2024 American Control Conference (ACC)
July 8-12, 2024. Toronto, Canada

Distributed Reinforcement Learning For Swarm Systems With Reward Machines

Shayan Meshkat Alsadat, Nasim Baharisangari, Yash Paliwal, and Zhe Xu

Abstract— **We introduce a decentralized reinforcement learning (RL) algorithm for swarm systems where reward machines (a type of Mealy machine) are used to encode the non-Markovian reward functions. We use the generalized moments (GMs) to characterize swarm features. Each agent estimates the GM of the swarm state and uses it to estimate the state of the reward machine. The agent then uses the estimated state of the reward machine to update its q-values. We use the gossip algorithm for communication between the agents. Agents exploit this communication to update their estimated GM of the swarm state, which leads to an update of their estimated state of the reward machine. We derive an upper bound for the error between the estimate GM of the swarm state and the ground truth GM of the swarm state, and using that upper bound; we prove the convergence of our proposed algorithm, swarm q-learning with reward machines (Swarm-QRM). To demonstrate the efficiency of our approach, we present two case studies wherein, for Case Study 1, a swarm of agents will perform a pickup and delivery task, and in Case Study 2, the swarm of agents will conduct a search and rescue task. We also show that Swarm-QRM outperforms the baselines, q-learning in augmented state space (QAS), and Double Deep Q-Network (DDQN).**

I. INTRODUCTION

A large number of autonomous agents or *swarm* can collaborate and accomplish tasks that an individual agent is not capable of performing [1]. Swarms find applications in a diverse array of scenarios, ranging from complex shape formation [2], [3] to missions involving exploration and rescue [1], [4], [5]. In swarm systems, as the number of states and agents increases, it generally leads to the complexity of control. In such cases, decentralized algorithms offer a more viable and efficient solution [1]. A variety of decentralized algorithms have been developed to tackle swarm control challenges, each offering unique advantages. Some notable approaches include algorithms based on geometrical patterns [6], integration with fuzzy control [7], utilization of generalized moments (GMs) such as swarm density [1], and strategies involving the minimization of the second smallest Laplacian eigenvalue [8].

We propose a distributed reinforcement learning (RL) approach for swarm systems that is fully decentralized and exploits reward machines, i.e., a type of Mealy machine, to encode non-Markovian reward functions on the swarm-level. Most of the existing works exploit some centralized entity to incorporate more information into the training. Centralized training and decentralized execution [9], [10] transforms the training phase into a fully observable environment problem, allowing for centralized information gathering and learning, while policy execution remains decentralized. In many complex tasks with sparse rewards, there are high-level structural relationships for the sub-tasks that can be exploited using reward machines. A method known as *q-learning for reward machines* (QRM) proposed by [11], demonstrated the effectiveness of the q-learning with reward machines and showed that QRM would converge to an optimal policy in tabular case. However, in their method agent knows the structure of the reward machine, which reduces its practicality.

In our proposed method for swarm systems, swarm q-learning with reward machines (Swarm-QRM), we use reward machines to incorporate high-level knowledge information into RL; however, instead of providing direct access to the reward machine for agents, we utilize the (GMs) [12] to represent swarm features. Each agent estimates the GM of the *swarm state*, i.e., the combination of the agents' states, in order to estimate the state of the reward machine. We present two case studies; one is a pickup and delivery task, and the other is a search and rescue task. We compare our proposed Swarm-QRM algorithm with two baselines, i.e., q-learning in augmented state space (QAS) and Double Deep Q-Network (DDQN).

Contributions. We make the following contributions: (a) We propose a novel distributed RL algorithm for swarm systems using reward machines. (b) We derive an upper bound for the estimation error between the estimated and the actual GMs and show the proposed RL algorithm convergence properties. (c) We evaluate the proposed algorithm on numerical examples in two case studies, each involving 50 agents.

II. RELATED WORK

Swarm-level RL. In MARL, Mean Field Theory [13] scales effectively for a large number of agents, considering each agent interacts with a finite set of others. It offers mean field Q-learning and Actor-Critic algorithms. Our method utilizes GMs for swarm feature extraction. An alternative is Q-learning [14] in augmented state space (QAS), incorporating label information in q-value updates. Our approach incorporates estimated reward machine states, providing more insight. DDQN [15] extends DQN using two separate neural networks, but demands substantial training data in swarm systems. Deep RL for Swarms [16] exploits

The authors are with School of Matter, Transport and Energy, Arizona State University, Tempe, AZ 85281 {shayan.meshkat,nbaharis,ypaliwa1,xzhe1}@asu.edu

979-8-3503-8265-5/$31.00 ©2024 AACC

swarm homogeneity, treating neighboring agents' state info as a random variable, and applies mean feature embeddings. Our method uses estimated GM of swarm state for encoding, offering permutation-invariant representation, and combines centralized learning with decentralized execution.

Control of Swarm Systems with Reward Machine. Reward machines used for cooperative MARL suggested by [17] where reward machines allow agents to break down cooperative tasks into sub-tasks, enabling decentralized learning. The proposed algorithm trains individual agents using sub-task reward machines, transforming the team's task into a set of single-agent reinforcement learning problems, eliminating the need for agents to learn simultaneously. In our proposed method, we maintain an estimated state of the reward machine for each agent.

Estimating Swarm Features. Our proposed algorithm is a decentralized training methodology based on an estimated state of the reward machine. Agents lack access to the swarm state, making this information hidden from them in order to have a more realistic condition. Despite this limitation, our proposed algorithm forces agents to harmonize their actions. Agents must collectively influence the GM of the swarm state and coordinate their actions to complete a task. Our proposed method addresses communication by establishing a mechanism for agents to exchange information through the gossip algorithm. This implementation encourages cooperative behavior at a high level, thereby facilitating task completion. We demonstrate that our proposed method possesses the guarantee of converging to an optimal policy. This guarantee, coupled with our algorithm's testing in two case studies, underscores its robustness and utility. In our approach, having information about the GM of the swarm state is important since GMs represent swarm features [18], and to estimate the GMs accurately, we adopt a strategy where each agent maintains an estimation. This estimation is about the GM of the swarm state made by other agents. Furthermore, the agents update their estimated GMs of the swarm state as they engage with their environment, which includes other agents. This iterative process persists until the agents achieve a consensus regarding the GMs, ensuring that their estimates align and converge in the limit.

III. Preliminaries

In this section, the basic knowledge of reinforcement learning and reward machines is introduced, which is necessary for developing a reinforcement learning algorithm for swarm systems.

Markov Decision Processes. A labeled Markov decision process (MDP) is defined as a tuple $\boldsymbol{M} = (S, s_I, A, p, \gamma, \mathcal{P}, L)$ where S is a finite set of states, s_I is the agent's initial state, A is a finite set of actions, $p : S \times A \times S \to [0, 1]$ is the transition probability function, $\gamma \in [0, 1)$ is the discount factor, $\mathcal{P}$ is the set of propositional symbols corresponding to high-level events within the environment, and $L : S \times A \times S \to 2^{\mathcal{P}}$ is the labeling function that assigns truth values to the propositional symbols in $\mathcal{P}$.

Remark 1: In our definition of MDP, we do not have a reward function defined since in our proposed method, Swarm-QRM, we define the reward on swarm-level. We explain our swarm definition in detail in Section IV. Additionally, we denote the set cardinality by $|\cdot|$, e.g., $|S|$ means the size of the set S.

We denoted *policy* by π and an agent under *policy* π in state s takes action a with probability $\pi(s, a)$, reaching a new state s' with probability $p(s, a, s')$. In our algorithm, we use a modified q-learning method since our setting is decentralized, which will be explained in detail in Section VII.

IV. Swarm system modeling with reward machine

A. Swarm Model

In this subsection, we present the modeling of the swarm in RL. We first model the communication structure of the swarm using an *undirected graph* defined as follows.

Definition 1: We denote an undirected graph by $G = (\mathcal{C}, \mathcal{E})$, where $\mathcal{C} = \{c_1, c_2, ..., c_{n_C}\}$ is a finite set of nodes, $\mathcal{E} \subseteq \mathcal{E}' = \{e_{1,2}, e_{1,3}, ..., e_{1,n_\mathcal{E}}, e_{2,3}, ..., e_{n_{\mathcal{E}-1},n_\mathcal{E}}\}$ is a finite set of edges where $e_{i,l} \in \mathcal{E}$ if nodes c_i and c_j are connected by an edge in the graph G, and $n_\mathcal{C}, n_\mathcal{E} \in \mathbb{N} = \{1, 2, ...\}$.

Each node c_i of the undirected graph G represents an agent in the swarm. Each edge $e_{i,j}$ connecting the nodes i and j represents the fact that agents i and j are neighbors, i.e., agents i and j can communicate with each other. Hereafter, we denote the set of agents in the MAS by $\mathcal{N}$ and we use $\mathcal{N}_i$ to denote the set of the neighboring agents of agent i. The communication is asynchronous, i.e., only two agents can communicate at each time step t and the probability of each agent i being *active* at time step t is $\frac{1}{|\mathcal{N}|}$ (here active means agent i can initiate communication with one of its neighboring agents).

Also, we denote the adjacency matrix of the graph G with $\boldsymbol{\mathcal{D}}$. For a swarm with $|\mathcal{N}|$ agents, we define a $|\mathcal{N}| \times |\mathcal{N}|$ matrix $\boldsymbol{W}$ where W_{ij} denotes the probability of agent i communicating with agent j.

Assumption 1: The graph G is time-invariant and each agent is aware of the graph structure of the swarm.

The swarm state $\boldsymbol{s} \in S$ is defined as $\boldsymbol{s} = [s^1, ..., s^N]$ where s^i denotes the state of agent i and $N = |\mathcal{N}| \in \mathbb{N}$. In our case, agent i-th state $s^i \in \mathbb{R}^2$ since x and y location of the agents represent their MDP state $s = [x, y]$. We use the generalized moment $\eta \in E \subseteq \mathbb{R}$ to represent a swarm feature. $U^D : S \to \mathbb{R}$ is a map from the swarm state to the generalized moment (GM), i.e., $\eta = U^D(\boldsymbol{s})$ [18] where $D(s)$ is a polynomial function of the agent i-th state s. The

common generalized moments are as follows:

$$\text{mean: } U^D(\boldsymbol{s}) = \boldsymbol{s} \rightarrow \eta_1 = \frac{1}{|\mathcal{N}|}\sum_{i=1}^{|\mathcal{N}|} D(s^i) \tag{1}$$

$$\text{variance: } U^D(\boldsymbol{s}) = (\boldsymbol{s}-\eta_1\mathbf{1}^T)^2 \rightarrow \eta_2 = \frac{1}{|\mathcal{N}|}\sum_{i=1}^{|\mathcal{N}|}(D(s^i)-\eta_1)^2 \tag{2}$$

$$\begin{aligned}\text{skewness: } U^D(\boldsymbol{s}) = \sum_{i=1}^{|\mathcal{N}|}[(D(s^i)-\eta_1)(\eta_2)^{-\frac{1}{2}}]^3 \rightarrow \\ \eta_3 = \frac{1}{|\mathcal{N}|}\sum_{i=1}^{|\mathcal{N}|}[(D(s^i)-\eta_1)(\eta_2)^{-\frac{1}{2}}]^3\end{aligned} \tag{3}$$

$$\begin{aligned}\text{kurtosis: } U^D(\boldsymbol{s}) = \sum_{i=1}^{|\mathcal{N}|}[(D(s^i)-\eta_1)(\eta_2)^{-\frac{1}{2}}]^4 - 3 \rightarrow \\ \eta_4 = \frac{1}{|\mathcal{N}|}\sum_{i=1}^{|\mathcal{N}|}[(D(s^i)-\eta_1)(\eta_2)^{-\frac{1}{2}}]^4 - 3\end{aligned} \tag{4}$$

In the following, we use *swarm GM* to represent the GM of the swarm state for brevity.

Definition 2 (Labeled swarm-MDP): We define the swarm-MDP as a tuple $\mathcal{M} = (N, \mathbb{A}, E, r, T, L^s)$ where N is the number of agents in the swarm, $\mathbb{A}$ is the swarm agent (defined in Subsection IV-C), $E \subseteq \mathbb{R}$ is the set of the generalized moment and $\eta \in E$ is the generalized moment (GM) of the swarm state which represents the swarm features, $r \in \mathbb{R}$ is the swarm-level reward, T corresponds to the transition function defined at the swarm-level $T : S^N \times \mathcal{A}^N \times S^N \rightarrow \mathbb{R}$, and swarm-level labeling function is denoted by $L^s : E \rightarrow 2^{\mathcal{P}}$.

Remark 2: Our definition of the labeled swarm-MDP is modified based on the definition of the *swarMDP* from [19].

B. Reward Machine

Definition 3 (Swarm-level Reward Machine): Reward machine defined on the swarm-level $\mathcal{J} = (V, v_I, E, 2^{\mathcal{P}}, \mathbb{R}, \vartheta, \delta, \sigma, L^s)$ consists of a finite set of states V, an initial state v_I, input alphabet $2^{\mathcal{P}}$, $\vartheta \in \Omega \subseteq \mathbb{R}$ transition error which allows transition to the next state if agents are within this error limit, the transition function $\delta : V \times 2^{\mathcal{P}} \times \Omega \rightarrow V$, and the output function denoted by $\sigma : V \times 2^{\mathcal{P}} \rightarrow \mathbb{R}$.

We established the reward machine at the swarm-level where transitions within the reward machine depend on the swarm GM.

C. Agent Model

Our agent model is based on modifications of the agent model presented in [19].

Definition 4 (Agent): We define the agent as tuple $\mathbb{A} = (S, \mathcal{A}, \hat{v}, \pi)$ where $\mathcal{A}$ is a set of actions, $\hat{v} \in V$ is the agent's estimated state of the reward machine, and $\pi(s, \hat{v}, a)$ is the policy of the agent.

V. Convergence of the GM

A. Generalized Moment Consensus

In this subsection, we introduce an approach for reaching consensus among the swarm based on the gossip algorithm discussed in [20], [12] for one dimension where we apply the same framework to higher dimensions. In what follows, we use $\boldsymbol{\theta}(t)$ to denote the vector $\boldsymbol{\theta}(t) = [D(s^1(t)), D(s^2(t)), ..., D(s^{|\mathcal{N}|}(t))]$, $\theta^i(t) = D(s^i(t))$, and $\Delta\boldsymbol{\theta}(t) = \boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1)$ at discrete time step t. For the swarm, we assume that at time step $t-1$, agents i and j have computed the estimated GM $\zeta^i(t-1)$ and $\zeta^j(t-1)$, respectively. At time step t, agents i and j communicate with each other and update their estimates of the actual GM $\eta(t)$ using Equations (5) and (6), respectively.

$$\zeta^i(t) = \frac{1}{2}(\zeta^i(t-1) + \zeta^j(t-1)) + \theta^i(t) - \theta^i(t-1), \tag{5}$$

$$\zeta^j(t) = \frac{1}{2}(\zeta^i(t-1) + \zeta^j(t-1)) + \theta^j(t) - \theta^j(t-1), \tag{6}$$

and other agents update their estimates of swarm GM using Equation (7)

$$\zeta^k(t) = \zeta^k(t-1) + \theta^k(t) - \theta^k(t-1),\ k \in \mathcal{N} \text{ and } k \neq i, j. \tag{7}$$

We can reformulate the equations of the update of the estimated of the swarm GM by the agents in the vector form using Equation (8)

$$\boldsymbol{\zeta}(t) = \boldsymbol{V}(t)\boldsymbol{\zeta}(t-1) + \boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1), \tag{8}$$

where $\boldsymbol{\zeta}(t) = [\zeta^1(t), \zeta^2(t), ..., \zeta^{|\mathcal{N}|}(t)]^T$ is the vector containing the estimated GMs made by $|\mathcal{N}|$ agents at time step t, and matrix $\boldsymbol{V}(t)$ has a probability of $\frac{1}{|\mathcal{N}|}W_{ij}$ to be equal to $\boldsymbol{\mathcal{V}}_{ij} = I_{|\mathcal{N}|} - \frac{(e_i - e_j)(e_i - e_j)^T}{2}$, where $I_{|N|}$ is a $|\mathcal{N}| \times |\mathcal{N}|$ identity matrix and $e_i = [0, ..., 1, ..., 0]^T$ is a $|\mathcal{N}| \times 1$ vector with the i-th entry to be 1 and zero in all other entries [20], [12]. In [20], it is proved that $\mathbb{E}(\boldsymbol{V}(t)) = \boldsymbol{V}$ is constant at each time step t. In [20], it is shown that the convergence rate of the estimation using Equations (5), (6) and (7) is governed by $0 \leq \lambda(\boldsymbol{V}) < 1$ where λ is the second largest eigenvalue of $\boldsymbol{V}$. We calculate $\boldsymbol{V}$ such that $\lambda(\boldsymbol{V})$ is minimized by solving the following optimization problem.

$$\begin{aligned}\arg\min_{\boldsymbol{V}} \quad & g, \\ \text{subject to} \quad & W_{ij} \geq 0,\ W_{ij} = 0,\ \text{if } e_{i,j} \notin \mathcal{E}, \\ & \boldsymbol{V} = \frac{1}{|\mathcal{N}|}\sum_{i=1}^{|\mathcal{N}|}\sum_{j=1}^{|\mathcal{N}|} W_{ij}\boldsymbol{\mathcal{V}}_{ij}, \\ & \boldsymbol{V} - \frac{1}{|\mathcal{N}|}\mathbf{1}\mathbf{1}^T \preceq gI_{|\mathcal{N}|}, \sum_{j=1}^{|\mathcal{N}|} W_{ij} = 1, \forall i.\end{aligned} \tag{9}$$

In the optimization problem (9), $\boldsymbol{V} - \frac{1}{|\mathcal{N}|}\mathbf{1}\mathbf{1}^T \preceq gI_{|\mathcal{N}|}$ means that $(gI_{|\mathcal{N}|} - \boldsymbol{V} + \frac{1}{|\mathcal{N}|}\mathbf{1}\mathbf{1}^T)$ is semi-definite.

We show that the estimated swarm GM converges to the actual swarm GM. Proof can be shown by first, calculating an upper bound for the estimation error at time step t defined as $\|\boldsymbol{\zeta}(t) - \eta(t)\mathbf{1}\|_\infty$, and then, showing that the estimation error converges to 0 when $t \to \infty$.

Theorem 1: The Upper bound on the actual GM estimation error equals Equation (10) and converges to 0 as $t \to 0$.

$$\rho(t) = \lambda^t(\boldsymbol{V})\sqrt{N}\zeta_{\max} + \sum_{t'=1}^{t} \lambda^{t-t'}(\boldsymbol{V})\theta_{\max} \tag{10}$$

where it is assumed that each agents knows $\mathbb{E}(\|\boldsymbol{\zeta}[0] - \overline{\zeta}[0]\mathbf{1}\|_\infty) \leq \zeta_{\max}$ with $\overline{\zeta}[0] = \frac{1}{|\mathcal{N}|}\sum_{i=1}^{|\mathcal{N}|}\zeta_i[0]$ also $\sqrt{\mathbb{E}(\|\boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1)\|^2)} \leq \theta_{\max}$ where $\zeta_{\max}$ is a user-defined scalar value and $\theta_{\max}$ is based on the MDP environment.

For instance, $\theta_{\max}$ in discrete MDP is 1 since each agent can change its state s maximum by 1 unit at each time step.

VI. Framework

In this section, we present a framework for distributed reinforcement learning for swarm systems using reward machines. Each agent uses its estimated swarm GM, ζ^i, to estimate the state of the reward machine $\hat{v}^i$ and then uses it to update its q-values. With this strategy, the q-functions are in the augmented state-space $q(s, v, a)$. Note that our approach only provides the agents with an estimated state of the reward machine, not the ground truth state of the reward machine. The reason is that the agents in our proposed method only have access to the estimated swarm GM, ζ, rather than the actual swarm GM, η.

The swarm is modeled as a labeled swarm-MDP $\mathcal{M}$ as defined in Definition 2. Agents use asynchronous communication in order to collect information from their neighbors to improve their estimated swarm GMs. During the asynchronous communication, agents update their estimations using Equations (5)-(7). Before proceeding, there are two points to clarify (1) In Swarm-QRM, we require the swarm GM to reach the *target states* to complete the task, meaning that we do not require the individual states of the agents to reach the target states (where the reward is given to the swarm for completing the task). (2) Each agent only communicates with its neighboring agents according to the graph structure of the swarm G.

We present our algorithm, Swarm-QRM, in Algorithm 1. First, we initialize the states for each agent and choose an action based on the initial q-values (Lines 1-4). Then Swarm-QRM chooses a random agent from the graph G and randomly chooses one of its neighbors for communication (Lines 5-7). Other agents will also update their estimated GMs (Line9). Each agent, based on its estimated GM, will update its estimated state of the reward machine (Line 11), then the swarm receives a reward based on the actual swarm GM, and agents will use this reward to update their q-values

Algorithm 1: Swarm q-learning with reward machine (Swarm-QRM)

1 **Hyperparameter**: episode length $episode\text{-}length$, learning rate α, discount factor γ
2 **Input**: reward machine as $(V, v_I, E, 2^P, \mathbb{R}, \delta, L)$ and q-function $Q(S, a)$
1: $s^i \leftarrow InitializeState()$, $\zeta^i \leftarrow s^i$, $\hat{v}^i \leftarrow \hat{v}^i_0$
2: **for** $0 \leq t \leq episode\text{-}length$ **do**
3: $a \leftarrow GetEpsilonGreedyAction(q^i, s^i)$
4: $s', done \leftarrow EnvExecuteAction(s, a)$
5: **if** $AgentIsActive(i)$ **then**
6: Communicate with neighbor $l \in |\mathcal{N}|_i$
7: $\zeta'^i \leftarrow \frac{(\zeta^i + \zeta^l)}{2} + \theta'^i - \theta^i$
8: **else**
9: $\zeta'^i \leftarrow \zeta^i + \theta'^i - \theta^i$
10: **end if**
11: $\hat{v}'^i \leftarrow \delta(\hat{v}, \vartheta, L(\zeta^i, a, \zeta'^i))$
12: Agent i recieves reward r from the ground truth reward machine
13: $q(s^i, \hat{v}^i, a) \leftarrow (1-\alpha)q(s^i, \hat{v}^i, a) + \alpha(r + \gamma \max_{a' \in A} q(s'^i, \hat{v}'^i, a'))$
14: **if** $done$ **then**
15: $StartNextEpisode()$
16: **end if**
17: $s^i \leftarrow s', v^i \leftarrow v', \zeta^i \leftarrow \zeta'^i$
18: **end for**

(Lines 11-13). Then, the Boolean variable $done$ is set to *True* if a terminal condition is met by the actual swarm GM (Line 14-16). Then the Swarm-QRM update the agent's state (s^i), initial state of the reward machine ($\hat{v}^i$), and estimated GM (ζ^i) (Line 17).

VII. Convergence of Swarm-QRM

In this section, we prove that using Swarm-QRM, the policy of each agent i converges to its optimal policy $\pi^{i,*}$. Bellman equation in the augmented state space follows the form $q(s^i, \hat{v}^i, a) \leftarrow (1-\alpha)q(s^i, \hat{v}^i, a) + \alpha(r + \gamma \max_{a' \in A} q(s'^i, \hat{v}'^i, a'))$. At time step $t = 0$, each agent i initializes its q-value while being at the initial state of the reward machine v_0, the initial state of agent i is s^i_0, and the initial estimated GM of the agent i, is $\zeta^i(0)$. Each agent then updates its estimation using the gossip algorithm. At each time step t, each agent i uses its estimation ζ^i to update its state of the reward machine using

$$\hat{v}^i_{t+1} = \delta(\hat{v}^i_t, L^s(\zeta^i(t), a, \zeta^i(t+1))) \tag{11}$$

which shows that the transition of $\hat{v}^i_t$ to $\hat{v}^i_{t+1}$ is a function of $\zeta^i(t)$ and $\zeta^i(t+1)$. Now, in order to show that Swarm-QRM converges to the optimal policy, we need to show that $v_t = \hat{v}^i_t, \forall i \in \mathcal{N}$ when $t \to \infty$.

First, we define the constraint in the ground truth reward machine in Definition 3. We denote the label associated with a target state κ by Λ, and define the following constraint.

$$v_{t+1} \neq v_t \text{ iff } |\eta(t) - \kappa|_d < \mu \tag{12}$$

where $|\cdot|_d$ denotes the absolute value of the difference between the actual swarm GM η and the target state, κ in the

d-th dimension. Constraint (12) enforces that the transition from v_t to the next state occurs only when the distance between $\eta(t)$ and κ is less than $\mu \in \mathbb{R}$ (user defined upper bound on the absolute value of the difference between the actual swarm GM and the target state) in dimension d, i.e., the label associated with Λ is $True$ only when the distance between $\eta(t)$ and κ is less than μ in dimension d (reward machine state transitioned to the next state; thus, it is not equal to its previous time step, $v_{t+1} \neq v_t$).

In order to prove that $v_t = \hat{v}_t^i, \forall i \in \mathcal{N}$ when $t \to \infty$, we need to guarantee that the transitions of the estimated states of the reward machine follow the same sequence of transitions as the actual states of the reward machine.

Using triangle inequality, we define the following constraint for the transition of $\hat{v}_t^i$ to $\hat{v}_{t+1}^i$.

$$\hat{v}_{t+1}^i \neq \hat{v}_t^i \text{ iff } |\zeta^i(t) - \kappa|_d < \mu - \epsilon \tag{13}$$

where ϵ is a user defined upper bound on the $\rho(t)$ in Equation (10) which is an upper bound on the estimation error $\|\boldsymbol{\zeta}(t) - \eta \mathbf{1}\|_\infty$, and $\mu - \epsilon > 0$ (transition error mentioned in Definition 3). Constraints (13) states that the transition of the estimated states of the reward machine to the next state occurs only when the absolute value of the difference between the estimated swarm GM $\zeta^i(t)$ and the target state κ in the d-th dimension is less than $\mu - \epsilon$. In other words, the label is Λ is $True$ only when the absolute difference of the estimated swarm GM, $\zeta^i(t)$ and the target state κ in the d-th dimension is less than $\mu - \epsilon$. Hence, the estimated state of the reward machine $\hat{v}_t^i$ follows the same sequence of transitions as the actual state of the reward machine v_t.

VIII. Case Studies

In our case studies, swarm agents must complete a sequence of tasks in a specific order. They synchronize their actions to ensure each task is finished before moving to the next. Our focus is on using first-order and second-order swarm GMs to describe swarm behavior. We use the QAS and DDQN baselines as mentioned in the introduction since these two methods are proven to converge to optimal policy for swarm systems using reward machines.

$$\bar{\zeta}_1(t) = \frac{1}{|\mathcal{N}|} \sum_{i=1}^{|\mathcal{N}|} \zeta^i(t) \tag{14}$$

$$\bar{\zeta}_2(t) = \frac{1}{|\mathcal{N}|} \sum_{i=1}^{|\mathcal{N}|} (\zeta^i(t) - \bar{\zeta}_1(t))^2 \tag{15}$$

In Equation (14) and Equation (15) the $\bar{\zeta}_1(t)$ and $\bar{\zeta}_2(t)$ are the first-order and second-order swarm GMs respectively.

A. Case Study 1

In our environment, a swarm of agents coordinate their actions and collaboratively achieve a predefined sequence of sub-tasks while avoiding obstacles and maintaining a certain formation. The agents in the swarm must first reach the *pickup* location before proceeding to reach the delivery destination. Thus, the order of task execution, determined by the truth value of the labels based on the swarm GM, is important. Figure 1 demonstrates the task.

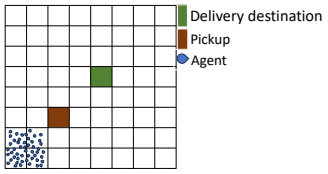

Fig. 1. Swarm must first reach the pickup location before proceeding to reach the delivery location.

The reward machine associated with Case Study 1 is shown in Figure 2.

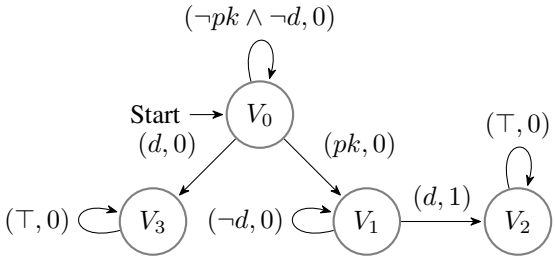

Fig. 2. Reward machine for Case Study 1. Swarm must first reach the pickup point pk, then move to the delivery destination d, meaning the transition from v_0 state to v_1 state while the associated reward is 0.

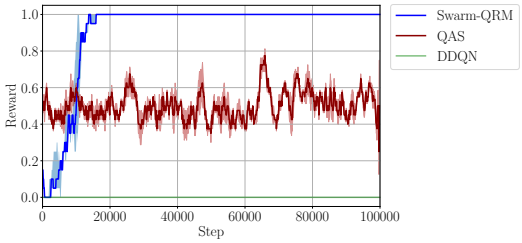

Fig. 3. Cumulative reward of 5 independent runs averaged for each 2000 steps for Case Study 1.

Figure 3 shows that the Swarm-QRM reaches the optimal policy within 20000 steps while the QAS reached 50% of the optimal policy, and DDQN is stuck at 0 cumulative rewards.

B. Case Study 2

In Case Study 2, the swarm must conduct a search and rescue task. The swarm must first find the survivors and then carry them to a safe location; however, there are regions that are forbidden for agents in the swarm (fire locations). Figure 4 demonstrates the task.

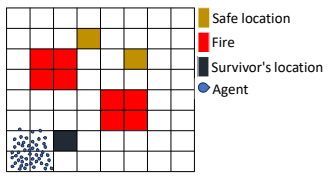

Fig. 4. Agents must first find the survivors' location and then carry the survivors to either of the two safe places while avoiding fires.

The associated reward machine for Case Study 2 is demonstrated in Figure 5.

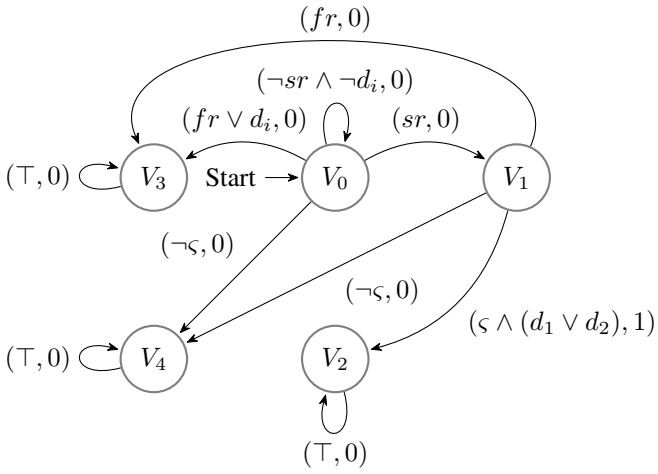

Fig. 5. Reward machine for Case Study 2. sr means finding the survivors; d_1 means safe location 1; d_2 means safe location 2; d_i means either d_1 or d_2; fr means fire (forbidden region); ς means second-order GM. Edge between v_0 and v_1 labeled as $(sr, 0)$ means transition from state of the reward machine v_0 to v_1 if sr is *True*, returning reward of 0.

In Case Study 2, agents in the swarm must first reach the survivor's location before going to any of the safe locations. We use ς or the second-order moment GM to control the spread of the agents in the swarm.

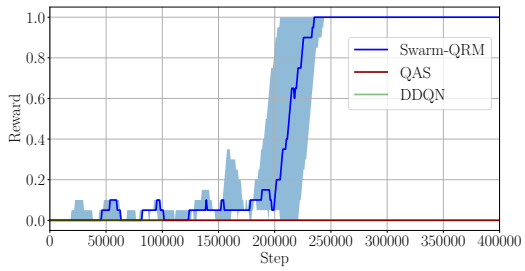

Fig. 6. cumulative reward of 5 independent runs averaged for each 2000 steps for Case Study 2.

Figure 6 shows that the Swarm-QRM reaches the optimal policy within 250000 steps while the QAS and DDQN are stuck at zero cumulative rewards for up to 500000 steps (with 400000 steps shown in Figure 6).

IX. CONCLUSIONS

We have proposed a decentralized reinforcement learning algorithm for swarm systems using reward machines. We have shown that incorporating the estimated state of the reward machine to each agent's q-values improves the reinforcement learning, meaning our proposed algorithm reaches the optimal policy sooner compared to the baselines. Furthermore, we have demonstrated that our algorithm will almost surely converge to the optimal policy. In the future, we will extend our method to deal with dynamic surroundings, e.g., dynamic graph structures or environments with partial observability.

ACKNOWLEDGMENT

This research is partially supported by the National Science Foundation under grants CNS 2304863 and CNS 2339774, and the Office of Naval Research under grant ONR N00014-23-1-2505.

REFERENCES

[1] Franck Djeumou, Zhe Xu, Murat Cubuktepe, and Ufuk Topcu. Probabilistic control of heterogeneous swarms subject to graph temporal logic specifications: A decentralized and scalable approach. *IEEE Transactions on Automatic Control*, 68(4):2245–2260, 2022.

[2] Alex Kushleyev, Daniel Mellinger, Caitlin Powers, and Vijay Kumar. Towards a swarm of agile micro quadrotors. *Autonomous Robots*, 35(4):287–300, 2013.

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

[4] James Kennedy. Swarm intelligence. In *Handbook of nature-inspired and innovative computing: integrating classical models with emerging technologies*, pages 187–219. Springer, 2006.

[5] Ruixuan Yan and Agung Julius. A decentralized b&b algorithm for motion planning of robot swarms with temporal logic specifications. *IEEE Robotics and Automation Letters*, 6(4):7389–7396, 2021.

[6] Teddy M Cheng and Andrey V Savkin. Decentralized control of multi-agent systems for swarming with a given geometric pattern. *Computers & Mathematics with Applications*, 61(4):731–744, 2011.

[7] Bijan Ranjbar-Sahraei, Faridoon Shabaninia, Alireza Nemati, and Sergiu-Dan Stan. A novel robust decentralized adaptive fuzzy control for swarm formation of multiagent systems. *IEEE Transactions on Industrial Electronics*, 59(8):3124–3134, 2012.

[8] Maria Carmela De Gennaro and Ali Jadbabaie. Decentralized control of connectivity for multi-agent systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 3628–3633. IEEE, 2006.

[9] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[10] Yihe Zhou, Shunyu Liu, Yunpeng Qing, Kaixuan Chen, Tongya Zheng, Yanhao Huang, Jie Song, and Mingli Song. Is centralized training with decentralized execution framework centralized enough for marl? *arXiv preprint arXiv:2305.17352*, 2023.

[11] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pages 2107–2116. PMLR, 2018.

[12] Ruixuan Yan and Agung Julius. Distributed consensus-based online monitoring of robot swarms with temporal logic specifications. *IEEE Robotics and Automation Letters*, 7(4):9413–9420, 2022.

[13] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In *International conference on machine learning*, pages 5571–5580. PMLR, 2018.

[14] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[15] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[16] Maximilian Hüttenrauch, Sosic Adrian, Gerhard Neumann, et al. Deep reinforcement learning for swarm systems. *Journal of Machine Learning Research*, 20(54):1–31, 2019.

[17] Cyrus Neary, Zhe Xu, Bo Wu, and Ufuk Topcu. Reward machines for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2007.01962*, 2020.

[18] Ruixuan Yan, Zhe Xu, and Agung Julius. Swarm signal temporal logic inference for swarm behavior analysis. *IEEE Robotics and Automation Letters*, 4(3):3021–3028, 2019.

[19] Adrian Šošić, Wasiur R KhudaBukhsh, Abdelhak M Zoubir, and Heinz Koeppl. Inverse reinforcement learning in swarm systems. *arXiv preprint arXiv:1602.05450*, 2016.

[20] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE transactions on information theory*, 52(6):2508–2530, 2006.