \times 4$, one has, $h(s) = [0 \;\; 0 \;\; 0 \;\; 0.8 \;\; 0.2]^\top$. If $s = 0.7 = 0 \times 0.3 + 1 \times 0.7$, one has $h(s) = [0.3 \;\; 0.7 \;\; 0 \;\; 0 \;\; 0]$.

Since we consider a centralized MDP, we vector stack each robot's encoding and adversaries encoding through $h(\cdot)$ to obtain a Multi-weighted hot state encoding as follows:

$$\widetilde{\boldsymbol{s}}^t = \text{vec}\{h(s_1^t), \cdots, h(s_n^t), h(z_1), \cdots, h(z_m)\},$$

which has a dimension of $[0, 1]^{(n+1)(L+1)}$. Ths allows us to individually encode robots' positions and adversary positions in a way that is easily understood by the neural network.

It is worth **remarking** that one-hot encoding is typically used for categorical variables. In our problem, the positions of robots, whether inside or outside of adversary-impacted zones, correspond to completely different properties and different feasible guard actions. This distinction favors one-hot encoding, as all input entries are orthogonal to each other. This fact further justifies why multi-weighted hot state encoding can enhance our learning efficiency.

Reward reshaping. Since our task requires all robots to move to the terminal position, it is common to introduce a one-time constant reward $Q(\boldsymbol{s}^t) = q$, if $s_i^t = L, \forall i$; $Q(\boldsymbol{s}^t) = 0$, otherwise. However, this terminal reward is so sparse that it provides limited guidance to robots for state-space exploration and policy updates. Due to the greedy nature of the action section process, robots are reluctant to enter adversary zones. To address this, we further introduce a reshaping reward

$$F(\boldsymbol{s}^t, \boldsymbol{s}^{t+1}) = c \sum_{i=1}^{n} \left(\gamma s_i^{t+1} - s_i^t\right), \tag{13}$$

which incites robots to move forward and speeds up the learning process [27]. The final reshaped reward reads

$$\widetilde{R}(\boldsymbol{s}^t, \boldsymbol{a}^t) = R(\boldsymbol{s}^t, \boldsymbol{a}^t) + Q(\boldsymbol{s}^t) + F(\boldsymbol{s}^t, \boldsymbol{s}^{t+1}) \tag{14}$$

where $\boldsymbol{s}^{t+1}$ is determined by $\boldsymbol{s}^t, \boldsymbol{a}^t$ through $\mathcal{T}$. We **note** that the reshaped reward does not change the optimal solution compared to the original formulation. This is guaranteed by [28], as both $Q(\boldsymbol{s}^t)$ and $F(\boldsymbol{s}^t, \boldsymbol{s}^{t+1})$ can be rewritten into a potential-based function: $\gamma\Phi(\boldsymbol{s}^{t+1}) - \Phi(\boldsymbol{s}^t)$, where $\Phi(\cdot)$ is a real-valued function of state and γ is the discount factor.

RL Implementation. Combining the formulated MPD with multi-weighted hot state encoding and reward reshaping, we use two proximal policy optimization (PPO) based RL algorithms to solve the multi-robot path traveling problem. The key difference lies in the way we handle the hybrid action space. First, for simplicity, we consider pure discrete action space, assuming robots only take integral speeds. This has led to a standard PPO with discrete actions (D-PPO), and the proposed multi-weighted hot state encoding degrades to multi-one hot encoding. Second, we consider PPO with hybrid action space (H-PPO), and let the actor-network simultaneously output continuous speed actions and discrete guard actions. The policy losses (*log* probabilities) of the two actions are combined and used for training the network parameters. A conceptual diagram of the RL implementation is shown in Fig. 2, with a centralized structure to handle all robots' rewards and actions.

IV. Simulated Experiments

In this section, we present simulated experiment results to validate the RL implementation for team coordination behaviors in Sec. III. For complex cases, we discuss the reasoning behind these behaviors and why they reduce team costs. We also provide comparisons with baselines and numerical methods based on MINLP using Surrogate optimization [29] and BONMIN Solver [30].

A. Simulation environment

We consider three different environments shown in Fig. 3, including 1-3 adversaries with potential overlaps over their impact zones. The route in Fig. 1 is abstracted as a linear distance from the starting point to the target. In M1, the adversary's position z_j is randomly chosen from a discrete integer set $\mathcal{D}_j$, with width 10 centered at the middle of the environment; In M2, the two adversary's positions are chosen from two discrete sets $\mathcal{D}_j$, each has a width 5; the risk zones may overlap with each other. For both M1 and M2, when risk functions $f_j(s_i^t, z_j^t)$ are homogeneous and depends linearly on the distance between the robot and an adversary (visualized by the height of the shades in the figure). In M3, we consider stationary adversary positions, but the risk functions are heterogeneous and nonlinear, and with more complex overlaps.

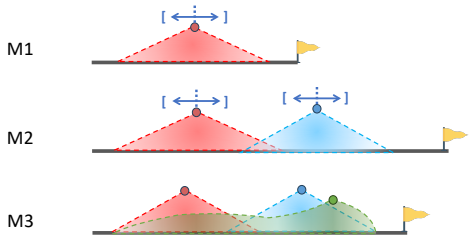
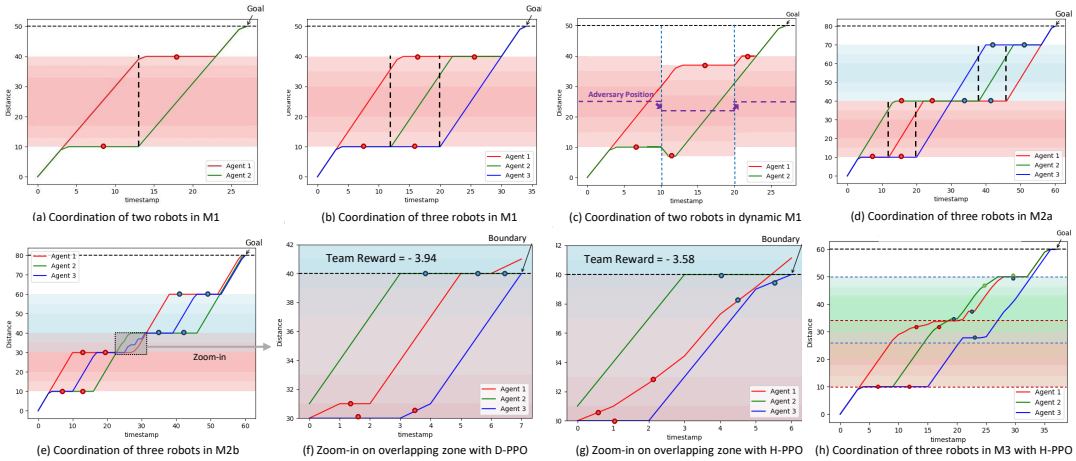


Fig. 3: Environments with different adversary configurations. Height represents the unit risk each adversary generates.

We choose the following environment parameters: time interval $\Delta t = 1$, max robot speed $v_{\max} = 3$, risk coefficient $\eta = 1$, time penalty $p = 1$, guard discount coefficient $\beta = 0.6$. All rewards, before being sent to D-PPO, H-PPO models, are re-scaled for normalization purposes. For each environment (M1, M2, M3), the learning model is individually trained. Within each environment, since adversary positions are part of the state space, a single model is trained with all possible adversary positions. During execution, this single model can adapt to adversary positions that are randomly generated for each test. This reflects the generalization capability of our model.

B. Team coordination and model generalization with homogeneous adversaries.

The results in Fig. 4a-c consider the M1 scenario with two and three robots, respectively. The D-PPO and H-PPO methods generate almost the same results except for slight differences in velocity when robots leave or approach the boundaries of adversary-impacted zones, which leads to minor changes to the final reward. For conciseness, we only visualize the results from D-PPO. In Fig. 4a, the behaviors of the two robots follow a 'bounding overwatch' pattern, i.e., one robot guards at the boundary of the adversary-impacted zone with minimal risk until the other robot moves across the zone at full speed. Then, the two robots switch roles to cooperatively accomplish the task with minimum cost. In Fig. 4b, as the number of robots increases, we observe (from the vertical dashes) a change in robots' coordination such that the guarding robots start moving 2 seconds before the traveling robots arrive at the other end. This adjustment is due to the increase in the number of robots; the time penalty encourages the robots to arrive at the destination more quickly. In Fig. 4c, we change the adversary's position in the middle of the execution. Although our algorithm is not designed to account for the dynamic behaviors of adversaries, the closed-loop nature of the policy and the fact that all possible adversary positions have been learned during the training process allow robots to quickly make adaptations. At the behavior level, robots maintain their position at the boundary of the zone when guarding. This demonstrates the model's generalization capability. If using optimization-based approaches, such as mixed-integer programming, then replanning is necessary. We employ D-PPO and H-PPO methods to solve optimal coordination under the M2 environment. Since the positions of the two adversaries are randomly initialized, it is possible that the two adversary-impacted zones may or may not overlap with each other, which will then impact the coordination patterns of the robots. Here, we choose two representative cases: without overlap and with overlap. In Fig.4d, without overlap, the three robots simply reproduce coordination behaviors in Fig. 4b over the two zones, respectively. In the case of Fig.4e, where two adversaries have an overlapped area (c.f. M2 in Fig. 3), the results of D-PPO and H-PPO show relatively consistent strategies for the $[10, 30]$ and $[40, 60]$ zones, as shown in



(a) Coordination of two robots in M1 (b) Coordination of three robots in M1 (c) Coordination of two robots in dynamic M1 (d) Coordination of three robots in M2a

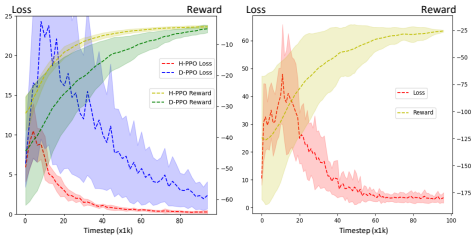
(e) Coordination of three robots in M2b (f) Zoom-in on overlapping zone with D-PPO (g) Zoom-in on overlapping zone with H-PPO (h) Coordination of three robots in M3 with H-PPO

Fig. 4: Using RL to solve team coordination problems in Fig. 3. The x-axis represents time and the y-axis represents the length the robot traveled in the environment. The slope represents the robot's speed. The color dots on the trajectories represent the adversary the robot is currently guarding against. In (c) the purple dash represents the current adversary position. The shades are the adversary-impacted zones with darker colors in the middle to represent higher risk, corresponding to Fig. 3.

Fig. 4e, but exhibit variant behaviors in the overlapping zone $[30, 40]$. To investigate this, we zoom into the overlapping zone and observe the difference between H-PPO and D-PPO results shown in Fig. 4f-g. Note that robots start at $\{30, 31, 30\}$ instead of $\{30, 30, 30\}$ because when entering the zone, robot 2 moves with $v_{\max} = 3$ from $s_2 = 28$ and directly arrives at $s = 31$. For a similar reason, the terminal states are $\{41, 40, 40\}$. In both Fig. 4f and g, at least one robot takes intermediate speeds that perform move and guard behaviors simultaneously. When robots are close to the boundaries of the overlapping zone, their risks are dominated by red and blue adversaries, respectively. As observed in plots, the stationary robot always guards the adversary which causes more risk, while the robot with intermediate speed will guard the other adversary. The asymmetry in robot's behavior may be due to time discretization, where the cost is computed at the end of each time step.

C. Behavior analysis of team coordination in complex environments with heterogeneous adversaries.

For the case of three adversaries with multiple overlaps and heterogeneous nonlinear risk functions (M3), we deploy three robots and obtain coordination results as shown in Fig. 4h. Here, the adversaries are stationary. We added colored dashes to better visualize the boundaries of red and blue adversaries. The environment's complexity makes it difficult to judge the optimality of the obtained coordination. In the following, we only discuss the reasoning behind the obtained result and explain why it reduces the overall team cost. First, according to Fig. 3, the M3 environment adds M2 with an extra green adversary. This green adversary generates an unsymmetric and nonlinear risk zone, which poses little risk early in the path but grows large as robots proceed. Consequently, in Fig. 4h, robots first follow a pattern very similar to that of Fig. 4e. However, midway through the path, as risks associated with the green adversary become large, the predecessor robot 1 does not fully stop to guard others. Instead, it takes an intermediate speed to guard the red adversary. This lasts until robot 1 meets robot 2 and both



(a) M2 environment with overlapping zone using H-PPO and D-PPO. (b) M3 environment using H-PPO.

Fig. 5: Convergence results for training loss and rewards.

leave the boundary of the red adversary. On the other hand, we observe robot 3 stops in the middle of the path to perform guard. This happens because, at the moment, the other two robots are suffering huge risks from both blue and green adversaries. By stopping and thereby increasing its own risk, robot 3 contributes to the cost-saving of the whole team. Finally, when robot 1 and 2 both arrives at the boundary, they help robot 3 by guarding red and blue adversaries, respectively. We note that the coordination presented in Fig. 4h may not be the global optimal, and its optimality gap is difficult to quantify. However, as discussed above, the observed robot coordination does exhibit rational behaviors, with the goal of reducing the overall team cost.

D. Reward Comparison with Baseline and MINLP Solvers

To visualize the training process, in Fig. 5, we use M2 and M3 environments as representative results to show training losses and rewards. We observe that H-PPO performs better than D-PPO and we note that D-PPO struggles to converge for the M3 environment.

For comparison purposes, we introduce a naive baseline strategy where all robots' actions are decoupled, and each robot individually uses greedy choices for move and guard. Additionally, we write the MINLP formulation of the problem and solve it using the BONMIN Solver [30] in Python and the Surrogate algorithm in MATLAB. When run indefinitely, BONMIN takes hours trying to close the optimality gap. Therefore, we stop BONMIN in 10 minutes

TABLE I: Comparison with Baseline and MINLP Solvers

Env	D-PPO	**H-PPO**	Baseline	BONMIN	Surrogate
M1	-5.80 ± 0.2	$\mathbf{-5.78 \pm 0.0}$	-9.94	$\mathbf{-5.78}$	-6.11
M2	-8.63 ± 0.7	-8.22 ± 0.4	-20.18	$\mathbf{-8.03}$	N/A
M3	-40.13 ± 8.5	$\mathbf{-26.87 \pm 2.0}$	-44.70	-30.32	N/A

if it does not converge and record the result. For all environments in Fig. 3, a fixed set of adversary positions is chosen, and three robots are deployed to test all approaches. Specifically for M2, there is an overlap between the two adversaries. As shown in Table I, the developed H-PPO generally outperforms or provides comparable performance to other methods. The Surrogate does not exhibit meaningful convergence on rewards for the two complex cases. As a final remark, while BONMIN is stopped after 10 minutes, H-PPO obviously requires significantly more training time. However, H-PPO can be trained offline and provides a general closed-loop policy for all possible adversary positions. In contrast, for BONMIN, each adversary position needs to be solved individually for an open-loop solution. Thus, for real-time applications, the proposed H-PPO is more desirable.

V. Conclusion

We have formulated a coordination problem considering a team of robots traversing a route with adversaries. The cost of the team was determined by the time penalty and the risk robots accumulated when crossing adversary-impacted zones, which can be reduced by robots' guard behavior when moving at a lower speed. We proposed and implemented an H-PPO method with reward reshaping and a new multi-weighted hot state encoding mechanism. Simulated experiments are performed under different environments and compared with alternative approaches to validate the effectiveness of the H-PPO method. Based on the simulation results, we discussed the reasoning behind these behaviors in terms of reducing the overall team cost. Future work will consider developing a decentralized learning paradigm to achieve scalable coordination.

References

[1] Z. Hu, M. Limbu, D. Shishika, X. Xiao, and X. Wang, "Learning coordinated maneuver in adversarial environments," *arXiv preprint arXiv:2407.09469*, 2024.

[2] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *Ieee Access*, vol. 6, pp. 28573–28593, 2018.

[3] Y. Zhou, X. Wang, and W. Jin, "D3g: Learning multi-robot coordination from demonstrations," *arXiv preprint arXiv:2207.08892*, 2022.

[4] M. Afrin, J. Jin, A. Rahman, A. Rahman, J. Wan, and E. Hossain, "Resource allocation and service provisioning in multi-agent cloud robotics: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 842–870, 2021.

[5] Y. Zhou, C. Nowzari, and X. Wang, "Distributed multi-robot flocking based on acoustic doppler effect," in *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 6406–6411, IEEE, 2023.

[6] C. A. Dimmig, K. C. Wolfe, and J. Moore, "Multi-robot planning on dynamic topological graphs using mixed-integer programming," *arXiv preprint arXiv:2303.11966*, 2023.

[7] M. Limbu, Z. Hu, S. Oughourli, X. Wang, X. Xiao, and D. Shishika, "Team coordination on graphs with state-dependent edge cost," in *20230 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023.

[8] C. A. Dimmig, K. C. Wolfe, M. Kobilarov, and J. Moore, "Uncertainty-aware planning for heterogeneous robot teams using dynamic topological graphs and mixed-integer programming," *arXiv preprint arXiv:2310.08396*, 2023.

[9] R. Bogue, "The role of robots in firefighting," *Industrial Robot: the international journal of robotics research and application*, vol. 48, no. 2, pp. 174–178, 2021.

[10] P. Mátyás and N. Máté, "Brief history of uav development," *Repülés-tudományi Közlemények*, vol. 31, no. 1, pp. 155–166, 2019.

[11] C. Zhang and J. A. Shah, "Co-optimizating multi-agent placement with task assignment and scheduling.," in *IJCAI*, pp. 3308–3314, 2016.

[12] X. Wang, J. Hudack, and S. Mou, "Distributed algorithm with resilience for multi-agent task allocation," in *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, pp. 112–117, IEEE, 2021.

[13] D. Ioan, I. Prodan, S. Olaru, F. Stoican, and S.-I. Niculescu, "Mixed-integer programming in motion planning," *Annual Reviews in Control*, vol. 51, pp. 65–87, 2021.

[14] R. Calegari, G. Ciatto, V. Mascardi, and A. Omicini, "Logic-based technologies for multi-agent systems: a systematic literature review," *Autonomous Agents and Multi-Agent Systems*, vol. 35, no. 1, p. 1, 2021.

[15] F. A. Potra and S. J. Wright, "Interior-point methods," *Journal of computational and applied mathematics*, vol. 124, no. 1-2, pp. 281–302, 2000.

[16] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, 2016.

[17] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," *Advances in neural information processing systems*, vol. 29, 2016.

[18] E. A. O. Diallo and T. Sugawara, "Learning strategic group formation for coordinated behavior in adversarial multi-agent with double dqn," in *PRIMA 2018: Principles and Practice of Multi-Agent Systems: 21st International Conference*, pp. 458–466, Springer, 2018.

[19] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

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

[21] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell, "Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 4213–4220, 2019.

[22] Z. Fan, R. Su, W. Zhang, and Y. Yu, "Hybrid actor-critic reinforcement learning in parameterized action space," *arXiv preprint arXiv:1903.01344*, 2019.

[23] W. Böhmer, V. Kurin, and S. Whiteson, "Deep coordination graphs," in *International Conference on Machine Learning*, pp. 980–991, PMLR, 2020.

[24] J. Wang and L. Sun, "Dynamic holding control to avoid bus bunching: A multi-agent deep reinforcement learning framework," *Transportation Research Part C: Emerging Technologies*, vol. 116, p. 102661, 2020.

[25] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[26] M. Botvinick, S. Ritter, J. X. Wang, Z. Kurth-Nelson, C. Blundell, and D. Hassabis, "Reinforcement learning, fast and slow," *Trends in cognitive sciences*, vol. 23, no. 5, pp. 408–422, 2019.

[27] S. M. Devlin and D. Kudenko, "Dynamic potential-based reward shaping," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), pp. 433–440, IFAAMAS, 2012.

[28] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the 16th International Conference on Machine Learning*, pp. 278–287, 1999.

[29] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker, "Surrogate-based analysis and optimization," *Progress in aerospace sciences*, vol. 41, no. 1, pp. 1–28, 2005.

[30] P. Bonami and J. Lee, "Bonmin user's manual," *Numer Math*, vol. 4, pp. 1–32, 2007.