Swarm-Level Task Learning via Generalized Moments in Reinforcement Learning With Reward Machines

Shayan Meshkat Alsadat, Vivek Sunil Kulkarni, and Zhe Xu, *Member, IEEE*

***Abstract*—Swarm-level task learning provides a basis for learning complex tasks in swarm systems using swarm-level features. We propose a method, SwaRM-L, to learn reward machines (RMs) that encode non-Markovian reward functions. We use reward machines to specify the task and its temporal structure. Our approach enables a swarm of agents to learn an RM that eventually becomes equivalent to ground truth RM (i.e., the specified task) in this environment (agents have no access to the ground truth RM and only learn it through environment interaction). We use generalized moments (GMs) to characterize swarm features and estimate the RM state. Each agent maintains an estimated GM to contribute to the collective learning process. The agents use a gossip algorithm to communicate with neighbors and update their estimated GMs. We prove that our method converges to an optimal policy and learns an equivalent RM to the ground truth RM within the environment. We evaluate our proposed method in three case studies involving forty agents with homogeneous dynamics. Our results demonstrate the effectiveness of our method in learning complex swarm behaviors.**

***Index Terms*—Automata learning, generalized moments, reinforcement learning, reward machines, swarm systems.**

I. Introduction

A SWARM of typically twenty or more agents can tackle tasks unmanageable by individuals [1]. Swarms apply to robotics, human-robot interaction, and intelligent systems, but controlling more agents increases complexity. Distributed control, though harder than centralized, offers robustness and scalability [2]. We propose a distributed reinforcement learning (RL) for decentralized swarm training and execution. Used in search and rescue, swarms provide robustness and scalability, yet struggle with partial observability, coordination, sparse rewards, and exponential state-action growth. We use reward machines (RMs) [3], a type of Mealy machine to encode task structures and define Markovian or non-Markovian reward functions for RL agents. In swarms, learning these structures is challenging due to partial observability, coordination, and swarm property preservation. RMs formally capture temporal dependencies, enabling temporal abstraction and sub-task decomposition. Unlike prior work [4] relying on known structures or expert demonstrations, SwaRM-L (Swarm RL with Reward Machines Learning) uses swarm-level features and generalized moments (GMs) to guide RM learning. RMs despite formal languages such as linear temporal logic (LTL) can encode task progression with rewards and state transitions, enabling efficient learning of reward and temporal structures by using swarm-level features, unlike LTL, which lacks reward integration.

Related Works: RL for Swarm Systems. Previous work in swarm systems has utilized Mean Field Theory to manage large agent populations [5]. Deep RL approaches leverage swarm homogeneity by treating neighboring agents' state information as random variable samples [6]. Value decomposition networks [7] tackle credit assignments in cooperative settings but lack support for temporal task specifications. **RL with Reward Machines.** Methods like q-learning with reward machine (QRM) [3] demonstrate the advantages of RMs for temporal task specifications, while counterfactual RMs [8] extend their application to multi-agent systems. However, these approaches often assume prior knowledge of the RM. Our work uses automata learning with satisfiability (SAT) checks [9] to learn RM structures. Centralized training with decentralized execution has also advanced multi-agent RL [10]. **Swarm Features.** GMs characterize swarm behavior in a permutation-invariant way [11]. Gossip algorithms aid coordination [12], but cannot learn reward structures. Learning RMs for swarms requires balancing local autonomy and global coordination. Existing methods often lack scalability or temporal awareness, but SwaRM-L uses GMs in conjunction with RMs to represent swarm behavior in a decentralized, task-relevant manner. Our work has the following contributions: (a) a framework that allows swarms to collectively learn RMs that eventually become equivalent to ground truth RMs within the environment through environment interaction, unlike prior methods that presume RMs are known or use expert demonstrations; (b) We learn the RM (i.e., the specified task) structure by utilizing GMs to characterize swarm

Received 17 March 2025; revised 5 May 2025; accepted 17 May 2025. Date of publication 26 May 2025; date of current version 11 June 2025. This work was supported in part by NSF under Grant CNS 2304863, Grant CNS 2339774, and Grant IIS 2332476; and in part by Office of Naval Research (ONR) under Grant N00014-23-1-2505. Recommended by Senior Editor C. Briat. *(Corresponding author: Zhe Xu.)*

The authors are with the School for Engineering of Matter, Transport and Energy, Arizona State University, Tempe, AZ 85281 USA (e-mail: smeshka1@asu.edu; vkulka16@asu.edu; xzhe1@asu.edu).

Digital Object Identifier 10.1109/LCSYS.2025.3573938

behavior, providing a compact, permutation-invariant swarm state representation vital for scalability; and (c) theoretical guarantee of convergence to an optimal policy, along with an upper bound error between estimated and actual swarm GMs. We validate our approach in three case studies, showing faster convergence to an optimal policy while preserving swarm coordination and distribution. Agents use local observations and gossip-based communication to update GMs and learn the RM structure.

II. Preliminaries

Before proceeding, we define the necessary background.

Labeled Markov Decision Process (MDP). A labeled MDP is defined as a tuple $\mathcal{G} = (\mathcal{S}, s_I, A, p, \gamma, \mathcal{P}, L)$ consisting of a finite set of states $\mathcal{S}$, an initial state s_I, a finite set of actions A, a transition probability function $p : \mathcal{S} \times A \times \mathcal{S} \to [0, 1]$, a discount factor $\gamma \in [0, 1)$, a finite set of atomic propositions $\mathcal{P}$, and a labeling function $L : \mathcal{S} \times A \times \mathcal{S} \to 2^{\mathcal{P}}$ that maps transitions to sets of propositions. We denote the floor function by $\lfloor\cdot\rfloor$, norm-2 by $\|\cdot\|$, and infinity norm by $\|\cdot\|_\infty$.

A policy $\pi(s, a)$ defines the probability of taking action $a \in A$ in state $s \in \mathcal{S}$. A trajectory $s_0 a_0 s_1 \ldots s_k a_k s_{k+1}$, $k \in \mathbb{N}$ represents states and actions visited by an agent at time step k, with corresponding label sequence $\lambda = l_0 l_1 \ldots l_k$ where label at time step k is $l_k = L(s_k, a_k, s_{k+1})$. A trace is a pair (λ, ρ) consisting of a label sequence λ and corresponding reward sequence $\rho = r_0 r_1 \ldots r_k$ with r_k is the reward at time step k.

Definition 1 (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, $E \subseteq \mathbb{R}$ 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 (obtained from Definition 2), T corresponds to the transition function defined at the swarm-level $T\colon S^N \times \mathcal{A}^N \times S^N \to \mathbb{R}$, and swarm-level labeling function is denoted by $L^s\colon E \to 2^{\mathcal{P}}$ [4].

Definition 2 (Swarm Reward Machine (RM)): A swarm RM is a tuple $\mathcal{A} = (\mathtt{V}, v_I, 2^{\mathcal{P}}, L^s, \delta, \sigma, \vartheta)$ where $\mathtt{V}$ denotes a finite set of states, $v_I \in \mathtt{V}$ denotes the initial state, $2^{\mathcal{P}}$ denotes the input alphabet (sets of atomic propositions). The transition error allows transition to the next state if the swarm is within this error limit, denoted by $\vartheta \in \Omega \subseteq \mathbb{R}$. The transition function is denoted by $\delta\colon \mathtt{V} \times 2^{\mathcal{P}} \times \Omega \to \mathtt{V}$ and the output function $\sigma\colon \mathtt{V} \times 2^{\mathcal{P}} \to \mathbb{R}$ that assigns the reward [4].

III. Generalized Moments for Swarm Systems

We use undirected graphs to model the communication topology of the swarm (see Definition 3).

Definition 3: We denote an undirected graph used by the swarm by $G = (\mathcal{C}, \mathfrak{E})$, where $\mathcal{C} = \{c_1, c_2, \ldots, c_{n_C}\}$ is a finite set of nodes, $\mathfrak{E} \subseteq \mathfrak{E}' = \{e_{1,2}, e_{1,3}, \ldots, e_{1,n_\mathfrak{E}}, e_{2,3}, \ldots, e_{n_{\mathfrak{E}-1},n_\mathfrak{E}}\}$ is a finite set of graph edges. Moreover, $e_{\mathtt{i}+l} \in \mathfrak{E}$ if nodes $c_\mathtt{i}$ and c_l are connected by an edge in the graph G, and $n_\mathcal{C}, n_\mathfrak{E} \in \mathbb{N}$. In $\mathfrak{E}$, $e_{\mathtt{i}+l}$ represents the edge that connects the nodes $c_\mathtt{i}$ and c_l.

In graph G, each node $c_\mathtt{i}$ represents an agent, and an edge $e_{\mathtt{i},\mathtt{j}}$ denotes communication between agents $\mathtt{i}$ and $\mathtt{j}$. The swarm is represented by $\mathcal{N}$, and $\mathcal{N}_\mathtt{i}$ is the neighbor set of agent $\mathtt{i}$. Communication is asynchronous, with two agents interacting at each time step k. An agent is *active* with probability $\frac{1}{|\mathcal{N}|}$. The adjacency matrix is $\mathcal{D}$, and $\boldsymbol{W}$ specifies communication probabilities, with $W_{\mathtt{ij}}$ representing the communication probability between $\mathtt{i}$ and $\mathtt{j}$. Each agent knows the time-invariant graph structure.

The swarm state is denoted by $\boldsymbol{s} \in \mathcal{S}$, where $\boldsymbol{s} = [\mathtt{h}^1, \ldots, \mathtt{h}^N]$ and $\mathtt{h}^\mathtt{i}$ is the observation of agent $\mathtt{i}$. The total number of agents in the swarm is $N = |\mathcal{N}|$. Here, the state $\mathtt{h}^\mathtt{i} = [\mathtt{h}^\mathtt{i}_1, \ldots, \mathtt{h}^\mathtt{i}_N]$ where $\mathtt{h}^\mathtt{i}$ is the observation of agent $\mathtt{i}$. For instance, $\mathtt{h}^\mathtt{i}_1 = [x^\mathtt{i}, y^\mathtt{i}]$ representing the x and y coordinates of the agent $\mathtt{i}$'s position, corresponding to its MDP state.

A generalized moment $\eta \in E \subseteq \mathbb{R}$ is used to describe a feature of the swarm. This is computed using a mapping function $U^\mathtt{Y} : \mathcal{S} \to \mathbb{R}$, where $\eta = U^\mathtt{Y}(\boldsymbol{s})$ and $U^\mathtt{Y}(\boldsymbol{s}) = \frac{1}{N}\sum_{\mathtt{i}=1}^{|\mathcal{N}|} \mathtt{Y}(\mathtt{h}^\mathtt{i})$ [11]. Here, $\mathtt{Y}(\mathtt{h}^\mathtt{i})$ is a polynomial function of the state s of an individual agent. We define swarm GMs as follows: mean (first-order) (1), variance (second-order, spatial dispersion) (2), skewness (third-order, asymmetry) (3), and kurtosis (fourth-order, clustering) (5).

$$U^\mathtt{Y}(\boldsymbol{s}) = \boldsymbol{s} \to \eta_1 = \frac{1}{|\mathcal{N}|}\sum_{i=1}^{|\mathcal{N}|} \mathtt{Y}\big(\mathtt{h}^\mathtt{i}\big) \tag{1}$$

$$U^\mathtt{Y}(\boldsymbol{s}) = \big(\boldsymbol{s} - \eta_1 \mathbf{1}^T\big)^2 \to \eta_2 = \frac{1}{|\mathcal{N}|}\sum_{i=1}^{|\mathcal{N}|}\big(\mathtt{Y}(\mathtt{h}^\mathtt{i}) - \eta_1\big)^2 \tag{2}$$

$$U^\mathtt{Y}(\boldsymbol{s}) = \sum_{i=1}^{|\mathcal{N}|}\Big[\big(\mathtt{Y}(\mathtt{h}^\mathtt{i}) - \eta_1\big)(\eta_2)^{-\frac{1}{2}}\Big]^3 \to \tag{3}$$

$$\eta_3 = \frac{1}{|\mathcal{N}|}\sum_{i=1}^{|\mathcal{N}|}[\big(\mathtt{Y}(\mathtt{h}^\mathtt{i}) - \eta_1\big)(\eta_2)^{-\frac{1}{2}}]^3 \tag{4}$$

$$U^\mathtt{Y}(\boldsymbol{s}) = \sum_{i=1}^{|\mathcal{N}|}\Big[\big(\mathtt{Y}(\mathtt{h}^\mathtt{i}) - \eta_1\big)(\eta_2)^{-\frac{1}{2}}\Big]^4 - 3 \to \tag{5}$$

$$\eta_4 = \frac{1}{|\mathcal{N}|}\sum_{i=1}^{|\mathcal{N}|}\Big[\big(\mathtt{Y}(\mathtt{h}^\mathtt{i}) - \eta_1\big)(\eta_2)^{-\frac{1}{2}}\Big]^4 - 3 \tag{6}$$

Each agent maintains moment estimates $\zeta^\mathtt{i}(k)$, updated via local communication with neighbors. These generalized moments offer a permutation-invariant representation of the swarm state, which is essential for scalable learning.

Example 1: A swarm of forty agents tasked with collaborative objective (Figure 1) does not know the task structure, but learns it through environmental interaction, gradually learning the task and its temporal structure via RM learning.

IV. Swarm GM Convergence

We use the gossip algorithm's error property (Proposition 1) in conjunction with swarm GM evolution in labeled swarm-MDP $\mathcal{M}$ to prove the convergence of the GM estimation process and derive a bound on the estimation error [4].

When agent $\mathtt{i}$ communicates with neighbor $\mathtt{j}$ at time k, they update their GM estimates according to (7) and (8).

$$\zeta^\mathtt{i}(k+1) = \frac{1}{2}\big(\zeta^\mathtt{i}(k) + \zeta^\mathtt{j}(k)\big) + \theta^\mathtt{i}(k+1) - \theta^\mathtt{i}(k) \tag{7}$$

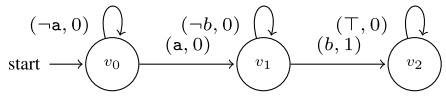


Fig. 1. Ground truth RM for Example 1: Swarm first collects a once enough agents have gathered (first-order swarm GM reaches a desired value), then delivers it to destination b.

$$\zeta^{\mathtt{j}}(k+1) = \frac{1}{2}\big(\zeta^{\mathtt{i}}(k) + \zeta^{\mathtt{j}}(k)\big) + \theta^{\mathtt{j}}(k+1) - \theta^{\mathtt{j}}(k) \quad (8)$$

where $\theta^{\mathtt{i}}(k)$ denotes agent $\mathtt{i}$'s local state observation at time k. Others update their estimates of swarm GM using (9).

$$\zeta^{\mathtt{w}}(k+1) = \zeta^{\mathtt{w}}(k) + \theta^{\mathtt{w}}(k+1) - \theta^{\mathtt{w}}(k),\ \mathtt{w} \in \mathcal{N} \text{ and } \mathtt{w} \neq \mathtt{i}, \mathtt{j}. \quad (9)$$

Writing (7) and (8) in vector form, we have:

$$\boldsymbol{\zeta}(k+1) = \boldsymbol{V}(k)\boldsymbol{\zeta}(k) + \Delta\boldsymbol{\theta}(k) \quad (10)$$

In (10), $\boldsymbol{\zeta}(k) = [\zeta^1(k), \ldots, \zeta^{|\mathcal{N}|}(k)]^T$ denotes a vector of estimated GMs at time step k. The matrix $\boldsymbol{V}(k)$ represents the communication topology and has a probability of $\frac{1}{|\mathcal{N}|}W_{\mathtt{ij}}$ to be equal to $\mathcal{V}_{\mathtt{ij}} = I_{|\mathcal{N}|} - \frac{(e_{\mathtt{i}}-e_{\mathtt{j}})(e_{\mathtt{i}}-e_{\mathtt{j}})^T}{2}$ where $I_{|\mathcal{N}|}$ is a $|\mathcal{N}| \times |\mathcal{N}|$ identity matrix and $e_{\mathtt{i}} = [0, \ldots, 1, \ldots, 0]^T$ is a $|\mathcal{N}| \times 1$ vector with only the $\mathtt{i}$-th entry to be one [13], [14].

To show the convergence of the swarm GM estimation, we first prove that the estimation error is bounded. Let $\lambda(\boldsymbol{V})$ be the second-largest eigenvalue of $\boldsymbol{V}$, and $\overline{\zeta}(0) = \frac{\mathbf{1}^T\boldsymbol{\zeta}(0)}{|\mathcal{N}|}$ be the initial estimation error, where $\boldsymbol{s}(0) = \boldsymbol{\zeta}(0)$ and $\mathbf{1}$ is a $|\mathcal{N}| \times 1$ vector of ones. Proposition 1 demonstrates that the estimation error is bounded by the initial error.

Proposition 1: The GM estimation error satisfies $\mathbb{E}\|\boldsymbol{\zeta}(k) - \eta(k)\mathbf{1}\|_\infty \leq \rho(k)$, with initial bound $\mathbb{E}(\|\boldsymbol{\zeta}(0) - \overline{\zeta}(0)\mathbf{1}\|^2) \leq \mathbb{E}\|\boldsymbol{\zeta}(0)\|^2$, a result from gossip algorithms [13], and converges to zero as $k \to \infty$ with $\lambda(\boldsymbol{V}) < 1$. Also, the one-step estimation error of $\overline{\zeta}(0)$ satisfies $\mathbb{E}(\|\boldsymbol{V}(0)\boldsymbol{\zeta}(0) - \overline{\zeta}(0)\mathbf{1}\|^2) = \mathbb{E}(\|\boldsymbol{V}(0)(\boldsymbol{\zeta}(0) - \overline{\zeta}(0)\mathbf{1})\|^2) \leq \lambda^2(\boldsymbol{V})\|\boldsymbol{\zeta}(0)\|^2$, and k-step estimation error of $\boldsymbol{\zeta}(0)$ satisfies [13]:

$$\mathbb{E}\Big(\|(\boldsymbol{V}(k-1))\cdots\boldsymbol{V}(0)\overline{\zeta}(0)\mathbf{1}\|^2\Big) \leq \lambda^{2k}\|\boldsymbol{\zeta}(0)\|^2 \quad (11)$$

We assume that each agent knows that the initial estimation error is bounded, i.e., $\mathbb{E}(\|\boldsymbol{\zeta}(0) - \overline{\zeta}(0)\mathbf{1}\|_\infty) \leq \zeta_{\max}$. Starting with $\mathbb{E}(\|\zeta(k) - \eta(k)\mathbf{1}\|_\infty)$, we apply Jensen's inequality to bound via the 2-norm: $\mathbb{E}(\|\boldsymbol{\zeta}(k) - \eta(k)\mathbf{1}\|_\infty) \leq \mathbb{E}(\|\boldsymbol{\zeta}(k) - \eta(k)\mathbf{1}\|)$, as in (12) where $\Delta\eta(k) = \eta(k) - \eta(k-1)$.

$$\begin{aligned}
&\mathbb{E}(\|\boldsymbol{\zeta}(k) - \eta(k)\mathbf{1}(k)\|_\infty) \leq \mathbb{E}(\|\boldsymbol{V}(k-1)\cdots\boldsymbol{V}(0)\boldsymbol{\zeta}(0) + \\
&\boldsymbol{V}(k-1)\cdots\boldsymbol{V}(1)\Delta\boldsymbol{\theta}(1) + \cdots + \Delta\boldsymbol{\theta}(k) - [\eta(k) - \eta(k-1) \\
&+ \eta(k-1) - \eta(k-2) + \cdots + \eta(1) - \overline{\zeta}(0) + \overline{\zeta}(0)]\mathbf{1}\|) \\
&= \mathbb{E}(\|\boldsymbol{V}(k-1)\cdots\boldsymbol{V}(0)\boldsymbol{\zeta}(0) - \overline{\zeta}(0)\mathbf{1} + \boldsymbol{V}(k-1) \\
&\cdots\boldsymbol{V}(1)\Delta\boldsymbol{\theta}(1) - \Delta\eta(1)\mathbf{1} + \cdots + \Delta\boldsymbol{\theta}(k) - \Delta\eta(k)\mathbf{1}\|) \\
&\leq \mathbb{E}(\|\boldsymbol{V}(k-1)\cdots\boldsymbol{V}(0)\boldsymbol{\zeta}(0) - \Delta\eta(0)\mathbf{1}\|) + \mathbb{E}(\|\boldsymbol{V}(k-1)\cdots \\
&\boldsymbol{V}(1)\Delta\boldsymbol{\theta}(1) - \Delta\eta(1)\mathbf{1}\|) + \cdots + \mathbb{E}(\|\Delta\boldsymbol{\theta}(k) - \Delta\eta(k)\mathbf{1}\|) \quad (12)
\end{aligned}$$

To prove $\mathbb{E}(\|\boldsymbol{\zeta}(k) - \eta(k)\mathbf{1}(k)\|_\infty) \to 0$ when $k \to \infty$, we first need to compute an upper bound on the one-step GM estimation error $\mathbb{E}(\|\boldsymbol{V}(k-1)\Delta\boldsymbol{\theta}(k-1) - \Delta\eta(k-1)\mathbf{1}\|)$.

Let $e_{k-1} = \boldsymbol{V}(k-1)\Delta\boldsymbol{\theta}(k-1) - \Delta\eta(k-1)\mathbf{1}$. Using Jensen's inequality, we get the following.

$$\begin{aligned}
&\mathbb{E}(\|e_{k-1}\|) \leq \sqrt{\mathbb{E}\big(\|e_{k-1}\|^2\big)} \\
&= \sqrt{\sum_{\tau=-\infty}^{\infty}\sum_{\mathtt{i},\mathtt{j}=1}^{|\mathcal{N}|}\tau^T\mathcal{V}_{\mathtt{ij}}^T\mathcal{V}_{\mathtt{ij}}\tau\frac{1}{|\mathcal{N}|}W_{\mathtt{ij}}} \cdot \\
&\sqrt{p[(\Delta\boldsymbol{\theta}(k-1) - \Delta\eta(k-1)\mathbf{1}) = \tau]} = \\
&\sqrt{\sum_{\tau=-\infty}^{\infty}\tau^T\boldsymbol{V}^T\boldsymbol{V}\tau p[(\Delta\boldsymbol{\theta}(k-1) - \Delta\eta(k-1)\mathbf{1}) = \tau]}.
\end{aligned} \quad (13)$$

where $p[(\Delta\boldsymbol{\theta}(k-1) - \Delta\eta(k-1)\mathbf{1}) = \tau]$ denotes the probability of $(\Delta\boldsymbol{\theta}(k-1) - \Delta\eta(k-1)\mathbf{1}) = \tau$. By applying Proposition 1 to $\tau^T\boldsymbol{V}^T\boldsymbol{V}\tau$, we can write (13) as

$$\begin{aligned}
&\mathbb{E}(\|\boldsymbol{V}(k-1)\Delta\boldsymbol{\theta}(k-1) - \Delta\eta(k-1)\mathbf{1}\|) \\
&\leq \lambda(V)\sqrt{\mathbb{E}\big(\|\Delta\boldsymbol{\theta}(k-1)\|^2\big)}
\end{aligned} \quad (14)$$

We assume that in MDP in discrete state space and discrete time domain, the following difference is bounded.

$$\sqrt{\mathbb{E}\|\Delta\boldsymbol{\theta}(k)\|^2} = \sqrt{\mathbb{E}}\Big(\|\boldsymbol{\theta}(k) - \boldsymbol{\theta}(k-1)\|^2\Big) \leq \theta_{\max}. \quad (15)$$

By using (15), (14), and Proposition 1, (12) becomes:

$$\begin{aligned}
\mathbb{E}(\|\boldsymbol{\zeta}(k) - \eta(k)\mathbf{1}(k)\|_\infty) &\leq \mathbb{E}\ \ (\|\boldsymbol{\zeta}(k) - \eta(k)\mathbf{1}(k)\|) \\
&\leq \lambda^k(\boldsymbol{V})\sqrt{N}\zeta_{\max}\ \ + \lambda^{k-1}(\boldsymbol{V}) + \cdots + 1 \\
&\leq \lambda^k(\boldsymbol{V})\sqrt{N}\zeta_{\max}\ \ + \sum_{k'=1}^{k}\lambda^{k-k'}(\boldsymbol{V})\theta_{\max}.
\end{aligned} \quad (16)$$

From (16) and the fact $\lambda(\boldsymbol{V}) < 1$, we conclude that the upper bound on the estimation error goes to 0 when $k \to \infty$. We denote this upper bound by $\rho(k)$.

$$\rho(k) = \lambda^k(\boldsymbol{V})\sqrt{N}\zeta_{\max} + \sum_{k'=1}^{k}\lambda^{k-k'}(\boldsymbol{V})\theta_{\max} \quad (17)$$

For instance, $\theta_{\max}$ in a discrete MDP is 1, as each agent can change its state s by at most 1 unit per time step. Bound (17) shows that estimation errors decrease exponentially with rate $\lambda(V)$, which can be optimized by solving:

$$\begin{aligned}
\underset{V,g}{\text{minimize}}\quad & g \\
\text{subjectto}\quad & V = \frac{1}{N}\sum_{\mathtt{i},\mathtt{j}} W_{\mathtt{ij}}\mathcal{V}_{\mathtt{ij}},\ W_{\mathtt{ij}} = 0 \ \text{ if } (\mathtt{i},\mathtt{j}) \notin \mathfrak{E} \\
& V - \frac{1}{N}\mathbf{1}\mathbf{1}^T \preceq gI,\ g \in \mathbb{R} \\
& \sum_{\mathtt{j}} W_{\mathtt{ij}} = 1 \quad \forall \mathtt{i},\ W_{\mathtt{ij}} \geq 0
\end{aligned} \quad (18)$$

where $W_{\mathtt{ij}}$ is the communication probability and $\mathcal{V}_{\mathtt{ij}}$ is the communication matrix. The gossip algorithm cost is $O(\log(|\mathcal{N}|))$ [13], while broadcasting may require $O(|\mathcal{N}|^2)$ [15]. This analysis guarantees that as $k \to \infty$, all agents' GM estimates converge to the actual swarm

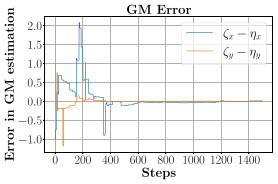


Fig. 2. Convergence of swarm GM estimation for Example 1 (ζ_x and ζ_y, the swarm GM in a two-dimensional space).

GM, enabling accurate reward machine state estimation (see Figure 2).

In the optimization problem (18), $\boldsymbol{V} - \frac{1}{|\mathcal{N}|}\mathbf{1}\mathbf{1}^T \preceq gI_{|\mathcal{N}|}$ means that $(gI_{|\mathcal{N}|} - \boldsymbol{V} + \frac{1}{|\mathcal{N}|}\mathbf{1}\mathbf{1}^T)$ is semi-definite.

V. Automata Learning for RM Learning

SwaRM-L uses counterexamples (inconsistent traces of learned and ground truth RMs) to iteratively construct a minimal RM [16], while ensuring consistency with $\mathcal{X} \subset 2^{\mathcal{P}}$. It employs SAT-based automata learning (using satisfiability solvers to learn a DFA [9]) to verify parametric systems, breaking the task into satisfiability checks for propositional logic formulas $\phi_n^{\mathcal{X}}$, where $n > 0$ [9].

Lemma 1: We denote the labeled swarm MDP process by $\mathcal{M}$, the ground truth RM $\mathcal{A}$, and the learned RM $\mathcal{U}$ encoding the rewards of labeled MDP $\mathcal{M}$, let

$$\Gamma = \max\left\{2|\mathcal{M}| \cdot (|\mathcal{A}| + 1),\ |\mathcal{M}|(|\mathcal{A}| + 1)^2\right\} \tag{19}$$

Then with episode length $\mathcal{Q} \geq \Gamma$ *almost surely* (meaning with probability 1) SwaRM-L learns an RM that is equivalent to the ground truth RM.

Proof: The algorithm maintains a set of counterexamples, which are used to learn the RM $\mathcal{U}$. By using the counterexamples $\mathcal{X}$, we will iteratively learn an equivalent RM $\mathcal{U}$ to the ground truth RM $\mathcal{A}$ (Algorithm 1 Line 17), conditioned on that all (s, a, l) are admissible [17] and encountered in $\mathcal{M}$. The algorithm visits all attainable trajectories [16] in the labeled swarm MDP $\mathcal{M}$ in the limit (no more $\mathcal{X}$ is found), meaning the set $\mathcal{X}$ becomes stable, i.e., no additional counterexamples are introduced or when the ground truth RM, $\mathcal{A}$, is identified. Once $\mathcal{X}$ stabilizes, following a similar reasoning to Neider et al. [16], the algorithm eventually learns the true RM. As long as the current $\mathcal{U}$ differs from $\mathcal{A}$, there exists a "short" trajectory revealing this. ■

Once $\mathcal{X}$ stabilizes, as in [16], the algorithm learns the correct RM. If $\mathcal{U}$ differs from the true RM, a short distinguishing trajectory exists and is added to $\mathcal{X}$ for $\mathcal{Q} \geq \Gamma$, ensuring convergence to the ground truth RM.

VI. Learning Swarm Reward Machine in RL

Using swarm GM, we learn an RM equivalent to the ground truth RM, enabling each agent $\mathtt{i}$ to estimate its RM state $\hat{v}_k^{\mathtt{i}}$ from its estimated GM $\zeta^{\mathtt{i}}$. This GM guides RM state transitions, while the ground truth RM state $v_k^{\mathtt{i}}$ is not known. Each agent relies on $\zeta^{\mathtt{i}}$ for transitions, with $\boldsymbol{\zeta}$ converging to the actual GM $\eta\mathbf{1}$. An optimal policy from SwaRM-L is a function of the RM state, $\pi^{\mathtt{i}}(s^{\mathtt{i}}, \hat{v}^{\mathtt{i}})$, and maximizes the expected cumulative reward defined by the RM, which represents the swarm-level task. The non-Markovian reward structure becomes Markovian by using RM through the augmented state (s, v). Swarm-level learning refines the RM using counterexamples $\mathcal{X}$ from swarm trajectories. Transitions occur when the swarm GM satisfies (20), $\hat{v}_k^{\mathtt{i}} \neq \hat{v}_{k+1}^{\mathtt{i}}$ if within a user-defined threshold.

Theorem 1: The estimated learned RM states converge to the actual learned RM states when the estimated swarm GMs are sufficiently close to the actual swarm GMs, i.e., the constraint (20) is satisfied.

$$\hat{v}_{k+1}^{\mathtt{i}} = \delta\big(\hat{v}_k^{\mathtt{i}}, L(\zeta^{\mathtt{i}}(k))\big) \tag{20}$$

Proof: We show that as estimated swarm GMs approach the true GM, the estimated RM states $\hat{v}_k^{\mathtt{i}}$ converge to the actual RM state $v_k = \hat{v}_k^{\mathtt{i}}\ \forall\ \mathtt{i} \in \mathcal{N}$ for all agents as $k \to \infty$ by enforcing the transition constraint (21).

$$\begin{cases} \delta(v_k, L(\eta(k))) = v_{k+1} & \text{iff } \|\eta(k) - \kappa\| < \mu \\ v_k = v_{k+1} & \text{else} \end{cases} \tag{21}$$

where κ is the target state and $\mu \in \mathbb{R}$ is a user-defined threshold. RM transitions depend on each agent's estimated state $\hat{v}^{\mathtt{i}}$. To ensure $v_k = \hat{v}_k^{\mathtt{i}}$ for all $\mathtt{i}$ as $k \to \infty$, the sequence $\hat{v}_k^{\mathtt{i}}, \ldots, \hat{v}_{k+n}^{\mathtt{i}}$ must follow the learned RM transitions $v_k, \ldots, v_{k+n}$. By the triangle inequality and constraint (21):

$$\delta\big(\hat{v}_k^{\mathtt{i}}, L(\zeta^{\mathtt{i}}(k))\big) = \hat{v}_{k+1}^{\mathtt{i}} \text{ iff } \|\zeta^{\mathtt{i}}(k) - \kappa\| < \mu - \xi \tag{22}$$

where $\xi \in \mathbb{R}$ is a user-defined upper bound on the GM estimation error from (17), i.e., $\|\boldsymbol{\zeta}(k) - \eta\mathbf{1}\|$. The constraint (22) triggers a transition only if the agent's estimated GM is within $\mu - \xi > 0$ of the target (*close enough*), while ξ ensures GM estimates are within the acceptable error bound to the actual GM η since ξ evolution follows (17). ■

Corollary 1: Let $\mathtt{T}$ be the communication interval (gossip occurs every $\mathtt{T}$ time steps). The time steps k for $\boldsymbol{\zeta}$ to converge to within ε of the $\eta\mathbf{1}$ satisfy $k = \mathtt{T} \cdot \log(1/\varepsilon)$, swarm GM errors remain acceptable for $\mathtt{T} < \mathtt{T}_{\max} = \lfloor(\mu - \xi)/\theta_{\max}\rfloor$, i.e., swarm GM estimation errors remain bounded.

Proof: Let GM error bound $\rho(k) < \xi < \mu/2$ and $k > \mathtt{T}C\log(1/\xi)$ (from (17)), with $C = 1/|\log(\lambda(V))|$. Assuming $\varepsilon \propto \xi$, we get $k = \mathtt{T}\log(1/\varepsilon)$. Stability holds for $\mathtt{T} < \mathtt{T}_{\max}$, ensuring RM errors stay below $\mu/2$. ■

Theorem 2: The SwaRM-L algorithm converges to an optimal policy when the agents' reward machine state transition is governed by constraint (20).

Proof: Theorem 2 follows immediately from Theorem 1, Lemma 1, Proposition 1, and correctness of the QRM [3]. ■

Algorithm 1 initializes the learned RM $\mathcal{U}$, agent q-values, MDP states, counterexample set $\mathcal{X}$, and each agent's estimated GM (Lines 1-2). Over M episodes, agents select ϵ-greedy actions and update states (Lines 5-8). Gossip-based communication updates GMs and RM states for active agents (Lines 9-10); others update via (9). The environment issues rewards, and agents update q-values (Lines 13-14). Counterexamples are added if $\mathcal{U}$ disagrees with the ground truth RM (Lines 15-16). We use automata learning and satisfiability checking [9] (Section V) to iteratively learn

Algorithm 1: SwaRM-L

Hyperparameters: M episodes, steps $\mathcal{Q}$,
Discount factor γ, Error bound ξ, Threshold μ
Input: Learned Reward Machine $\mathcal{U}$, q-function q

1 $\mathcal{U} \leftarrow \emptyset$, $q^{\mathtt{i}} \leftarrow$ InitQFun(), $\mathtt{i} \in \mathcal{N}$, $\mathcal{X} \leftarrow \emptyset$;
2 $s \leftarrow$ InitState(), $\hat{v}^{\mathtt{i}} \leftarrow 0$, $\mathtt{i} \in \mathcal{N}$, $\zeta^{\mathtt{i}} \leftarrow 0$, $\mathtt{i} \in \mathcal{N}$;
3 **for** $1 \leq episode \leq M$ **do**
4 $\mathcal{X}_{\text{init}} \leftarrow \mathcal{X}$;
5 **for** $0 \leq k < \mathcal{Q}$ **do**
6 **for** *each agent* $\mathtt{i} \in \{1, \ldots, N\}$ **do**
7 $a_k^{\mathtt{i}} \leftarrow$ SelectAction$\big(\pi^{\mathtt{i}}(a_k^{\mathtt{i}} | s_k^{\mathtt{i}}, \hat{v}_k^{\mathtt{i}}), \epsilon\big)$;
8 $s_{k+1}^{\mathtt{i}} \leftarrow$ ExecuteAction$\big(s_k^{\mathtt{i}}, a_k^{\mathtt{i}}\big)$;
9 **if** *agent* $\mathtt{i}$ *is active* **then**
10 $(\zeta_{k+1}^{\mathtt{i}}, \hat{v}_{k+1}^{\mathtt{i}}, \zeta_{k+1}^{\mathtt{j}}, \hat{v}_{k+1}^{\mathtt{j}}) \leftarrow$ UpdateGM$\big(i, \mathcal{N}_{\mathtt{i}}, \zeta_k^{\mathtt{i}}, \hat{v}_k^{\mathtt{i}}\big)$;
11 **else**
12 Update $(\zeta_{k+1}^{\mathtt{i}}, \hat{v}_{k+1}^{\mathtt{i}})$ using Equation (9);
13 $r_k \leftarrow \sigma(v_k, l_k)$, $\rho \leftarrow \rho \cup \{r_k\}$;
14 Update q-values $q^{\mathtt{i}}(s_k^{\mathtt{i}}, a_k^{\mathtt{i}}, \hat{v}_k^{\mathtt{i}})$ for all agents;
15 **if** $\mathcal{U}(\lambda) \neq \rho$ **then**
16 $\mathcal{X} \leftarrow \mathcal{X} \cup \{(\lambda, \rho)\}$;
17 **if** $\mathcal{X} \neq \mathcal{X}_{\text{init}}$ **then**
18 $\mathcal{U} \leftarrow$ LearnRM$(\mathcal{X})$, $q \leftarrow$ InitQFun();
19 **return** (q, ρ, λ);

Algorithm 2: UpdateGM

Input: Agents $\mathtt{i}$, Neighbors $\mathcal{N}_{\mathtt{i}}$, estimated GMs $\{\zeta_k^{\mathtt{i}}\}$
Output: $\zeta_{k+1}^{\mathtt{i}}$, $\hat{v}_{k+1}^{\mathtt{i}}$, $\zeta_{k+1}^{\mathtt{j}}$, $\hat{v}_{k+1}^{\mathtt{j}}$

1 $\mathtt{j} \leftarrow$ SelectNeighbor$(\mathtt{i}, \mathcal{N}_{\mathtt{i}})$;
2 $\zeta_{k+1}^{\mathtt{i}} \leftarrow$ using Equation (7);
3 $\zeta_{k+1}^{\mathtt{j}} \leftarrow$ using Equation (8);
4 **for** $\mathtt{K} \in \{\mathtt{i}, \mathtt{j}\}$ **do**
5 **if** $\|\zeta_{k+1}^{\mathtt{K}} - \kappa\| < \mu - \xi$ **then**
6 $\hat{v}_{k+1}^{\mathtt{K}} \leftarrow \delta(\hat{v}_k^{\mathtt{K}}, L(\zeta_{k+1}^{\mathtt{K}}))$;
7 **else**
8 $\hat{v}_{k+1}^{\mathtt{K}} \leftarrow \hat{v}_k^{\mathtt{K}}$;
9 **return** $\zeta_{k+1}^{\mathtt{K}}, \hat{v}_{k+1}^{\mathtt{K}}$;

an equivalent RM which eventually becomes equivalent to the ground truth RM by using SAT solvers [16], i.e., Line 18 (see Lemma 1). Updated $\mathcal{X}$ trigger RM relearning for consistency and q-function reinitialization to avoid bias (Lines 17-18).

Algorithm 2 updates the estimated swarm GMs and RM states for the communicating agents. The active agent selects a neighbor (Line 1), both update their GMs via gossip (Lines 2-3), and then update their RM states if the GM estimate is within the error bound (Lines 5-8). The RM state updates occur only when the GM estimation error is below ξ.

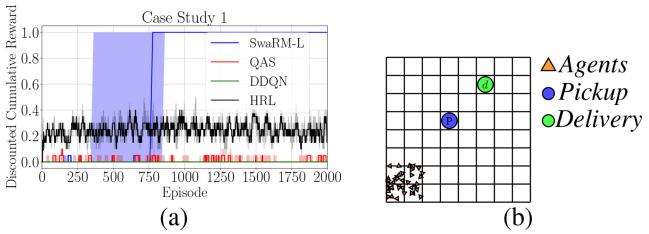


Fig. 3. (a) SwaRM-L uses task's temporal structure to learn an optimal policy. (b) Case Study 1: forty-agent swarm, with p (pickup) and d (delivery) labels encoding the task.

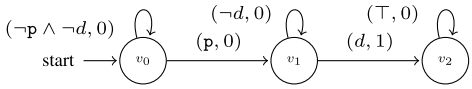


Fig. 4. Case Study 1 ground truth RM, label $\mathtt{p}$ shows pickup location reached (verified by sufficient agents via first-order swarm GM, all transitions are governed by swarm GM), and d signals the delivery location is reached.

VII. Case Studies

The swarm learns the temporal structure and ground truth RM under sparse rewards (only given when all sub-tasks are completed in order). The q-learning in augmented state space (QAS) uses label information as one-hot vectors in the state space [17]. The double deep q-network (DDQN) [18] learns policies via state-action pairs, assigning q-values per agent. The hierarchical RL (HRL) [19] uses a high-level controller and low-level policies to decompose tasks into subtasks (a similar strategy to SwaRM-L), learning a policy for each.

Case Study 1. A swarm of forty agents has no knowledge of the underlying task structure and learns the task structure by interacting with the environment.

The ground truth RM for this task is shown in Figure 4. Figure 3(a) shows the discounted cumulative reward averaged every 30 episodes (500 steps per episode) over five runs. SwaRM-L achieves higher sampling efficiency and converges to an optimal policy, while QAS and HRL obtain higher rewards but fail to learn the temporal structure; DDQN struggles to learn effective policies. SwaRM-L scales efficiently by using swarm GM to estimate RM state and transitions.

Case Study 2. A swarm of forty agents performs search and rescue in a 12 × 12 grid, maintaining spatial distribution (first and second-order swarm GMs) for efficient search, then transporting survivors to safe zones while avoiding hazards. The RM for this task is shown in Figure 5.

Figure 6(a) shows discounted cumulative reward (averaged every 50 episodes, 500 steps per episode). SwaRM-L converges faster than baselines in a more complex scenario by using swarm GMs to learn the task's temporal structure.

Case Study 3. Extends search and rescue by adding a decontamination sub-task, toxic areas, and expanding the grid to 15 × 15 with 60 agents (RM in Figure 7).

Figure 8(a) shows the reward averaged over 50 episodes (900 steps per episode). SwaRM-L converges to an optimal policy, but baselines struggle with learning the increased complexity of the underlying temporal structure of the task.

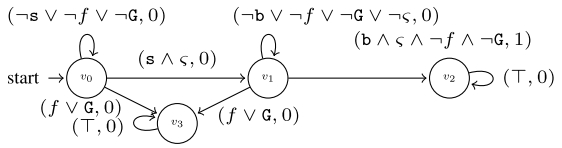
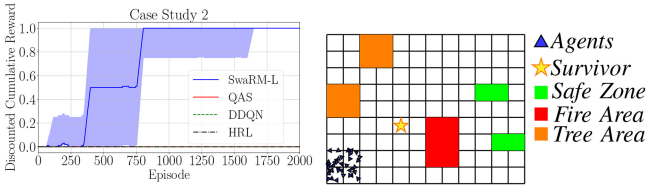


Fig. 5. Case Study 2 RM: $\mathtt{b}$ safe zone, f fire, $\mathtt{G}$ trees, $\mathtt{s}$ survivors, and ς swarm position variance (second-order GM).



(a) (b)

Fig. 6. (a) SwaRM-L converges to an optimal policy; while the baselines fail to learn the temporal structure. (b) Case Study 2: Swarm performs search and rescue, taking survivors to safety while avoiding hazards.

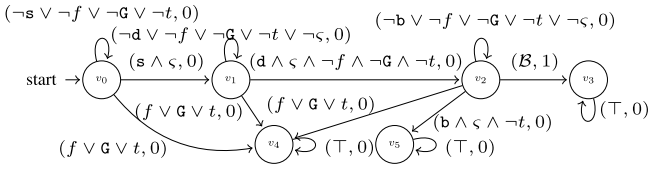
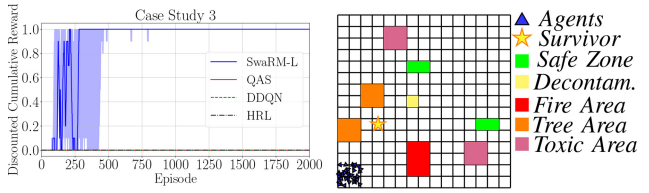


Fig. 7. The ground truth RM in Case Study 3 uses labels for toxic areas t, and decontamination $\mathtt{d}$ to encode additional sub-tasks ($\mathcal{B} = \mathtt{b} \wedge \varsigma \wedge \neg f \wedge \neg \mathtt{G} \wedge \neg\ t$).



(a) (b)

Fig. 8. (a) SwaRM-L completes the task. (b) Case Study 3: with survivors to be decontaminated first.

VIII. Conclusion

SwaRM-L enables swarm systems to learn RMs by using GMs. We showed SwaRM-L learns an equivalent RM to the ground truth RM within the environment. Future work extends SwaRM-L to heterogeneous swarms and communication-constrained settings and methods, e.g., broadcasting.

References

[1] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[2] F. Djeumou, Z. Xu, M. Cubuktepe, and U. Topcu, "Probabilistic control of heterogeneous swarms subject to graph temporal logic specifications: A decentralized and scalable approach," *IEEE Trans. Autom. Control*, vol. 68, no. 4, pp. 2245–2260, Apr. 2023.

[3] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using reward machines for high-level task specification and decomposition in reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 2107–2116.

[4] S. M. Alsadat, N. Baharisangari, Y. Paliwal, and Z. Xu, "Distributed reinforcement learning for swarm systems with reward machines," in *Proc. Am. Control Conf. (ACC)*, 2024, pp. 33–38.

[5] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 5571–5580.

[6] M. Hüttenrauch, A. Šošić, and G. Neumann, "Deep reinforcement learning for swarm systems," *J. Mach. Learn. Res.*, vol. 20, no. 54, pp. 1–31, 2019.

[7] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning," 2017, *arXiv:1706.05296*.

[8] C. Neary, Z. Xu, B. Wu, and U. Topcu, "Reward machines for cooperative multi-agent reinforcement learning," 2020, *arXiv:2007.01962*.

[9] D. Neider, "Applications of automata learning in verification and synthesis," Ph.D. dissertation, Faculty Math., Comput. Sci. Nat. Sci., RWTH Aachen Univ., Aachen, Germany, 2014.

[10] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 1–12.

[11] R. Yan, Z. Xu, and A. Julius, "Swarm signal temporal logic inference for swarm behavior analysis," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 3021–3028, Jul. 2019.

[12] Z. Zhang, X. Zhao, B. Tao, and H. Ding, "Distributed gossip-triggered control for robot swarms with limited communication range," *IEEE Trans. Ind. Electron.*, vol. 70, no. 12, pp. 12511–12521, Dec. 2023.

[13] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2508–2530, Jun. 2006.

[14] R. Yan and A. Julius, "Distributed consensus-based online monitoring of robot swarms with temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 9413–9420, Oct. 2022.

[15] T. Locher, "Byzantine reliable broadcast with low communication and time complexity," 2024, *arXiv:2404.08070*.

[16] D. Neider, J.-R. Gaglione, I. Gavran, U. Topcu, B. Wu, and Z. Xu, "Advice-guided reinforcement learning in a non-Markovian environment," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 9073–9080.

[17] Z. Xu et al., "Joint inference of reward machines and policies for reinforcement learning," in *Proc. Int. Conf. Autom. Plan. Schedul. (ICAPS)*, 2020, pp. 590–598.

[18] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[19] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3682–3690.