Neurocomputing 564 (2024) 126974

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Decentralized graph-based multi-agent reinforcement learning using reward machines

Jueming Hu [a], Zhe Xu [a], Weichang Wang [b], Guannan Qu [c], Yutian Pang [a], Yongming Liu [a,*]

[a] School for Engineering of Matter, Transport and Energy, Arizona State University, Tempe, AZ, United States of America
[b] School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, United States of America
[c] Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, United States of America

ARTICLE INFO

Communicated by K.G. Vamvoudakis

Keywords:
Decentralized
Multi-agent
Reinforcement learning
Reward machine
Efficiency

ABSTRACT

In multi-agent reinforcement learning (MARL), it is challenging for a collection of agents to learn complex temporally extended tasks. The difficulties lie in computational complexity and how to learn the high-level ideas behind reward functions. We study the graph-based Markov Decision Process (MDP), where the dynamics of neighboring agents are coupled. To learn complex temporally extended tasks, we use a reward machine (RM) to encode each agent's task and expose reward function internal structures. RM has the capacity to describe high-level knowledge and encode non-Markovian reward functions. We propose a decentralized learning algorithm to tackle computational complexity, called decentralized graph-based reinforcement learning using reward machines (DGRM), that equips each agent with a localized policy, allowing agents to make decisions independently based on the information available to the agents. DGRM uses the actor-critic structure, and we introduce the tabular Q-function for discrete state problems. We show that the dependency of the Q-function on other agents decreases exponentially as the distance between them increases. To further improve efficiency, we also propose the deep DGRM algorithm, using deep neural networks to approximate the Q-function and policy function to solve large-scale or continuous state problems. The effectiveness of the proposed DGRM algorithm is evaluated by three case studies, two wireless communication case studies with independent and dependent reward functions, respectively, and COVID-19 pandemic mitigation. Experimental results show that local information is sufficient for DGRM and agents can accomplish complex tasks with the help of RM. DGRM improves the global accumulated reward by 119% compared to the baseline in the case of COVID-19 pandemic mitigation.

1. Introduction

In multi-agent reinforcement learning (MARL), a collection of agents interact within a common environment and learn to maximize a long-term reward jointly. We study MARL in a graph-based Markov Decision Process (MDP) setting, where for each agent, the transition model is represented by a dynamic Bayesian network [1,2]. In graph-based MDPs, the transition function between an agent's states may depend on its current state and action and the neighboring agents [3,4]. In this work, the agents can have different reward functions from different tasks, and they can perceive their own rewards.

The key challenge in MARL is the combinatorial nature, which results in high computational complexity. The combinatorial nature refers to the exponentially increased size of the joint state and action space in the multi-agent system [5,6]. Since all agents are learning simultaneously, the behavior of each individual agent relies on what has been learned by the respective other agents. Thus, the learning problem is non-stationary from each agent's perspective. To tackle non-stationarity, each individual agent has to consider the joint state and action space, whose dimension increases exponentially with the number of agents.

RL becomes more challenging when solving complex temporally extended tasks. In many complex tasks, agents only receive sparse rewards for complex behaviors over a long time. However, agents do not have access to the high-level ideas behind the sparse rewards. Moreover, in a sparse-reward formulation, if it is improbable that the agent achieves the task by chance when exploring the environment, the learning agent will rarely obtain rewards.

In this paper, we provide a decentralized framework that enables a collection of agents to solve complex temporally extended tasks under

* Corresponding author.
E-mail addresses: jueming.hu@asu.edu (J. Hu), xzhe1@asu.edu (Z. Xu), wwang195@asu.edu (W. Wang), gqu@andrew.cmu.edu (G. Qu), yutian.pang@asu.edu (Y. Pang), yongming.liu@asu.edu (Y. Liu).

https://doi.org/10.1016/j.neucom.2023.126974
Received 16 June 2022; Received in revised form 15 March 2023; Accepted 26 October 2023
Available online 30 October 2023

coupled dynamics with enhanced computational efficiency. Specifically, we use reward machines (RMs) to describe the environment, track the progress through the task, and encode a sparse or non-Markovian reward function for each agent. The proposed algorithm, called decentralized graph-based reinforcement learning using reward machines (DGRM), uses the actor-critic structure, specifically adopting temporal difference (TD) learning to train the truncated Q-function and policy gradient for the localized policy. To tackle computational efficiency, we propose a truncated Q-function considering only the information of the κ-hop neighborhood. The RM state is augmented with the low-level MDP state as the input for the truncated Q-function and the policy function. Each agent is equipped with a localized policy, and information exchange among agents when implementing the learned policies is not required. We provide the proof that the truncated Q-function can approximate the Q-function with high accuracy, and the error of policy gradient approximation is bounded. Also, DGRM can find an $O(\gamma^{\kappa+1})$-approximation of a stationary point of the objective function, where γ is the discount factor. To further improve efficiency, we develop deep DGRM which uses deep neural networks for function approximation to improve scalability. We demonstrate the effectiveness of DGRM and deep DGRM on a discrete state problem of wireless communication and a continuous state problem of COVID-19 pandemic mitigation, respectively. In the experiments, agents can accomplish complex tasks under the guidance of RMs.

2. Related work

Recent years have witnessed rapid development in MARL [7–10]. AnMARL system can be categorized into centralized and decentralized control. The central controller decides the actions for all agents [11,12], while within the decentralized control, agents make their own decisions based on their observations [12,13]. The central controller needs to collect information of all agents and communicate with all agents, which mitigates the non-stationarity but may decrease the scalability of the multi-agent system. Therefore, we consider a decentralized learning algorithm to solve typically graph-based MDP, where the dynamics of agents are coupled. Graph-based MDP can also be referred to as networked multi-agent MDP [14], factored MDP [2,15,16].

The most relevant studies to our proposed DGRM algorithm are the methods utilizing the actor-critic structure for MARL. Foerster et al. [17],Lowe et al. [18] develop decentralized actor-centralized critic models for MARL. The centralized critic has access to the joint action and all available state information. The decentralized actor considers the local information of each agent. Additionally, the proposed algorithms in Foerster et al. [17],Lowe et al. [18] adopt the framework of centralized training with decentralized execution. After training is completed, only the local actors are used at the execution phase. The communication requirement in our proposed DGRM algorithm is weaker than that in Foerster et al. [17],Lowe et al. [18], where DGRM uses information of κ-hop neighborhood to learn the critics during training, while global information is required to learn the critics in Foerster et al. [17],Lowe et al. [18]. A decentralized actor-critic algorithm with provable convergence guarantees is proposed in Zhang et al. [14]. They use a linear approximation for the Q-function, but it is unclear whether the loss caused by the function approximation is small. The proposed DGRM algorithm is inspired by the Scalable Actor-Critic (SAC) algorithm in Qu et al. [19], and we further propose the DGRM algorithm to solve complex temporally extended tasks and the deep DGRM algorithm for large-scale or continuous state problems.

Recently, reward machines (RMs) have received much attention for task specification. Icarte et al. [20] propose reward machines, i.e., a type of Mealy machine, to encode structures or high-level knowledge behind the tasks. They develop the Q-learning for Reward Machines (QRM) algorithm in the single-agent setting and show that QRM can converge to an optimal policy in the tabular case. Later, they extend their work and develop counterfactual experiences for reward machines (CRM) and hierarchical RL for reward machines (HRM).

In the multi-agent setting, [21] use RMs to describe cooperative tasks and introduce how to decompose the cooperative task into a collection of new RMs, each encoding a sub-task for an individual agent. They assume agents only observe their local state and abstracted representations of their teammates [21]. Agents in our work have access to the information of neighboring agents since we consider the graph-based MDP, where the dynamics of neighboring agents are coupled. Thus, the dimension of the Q-function in our work is larger than that in Neary et al. [21] and needs complexity reduction for better scalability. RMs have also been applied to robotics, such as quadruped locomotion learning [22] and planning [23]. There are also studies focusing on learning RMs from experience instead of giving as a priori to the learning agent [24–29]. Hierarchical reinforcement learning (HRL) is another classical approach to solve complex tasks, which decomposes an RL problem into a hierarchy of subtasks, including methods such as HAMs [30], Options [31], and MAXQ [32]. However, these HRL approaches cannot guarantee convergence to the optimal policy.

3. Preliminaries

3.1. Markov decision processes and reward machines

A labeled MDP is defined by a tuple, $\mathcal{M} = (S, s_I, A, R, P, \gamma, \mathcal{P}, L)$, where S is a finite set of states; $s_I \in S$ is an initial state; A is a finite set of actions; $R : S \times A \to \mathbb{R}$ is the reward function, $R(s,a)$ represents the reward obtained at state x after taking action a; $P : S{\times}A{\times}S \to [0,1]$ is the transition function, $P(s,a,s')$ represents the probability of going to s' from s after taking action a; γ is the discount factor, $\gamma \in (0,1]$; $\mathcal{P}$ is a finite set of events; and $L : S \times A \times S \to \mathcal{P}$ is a labeling function, $L(s,a,s')$ maps the state transition to a relevant high-level event.

Definition 1. A reward machine (RM) *is a tuple,* $\mathcal{R} = (U, u_I, \mathcal{P}, \delta, \sigma)$, *where* U *is a finite set of RM states,* $u_I \in U$ *is an initial state,* $\mathcal{P}$ *is a finite set of events,* δ *is the transition function:* $U \times \mathcal{P} \to U$, and σ is a reward function: $U \times \mathcal{P} \to \mathbb{R}$.

Reward machines are a way to encode a non-Markovian reward function [33]. A run of a reward machine $\mathcal{R}$ on the sequence of labels $\ell_1\ell_2\dots\ell_k \in \mathcal{P}^*$ is a sequence $u_0(\ell_1,r_1)u_1(\ell_2,r_2)\dots u_{k-1}(\ell_k,r_k)u_k$ of states and label-reward pairs such that $u_0 = u_I$. For all $i \in \{0,\dots,k-1\}$, we have $\delta(u_i,l_{i+1}) = u_{i+1}$ and $\sigma(u_i,l_{i+1}) = r_{i+1}$. We write $\mathcal{R}(\ell_1\ell_2\dots\ell_k) = r_1r_2\dots r_k$ to connect the input label sequence to the sequence of rewards produced by the machine $\mathcal{R}$. We say that a reward machine $\mathcal{R}$ encodes the reward function R of an MDP if for every trajectory $s_0a_1s_1\dots a_ks_k$ and the corresponding label sequence $\ell_1\ell_2\dots\ell_k$, the reward sequence equals $\mathcal{R}(\ell_1\ell_2\dots\ell_k)$.

Given a labeled MDP $\mathcal{M} = (S, s_I, A, R, P, \gamma, \mathcal{P}, L)$ with a non-Markovian reward function and a reward machine $\mathcal{R} = (U, u_I, \mathcal{P}, \delta, \sigma)$ which encodes the reward function R of $\mathcal{M}$, we can obtain a product MDP $\mathcal{M}_\mathcal{R}$, whose reward function is Markovian such that every attainable label sequence of $\mathcal{M}_\mathcal{R}$ gets the same reward as in $\mathcal{M}$. Furthermore, any policy for $\mathcal{M}_\mathcal{R}$ achieves the same expected reward in $\mathcal{M}$ [24]. We define the product MDP $\mathcal{M}_\mathcal{R} = (S', s'_I, A, R', P', \gamma', \mathcal{P}', L')$ by:

$$\begin{aligned}
S' &= S \times U \\
s'_I &= (s_I, u_I) \\
A &= A \\
R'(s,u,a) &= \sigma(u, L(s,a,s')) \\
P'(s,u,a,s',u') &= \begin{cases} P(s,a,s') & u' = \delta(u, L(s,a,s')) \\ 0 & \text{otherwise} \end{cases} \\
\gamma' &= \gamma; \mathcal{P}' = \mathcal{P}; L' = L
\end{aligned} \tag{1}$$

3.2. Graph-based multi-agent MDP with reward machines

We denote $G=(V,E)$ as an undirected graph, where $V=\{1,\dots,n\}$ is a finite set of nodes that represents the agents, and E is a finite set of edges. For agents i,j, we call j a neighboring agent of i, if there is an edge that connects i and j. Let $N(i)\subseteq V$ be the set including agent i and the neighboring agents of the agent i.

We consider a multi-agent system, including n agents and formulate the system as a labeled graph-based MDP $\mathcal{M}_G=(S_G,s_{I_G},A_G,R_G,P_G,\gamma,\mathcal{P}_G,L_G)=(\{S_i\}_{i=1}^n,\{s_I^i\}_{i=1}^n,\{A_i\}_{i=1}^n,\{R_i\}_{i=1}^n,\{P_i\}_{i=1}^n,\gamma,\{\mathcal{P}_i\}_{i=1}^n,\{L_i\}_{i=1}^n)$. We consider a factored form for $\mathcal{M}_G$, i.e, S_G is a Cartesian product of the states for each agent i in V, $S_G=S_1\times\cdots\times S_n$. Similarly, $s_{I_G}=s_I^1\times\cdots\times s_I^n$; $A_G=A_1\times\cdots\times A_n$; $R_G=R_1\times\cdots\times R_n$; $P_G=P_1\times\cdots\times P_n$; $\mathcal{P}_G=\mathcal{P}_1\times\cdots\times\mathcal{P}_n$; and $L_G=L_1\times\cdots\times L_n$. For each agent i, $P_i: S_{N(i)}\times A_{N(i)}\times S_i\to[0,1]$, meaning $s_i(t+1)$ depends on $s_{N(i)}(t)\in S_{N(i)}$ and $a_{N(i)}(t)\in A_{N(i)}$, where $S_{N(i)}$ and $A_{N(i)}$ denote the Cartesian product of the sets of states and actions of the agents in $N(i)$ respectively. R_i and L_i only depend on the information of agent i itself, $R_i: S_i\times A_i\to\mathbb{R}$ and $L_i: S_i\times A_i\times S_i\to\mathcal{P}_i$.

We design a reward machine $\mathcal{R}_i$ for each agent i. $\mathcal{R}_i=(U_i,u_I^i,\mathcal{P}_i,\delta_i,\sigma_i)$ encodes the reward function R_i of $\mathcal{M}_G$ individually. $\mathcal{R}_i$ defines the task of agent i in terms of high-level events from $\mathcal{P}_i$. Given a labeled graph-based MDP $\mathcal{M}_G=(S_G,s_{I_G},A_G,R_G,P_G,\gamma,\mathcal{P}_G,L_G)$ with a non-Markovian reward function collectively defined by Reward Machines $\mathcal{R}_i$, we can obtain a product MDP $\mathcal{M}_{GR}=(S_{GR},s_{I_{GR}},A_{GR},R_{GR},P_{GR},\gamma_{GR},\mathcal{P}_{GR},L_{GR})$ whose reward function is Markovian,

$$\begin{aligned}
S_{GR}&=S_G\times U_1\times\cdots\times U_n\\
s_{I_{GR}}&=(s_{I_G},u_I^1\times\cdots\times u_I^n)\\
A_{GR}&=A_G\\
R_{GR_i}(s_i,u_i,a_i)&=\sigma_i(u_i,L_i(s_i,a_i,s_i'))\\
P_{GR}(s,u,a,s',u')&=P_{GR_1}\times\cdots\times P_{GR_n}\\
P_{GR_i}(s_{N(i)},u_{N(i)},a_{N(i)},s_i',u_i')&=\begin{cases}P_i(s_{N(i)},a_{N(i)},s_i') & \text{if } u_i'=\delta(u_i,L_i(s_i,a_i,s_i'))\\ 0 & \text{otherwise}\end{cases}\\
\gamma_{GR}&=\gamma;\mathcal{P}_{GR}=\mathcal{P}_G;L_{GR}=L_G
\end{aligned}\tag{2}$$

For simplicity of notation, we define the global state $s=(s_1,\dots,s_n)\in S_G$, global RM state $u=(u_1,\dots,u_n)\in U_G:=U_1\times\cdots\times U_n$, global action $a=(a_1,\dots,a_n)\in A_G$, and global reward $R(s,u,a)=\frac{1}{n}\sum_{i=1}^n R_i(s_i,u_i,a_i)$.

$\mathcal{M}_{GR}$ augments RM state and MDP state, which tracks the progress through the task for each agent. The goal in an $\mathcal{M}_{GR}$ is to find the optimal joint policy $\pi^\theta: S_{GR}\to Distr(A_{GR})$, which is parameterized by θ. $\pi^\theta(a|s,u)$ is expected to maximize the discounted global reward, $J(\theta)$ starting from the initial state distribution s_I and u_I,

$$\max_\theta J(\theta)=\mathbb{E}_{(s,u)\sim s_I,u_I}\mathbb{E}_{a(t)\sim\pi^\theta(\cdot|s(t),u(t))}[\sum_{t=0}^{\infty}\gamma^t\cdot R(s(t),u(t),a(t))|s(0)=s_I,u(0)=u_I]\tag{3}$$

The Q-function under the policy π^θ, $Q^\theta(s,u,a)$, is defined as the expected discounted future reward of taking action a given a pair (s,u) and then following the policy π^θ,

$$\begin{aligned}
&Q^\theta(s,u,a)\\
&=\mathbb{E}_{a(t)\sim\pi^\theta(\cdot|s(t),u(t))}[\sum_{t=0}^{\infty}\gamma^t R(s(t),u(t),a(t))|s(0)=s_I,u(0)=u_I,a(0)=a]\\
&=\frac{1}{n}\sum_{i=1}^n\mathbb{E}_{a(t)\sim\pi^\theta(\cdot|s(t),u(t))}[\sum_{t=0}^{\infty}\gamma^t R_i(s_i(t),u_i(t),a_i(t))|s(0)=s,u(0)=u,a(0)=a]\\
&=\frac{1}{n}\sum_{i=1}^n Q_i^\theta(s,u,a)
\end{aligned}\tag{4}$$

where $Q_i^\theta(s,u,a)$ is the Q-function for the individual reward R_i, with the dimension of $(S_1\times\cdots\times S_n)\times(U_1\times\cdots\times U_n)\times(A_1\times\cdots\times A_n)$. Compared to standard Q-learning, $Q^\theta(s,u,a)$ also keeps track of RM states, allowing agents to take the high-level stages into consideration when selecting actions.

A commonly used RL algorithm is policy gradient [34]. Let d^θ be a distribution on the S_{GR} given by $d^\theta(s,u)=(1-\gamma)\sum_{t=0}^\infty\gamma^t d_t^\theta(s,u)$, where $d_t^\theta(s,u)$ is the distribution of $(s(t),u(t))$ under fixed policy θ when $(s(0),u(0))$ is drawn from d_0. Then,

$$\nabla J(\theta)=\frac{1}{1-\gamma}\mathbb{E}_{(s,u)\sim d^\theta,a\sim\pi^\theta(\cdot|s,u)}Q^\theta(s,u,a)\nabla\log\pi^\theta(a|s,u)\tag{5}$$

Since the sizes of the joint state and action spaces, S_{GR} and A_{GR}, grow exponentially with the number of agents, the corresponding Q-functions Q^θ and Q_i^θ can be large tables and therefore, are intractable to compute and store.

4. Decentralized graph-based reinforcement learning using reward machines (DGRM)

We propose the truncated Q-function ($\tilde{Q}_i^\theta$), which takes the local information of each agent as the input, and thus, the dimension of $\tilde{Q}_i^\theta$ is much smaller than Q_i^θ. We present that $\tilde{Q}_i^\theta$ is able to approximate Q_i^θ with high accuracy. Then, we propose the truncated policy gradient based on $\tilde{Q}_i^\theta$, and the error of policy gradient approximation is bounded. Moreover, the DGRM algorithm can find an $O(\gamma^{\kappa+1})$-approximation of a stationary point of $J(\theta)$. To further improve efficiency, we also develop the deep DGRM algorithm, using deep neural networks to approximate the Q-function and policy function to solve large-scale or continuous state problems.

4.1. DGRM algorithm

To reduce complexity, we propose a decentralized learning algorithm that equips each agent with a localized policy $\pi_i^{\theta_i}$, parameterized by θ_i. The localized policy for agent i, $\pi_i^{\theta_i}(a_i|s_i,u_i)$, is a distribution on the local action a_i, depending on the local state s_i and local RM state u_i. Thus, each agent acts independently, and information exchange is not required with learned policies. Moreover, the RM state is augmented with the low-level MDP state for the policy, which provides guidance to the agent in the task level. The joint policy, $\pi^\theta(a|s,u)$, is defined as $\prod_{i=1}^n\pi_i^{\theta_i}(a_i|s_i,u_i),\theta=(\theta_1,\dots,\theta_n)$. $\pi_i^{\theta_i}$ is similar to the policy defined in the Scalable Actor-Critic (SAC) algorithm [19], which is conditioned only on the local state s_i. $\pi_i^{\theta_i}$ differs from the policy in graph-based MDP [4] because their policy is conditioned on the agent's neighborhood.

Further, DGRM algorithm uses local information of each agent i to approximate Q_i^θ, inspired by Qu et al. [19]. We assume that the agent i has access to the neighborhood of radius κ (or κ-hop neighborhood) of agent i, denoted by N_i^κ, $\kappa\geq 0$. κ-hop neighborhood of i includes the agents whose shortest graph distance to agent i is less than or equal to κ, including agent i itself. We define $N_{-i}^\kappa=V/N_i^k$, meaning the set of agents that are outside of agent i's κ-hop neighborhood. The global state s can also be written as $(s_{N_i^\kappa},s_{N_{-i}^\kappa})$, which are the states of agents that are inside and outside agent i's κ-hop neighborhood respectively, similarly, the global RM state $u=(u_{N_i^\kappa},u_{N_{-i}^\kappa})$ and the global action $a=(a_{N_i^\kappa},a_{N_{-i}^\kappa})$. First, we define the exponential decay property of the Q-function.

Definition 2. Q-function has the (λ,ρ)-exponential decay property if, for any localized policy θ, for any $i\in V,s_{N_i^\kappa}\in S_{G_{N_i^\kappa}},s_{N_{-i}^\kappa},s'_{N_{-i}^\kappa}\in S_{G_{N_{-i}^\kappa}},u_{N_i^\kappa}\in U_{G_{N_i^\kappa}},u_{N_{-i}^\kappa},u'_{N_{-i}^\kappa}\in U_{G_{N_{-i}^\kappa}},a_{N_i^\kappa}\in A_{G_{N_i^\kappa}},a_{N_{-i}^\kappa},a'_{N_{-i}^\kappa}\in A_{G_{N_{-i}^\kappa}}$, Q_i^θ satisfies,

$$|Q_i^\theta(s_{N_i^\kappa},s_{N_{-i}^\kappa},u_{N_i^\kappa},u_{N_{-i}^\kappa},a_{N_i^\kappa},a_{N_{-i}^\kappa})-Q_i^\theta(s_{N_i^\kappa},s'_{N_{-i}^\kappa},u_{N_i^\kappa},u'_{N_{-i}^\kappa},a_{N_i^\kappa},a'_{N_{-i}^\kappa})|\leq\lambda\rho^{\kappa+1}\tag{6}$$

Theorem 1. *Assume $\forall i$, R_i is upper bounded by $\overline{R}$, the $(\frac{\overline{R}}{1-\gamma}, \gamma)$-exponential decay property of Q_i^θ holds.*

The proof of Theorem 1 can be found in the supplementary material. Theorem 1 indicates that the dependency of Q_i^θ on other agents decreases exponentially as the distance between them grows. Information on the agents that are far away from the learning agent could be unnecessary for determining the value of Q-function. Then, we propose the following truncated Q-functions for the state-(RM state)-action triples,

$$\tilde{Q}_i^\theta(s_{N_i^\kappa}, u_{N_i^\kappa}, a_{N_i^\kappa}) = \sum_{s_{N_{-i}^\kappa} \in S_{G_{N_{-i}^\kappa}}, u_{N_{-i}^\kappa} \in U_{G_{N_{-i}^\kappa}}, a_{N_{-i}^\kappa} \in A_{G_{N_{-i}^\kappa}}} c_i(s_{N_{-i}^\kappa}, u_{N_{-i}^\kappa}, a_{N_{-i}^\kappa}; s_{N_i^\kappa}, u_{N_i^\kappa}, a_{N_i^\kappa}) \cdot Q_i^\theta(s_{N_i^\kappa}, s_{N_{-i}^\kappa}, u_{N_i^\kappa}, u_{N_{-i}^\kappa}, a_{N_i^\kappa}, a_{N_{-i}^\kappa}) \tag{7}$$

where $c_i(s_{N_{-i}^\kappa}, u_{N_{-i}^\kappa}, a_{N_{-i}^\kappa}; s_{N_i^\kappa}, u_{N_i^\kappa}, a_{N_i^\kappa})$ is any non-negative weight and satisfies,

$$\forall (s_{N_i^\kappa}, u_{N_i^\kappa}, a_{N_i^\kappa}) \in S_{G_{N_i^\kappa}} \times U_{G_{N_i^\kappa}} \times A_{G_{N_i^\kappa}}, \sum_{s_{N_{-i}^\kappa} \in S_{G_{N_{-i}^\kappa}}, u_{N_{-i}^\kappa} \in U_{G_{N_{-i}^\kappa}}, a_{N_{-i}^\kappa} \in A_{G_{N_{-i}^\kappa}}} c_i(s_{N_{-i}^\kappa}, u_{N_{-i}^\kappa}, a_{N_{-i}^\kappa}; s_{N_i^\kappa}, u_{N_i^\kappa}, a_{N_i^\kappa}) = 1. \tag{8}$$

We use the term "truncated Q-function" to emphasize that the proposed Q-function is dependent on the information of the triple of state-RM state–action within a κ-hop neighborhood rather than global information. According to Lemma 6 and Proof of Lemma 17 in Qu et al. [19], the unique fixed point (optimal Q-function) of TD learning can be a linear combination of the full Q-function, where the full Q-function is the Q-function with global information as input. As we use the truncated Q-function to approximate the full Q-function, we define that the truncated Q is a linear combination of the full Q-function. In the implementation, the definition of truncated Q-function is not directly used. The truncated Q-function is learned by iteratively updating.

Theorem 2. *Under the (λ, γ)-exponential decay property of Q_i^θ, $\tilde{Q}_i^\theta$ satisfies,*

$$\sup_{(s,u,a) \in S_G \times U_G \times A_G} |\tilde{Q}_i^\theta(s_{N_i^\kappa}, u_{N_i^\kappa}, a_{N_i^\kappa}) - Q_i^\theta(s,u,a)| \leq \lambda \gamma^{\kappa+1} \tag{9}$$

The proof of Theorem 2 can be found in the supplementary material. Theorem 2 shows that $\tilde{Q}_i^\theta$ approximates Q_i^θ with high accuracy. $\tilde{Q}_i^\theta$ is a function of the κ-hop neighborhood's state-(RM state)-action triples, thus the dimension of $\tilde{Q}_i^\theta$ is smaller, which leads to the potential to handle large-scale systems. Next, we use $\tilde{Q}_i^\theta$ to approximate the policy gradient. The truncated policy gradient for agent i, $\tilde{J}_i$, is defined as $\tilde{J}_i(\theta) = \frac{1}{1-\gamma}\mathbb{E}_{(s,u)\sim d^\theta, a\sim\pi^\theta(\cdot|s,u)}[\frac{1}{n}\sum_{j\in N_i^\kappa} \tilde{Q}_j^\theta(s_{N_j^\kappa}, u_{N_j^\kappa}, a_{N_j^\kappa})] \cdot \nabla_{\theta_i} \log \pi_i^{\theta_i}(a_i|s_i.u_i)$, where $\tilde{Q}_j^\theta$ is any truncated Q-function in the form of Eq. (7).

Theorem 3. *If $\|\nabla_{\theta_i} \log \pi_i^{\theta_i}(a_i|s_i, u_i)\|$ is bounded by B_i, $\forall a_i, s_i, u_i$, $\|\tilde{J}_i(\theta) - \nabla_{\theta_i} J(\theta)\| \leq \frac{\lambda B_i}{1-\gamma}\gamma^{\kappa+1}$*

The proof of Theorem 3 can be found in the supplementary material. Theorem 3 indicates that the error of the truncated policy gradient is bounded. Thus, the truncated Q-function can be used to compute the policy gradient for policy improvement.

The DGRM algorithm adopts the actor-critic framework, specifically using temporal difference (TD) learning to train the truncated Q-function and policy gradient for the localized policy. The pseudocode of DGRM is given in Algorithm 1. The truncated Q-function is updated at every time step, and the policy parameters are updated in every episode. DGRM reduces the computational complexity as the truncated Q-function is conditioned on the state-(RM state)-action triple of κ-hop neighborhood instead of the global information, which results in a much smaller dimension. Thus, the complexity of DGRM is the local state-(RM state)-action space size of the largest κ-hop neighborhood. The approximated gradient of policy parameters is related to the truncated Q-function of the κ-hop neighborhood. Further, the authors in Qu et al. [19] claimed that under certain assumptions, the truncated Q-function generated by TD learning can be a good estimate of the optimal full Q-function, and the actor-critic algorithm will eventually find an $O(\gamma^{\kappa+1})$-approximation of a stationary point of $J(\theta)$. DGRM also follows the assumptions with the RM state, thus the approximating convergence conclusion still holds, which is illustrated in Theorem 4.

Assumption 1. The reward is bounded, $R_i(s_i, u_i, a_i) \leq \overline{R}, \forall i, s_i, u_i, a_i$. The individual state and action space sizes are upper bounded as $|S_i| \leq \overline{S}, |A_i| \leq \overline{A}, \forall i$.

Assumption 2. Q_i^θ holds the (λ, γ) exponential decay property.

The $(\frac{\overline{R}}{1-\gamma}, \gamma)$ exponential decay property in Theorem 1 holds for Q_i^θ if R_i is upper bounded by $\overline{R}$. Such requirement (i.e., bounded R_i) is assumed in Assumption 1. Therefore, Assumption 2 automatically holds under Assumption 1.

Assumption 3. The MDP environment gets sufficient local exploration. Every $(s_{N_i^\kappa}, u_{N_i^\kappa}, a_{N_i^\kappa})$ pair must be visited with some positive probability after some time.

Assumption 4. The policy gradient is bounded and Lipschitz continuous. For any i, a_i, s_i, u_i and θ_i, we assume $\|\nabla_{\theta_i} \log \pi_i^{\theta_i}(a_i|s_i, u_i)\| \leq B_i$. As a result, $\|\nabla_\theta \log \pi^\theta(a|s,u)\| \leq B = \sqrt{\sum_{i=1}^n B_i^2}$. Further, assume $\nabla J(\theta)$ is B'-Lipschitz continuous in θ.

Theorem 4. *Under Assumptions 1, 2, 3, and 4, DGRM will find an $O(\gamma^{\kappa+1})$-approximation of a stationary point of the objective function $J(\theta)$.*

Remark 1. Section 3.2 defines the reward function and labeling function as dependent solely on the local state (s_i), RM state (u_i), and action (a_i) of the learning agent. However, with a straightforward change of notation, our model, algorithm, and analysis can be readily extended to include a more general dependence on not only s_i, u_i, and a_i, but also the actions of the agent's neighbors $a_{N(i)}$.

Algorithm 1 DGRM Algorithm

```
Input: initial policy $\pi_i^{\theta_i}(0)$, initial Q-function, $s(0)$, $u(0)$, $\kappa, T$, $\gamma, \alpha_{\tilde{Q}}, \alpha_\pi$.
1: for each Episode $\tilde{e}$ do
2:   Take action $a_i(0) \sim \pi_i^{\theta_i}(\cdot|s_i(0), u_i(0))$ for all $i$.
3:   Get reward $R_i(0)$ for all $i$.
4:   while exist agent not finish the task and $t \leq T$ do
5:     Get state $s_i(t)$, RM state $u_i(t)$ for all $i$.
6:     Take action $a_i(t) \sim \pi_i^{\theta_i}(\cdot|s_i(t), u_i(t))$ for all $i$.
7:     Get reward $R_i(t)$ for all $i$.
8:     Calculate TD error for all $i$,
9:     $TD_i \leftarrow R_i(t-1) + \gamma\tilde{Q}_i^{t-1}(s_{N_i^k}(t), u_{N_i^k}(t), a_{N_i^k}(t)) - \tilde{Q}_i^{t-1}(s_{N_i^k}(t-1), u_{N_i^k}(t-1), a_{N_i^k}(t-1))$.
10:    Update the truncated Q-function for all $i$,
11:    $\tilde{Q}_i^t(s_{N_i^k}(t-1), u_{N_i^k}(t-1), a_{N_i^k}(t-1)) \leftarrow \tilde{Q}_i^{t-1}(s_{N_i^k}(t-1), u_{N_i^k}(t-1), a_{N_i^k}(t-1)) + \alpha_{\tilde{Q}_{t-1}} TD_i$.
12:  end while
13:  Calculate the approximated gradient for all $i$,
14:  $\tilde{g}_i(\tilde{e}) \leftarrow \sum_{t'=0}^t \gamma^t \frac{1}{n} \sum_{j\in N_i^k} \tilde{Q}_j^t(s_{N_i^k}(t'), u_{N_i^k}(t'), a_{N_i^k}(t')) \nabla_{\theta_i} \log \pi_i^{\theta_i(\tilde{e})}(a_i(t')|s_i(t'), u_i(t'))$.
15:  Update the policy parameters for all agents,
16:  $\theta_i(\tilde{e}+1) \leftarrow \theta_i(\tilde{e}) + \alpha_{\pi_{\tilde{e}}} \tilde{g}_i(\tilde{e})$.
17: end for
```

4.2. Deep DGRM algorithm

To further improve efficiency, we approximate both the policy and truncated Q-functions by deep neural networks. The pseudocode of deep DGRM is given in Algorithm 2. In every episode from line 2 to line 7, all the agents first interact with the environment. Then in line 8, each agent collects the necessary information during the episode for later actor and critic updating. Both the Q-network and policy network are updated in every episode. In line 11, the TD error is the averaged error over one episode. In line 15, the gradient of policy is conditioned on the averaged TD error of the κ-hop neighborhood, which indicates communication is required for policy improvement during training. When implementing the learned policy after training, communication is not required since the policy is only conditioned on each agent's own state-(RM state) pair.

Algorithm 2 Deep DGRM Algorithm

```
Input: initialized policy $\theta_i(0)$, initialized Q-network $\beta_i(0)$, $s(0)$, $u(0)$, $\kappa$, $T$, $\gamma$, $\alpha_{\tilde{Q}}$, $\alpha_\pi$
 1: for each Episode $\tilde{e}$ do
 2:   Take action $a_i(0) \sim \pi_i^{\theta_i}(\cdot|s_i(0), u_i(0))$ for all $i$.
 3:   Get reward $R_i(0)$ for all $i$.
 4:   while exist agent not finish the task and $t \leq T$ do
 5:     Get state $s_i(t)$, RM state $u_i(t)$ for all $i$.
 6:     Take action $a_i(t) \sim \pi_i^{\theta_i}(\cdot|s_i(t), u_i(t))$ for all $i$.
 7:     Get reward $R_i(t)$ for all $i$.
 8:     Store transition $\left(s_{N_i^\kappa}(t-1), u_{N_i^\kappa}(t-1), a_{N_i^\kappa}(t-1), R_i(t-1), s_{N_i^\kappa}(t), u_{N_i^\kappa}(t), a_{N_i^\kappa}(t)\right)$ for all $i$.
 9:   end while
10:   Calculate the TD error for all $i$,
11:   $TD_i \leftarrow \frac{1}{t}\sum_{t'=1}^{t}\left(R_i(t'-1) + \gamma\tilde{Q}_i^{\tilde{e}}\left(s_{N_i^\kappa}(t'), u_{N_i^\kappa}(t'), a_{N_i^\kappa}(t')\right) - \tilde{Q}_i^{\tilde{e}}\left(s_{N_i^\kappa}(t'-1), u_{N_i^\kappa}(t'-1), a_{N_i^\kappa}(t'-1)\right)\right)$.
12:   Update Q-network parameters for all $i$,
13:   $\beta_i(\tilde{e}+1) \leftarrow \beta_i(\tilde{e}) + \alpha_{\tilde{Q}} TD_i \nabla_{\beta_i}\tilde{Q}_i^{\tilde{e}}$.
14:   Update the policy-network parameters for all $i$,
15:   $\theta_i(\tilde{e}+1) \leftarrow \theta_i(\tilde{e}) + \alpha_\pi \frac{1}{|N_i^\kappa|}\sum_{j\in N_i^\kappa} TD_j \nabla_{\theta_i}\ln\pi_i$.
16: end for
```

5. Case studies

In this section, we implement DGRM on three case studies: (1) a wireless communication case with independent reward functions, (2) a wireless communication case with dependent reward functions, and (3) COVID-19 pandemic mitigation adapted from Della Rossa et al. [35].

5.1. Case study I: Wireless communication with independent reward functions

We consider a system consisting of ten users sending packets to the same destination. The destination is a fixed location that receives packets. Two types of packets are available, packet $\widetilde{A}$ and packet $\widetilde{B}$. The task for each user is to first send packet $\widetilde{A}$ successfully and then send packet $\widetilde{B}$ successfully within 16 time steps. A packet is sent successfully if the destination receives the packet. The interaction graph of the users is shown in Fig. 1. The interaction graph affects the task completion in the way that it defines the neighbors, which further determines the state transitions as the agent's state changes with respect to the agent's and its neighbor's states and actions. Whether an agent sends a packet successfully depends on its left neighbor's action at last time step. The detailed transition model is defined in Eq. (10). For example, agent i sends packet $\widetilde{A}$ successfully if the left neighbor of agent i $(i > 0)$ sent packet $\widetilde{B}$ successfully at last time step.

Fig. 2 illustrates the reward machine for each user to specify the task. We note that in this case study, the reward function and labeling function are independent, which depend on the local information

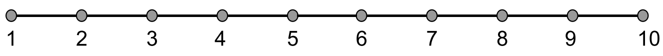


Fig. 1. The interaction graph of users in Case Study I. Number: user's index.

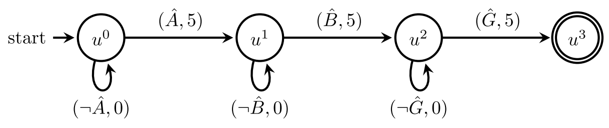


Fig. 2. Reward machines for each user in Case Study I. u^3: goal state.

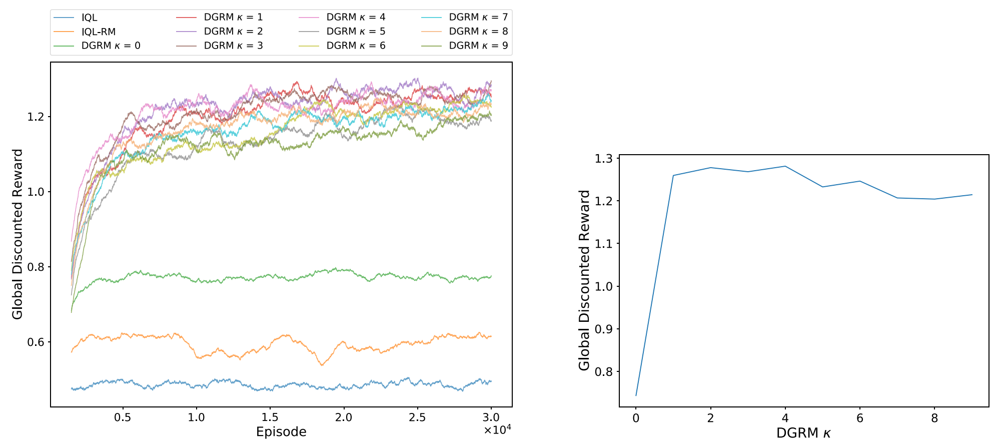
(s_i, u_i, a_i). Each edge is labeled by a tuple $(\hat{p}, \hat{r})$, where $\hat{p}$ is a propositional logic formula over $\mathcal{P}$, $\mathcal{P} = \{\hat{A}, \hat{B}, \hat{G}\}$, and $\hat{r} \in \mathbb{R}$ is a real number, defining the reward of the corresponding transition. The user starts from the RM state u^0. To transition from u^0 to u^1, the proposition $\hat{A}$ has to be true, and the transition outputs a reward of 5. $\hat{A}$ means the destination receives packet $\widetilde{A}$ from the user before the time step reaches 16. Additionally, the user is rewarded by 5 when it sends packet $\widetilde{B}$ successfully before the time step reaches 16 and after it sends $\widetilde{A}$ successfully, which is denoted by $\hat{B}$, and reaches u^2. u^3 is the goal state, expressing task accomplishment. When the time step reaches 16, the user arrives at u^3 and obtains a reward of 5.

The local state of user i at time t, $s_i(t) = (t, f_p^i)$, includes the current time step t, and f_p^i which indicates the type of packet that user i sends successfully after taking action $a_i(t-1)$. The action of user i at time t is to select from {0: send nothing, 1: send $\widetilde{A}$, 2: send $\widetilde{B}$}. We assume that sufficient packets are available during the entire planning horizon. The transition model of f_p^i is defined as,

$$\forall i > 0, \quad f_p^i(t+1) = \begin{cases} 1, & \text{if } f_p^{i-1}(t) = 2,\ a_i(t) = 1, \\ 2, & \text{if } f_p^{i-1}(t) = 1,\ a_i(t) = 2, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$
$$i = 0, \quad \text{if } a_i(t) = 1, \quad P(f_p^i(t+1) = 1) = 0.5, P(f_p^i(t+1) = 0) = 0.5.$$
$$i = 0, \quad \text{if } a_i(t) = 2, \quad P(f_p^i(t+1) = 2) = 0.5, P(f_p^i(t+1) = 0) = 0.5.$$

f_p^i obtains a value of 1 if sending $\widetilde{A}$ successfully and a value of 2 if sending $\widetilde{B}$ successfully. γ is set to 0.7 in the experiment. We utilize the softmax policy for the localized policy, which forms parameterized numerical probabilities for each state–action pair [36].

We run the DRGM algorithm with κ ranging from 0 to 9 and plot the global discounted reward during the training process in Fig. 3(a). To further evaluate the performance of the DGRM algorithm, a testing run of 1000 episodes is conducted for each DGRM with different κ values, and the averaged global discounted reward of 1000 episodes is plotted in Fig. 3(b). Fig. 3(b) shows that the averaged global discounted reward increases as κ increases from 0 to 2. DGRM with $\kappa = 3$ and 4 exhibit comparable performance to $\kappa = 2$. These findings are consistent with Theorem 2, which indicates that the error caused by the truncation of the Q-function decreases exponentially with κ, and Theorem 4, which implies that the truncated Q-function with a small κ can find the stationary point of the objective function $J(\theta)$. The contributions of agents located far from the learning agent are relatively small and have a lesser impact on the Q-function. The exponential decay appears to stop when $\kappa > 4$, which can be attributed to the increasing learning complexity with the size of the input information for the Q-function. In particular, DGRM with a large κ considers a triple of state-RM state–action with a large size, i.e., information of a large neighborhood, to learn the truncated Q-function. As a result, accurately estimating the truncated Q-function for large κ becomes more challenging, heavily relying on the learning rate. Furthermore, we conduct a comparison between DRGM and two baseline methods, namely independent Q-learning (IQL) [37] and independent Q-learning with reward machines (IQL-RM). In IQL and IQL-RM, each agent learns a localized Q-function.



(a) Global discounted rewards during training in Case Study I (denoised using a rolling window).

(b) Averaged global discounted rewards of 1000 episodes during testing for DGRM $\kappa = 0$ to 9 in Case Study I.

Fig. 3. Training and testing results of global discounted reward in Case Study I.

Specifically, the state for the Q-function in IQL includes the same information as the state for the localized policy of our DGRM algorithm. Due to the non-Markovian nature of the task, we provide IQL with two additional states detecting whether the user has sent $\widetilde{A}$ and $\widetilde{B}$ successfully in the required order. On the other hand, Q-functions in IQL-RM utilize the information as DGRM with $\kappa = 0$. The key difference between IQL-RM and DGRM with $\kappa = 0$ lies in their algorithmic structure, with IQL-RM using standard Q-learning and DGRM adopting an actor-critic structure. Fig. 3(a) shows that DGRM with $\kappa = 0$ outperforms IQL-RM, highlighting the advantages of our actor-critic structure. Additionally, IQL-RM achieves a higher reward compared to IQL, thereby highlighting the effectiveness of task specifications through reward machines.

5.2. Case study II: Wireless communication with dependent reward functions

This case study aims to evaluate the effectiveness of DGRM in the multi-agent system, as noted in Remark 1. The reward function and the labeling function depend on not only the local information (s_i, u_i, a_i), but also the actions of the learning agent's neighbors $a_{N(i)}$.

The environment in this case study is similar to that of Case Study I. The system consists of 5 agents and the interaction graph follows a linear structure where the agents are positioned in a line, identical to the interaction graph in Fig. 1. Node 1 represents the leftmost agent, and node 5 represents the rightmost agent. The task, state, action, and transition model are the same as the settings used in Case Study I. Fig. 4 displays the reward machine for each agent, which follows the same structure as the reward machine defined in Case Study I. The distinction between Case Study I and Case Study II lies in the approach employed to determine the required events for RM state transitions and rewards. In Case Study II, neighboring agents' actions are considered when determining the events and rewards, whereas in Case Study I, RM state transitions and rewards are based on local information (s_i, u_i, a_i). In detail, when the agent is in state u^0, it receives a reward of 5 if the event $\hat{C}$ becomes true. However, the occurrence of the event $\hat{C}$ is subject to two conditions being met. Firstly, the agent successfully sends packet $\tilde{A}$ within 16 time steps. Secondly, at the same time step, the agent's left neighbor chooses to send packet $\tilde{B}$. Similarly, event $\hat{D}$ requires that the agent sends packet $\tilde{B}$ successfully within 16 time steps after it sends $\tilde{A}$ successfully. Furthermore, its left neighbor selects to send $\tilde{A}$ at the current time step.

In this case study, we vary the parameter κ from 0 to 4 in DGRM and compare it with the performance of IQL and IQL-RM. We measure

$$\begin{tikzcd}
\text{start} \arrow[r] & u^0 \arrow[r, "(\hat{C},5)"] \arrow[loop below, "(\neg\hat{C},0)"] & u^1 \arrow[r, "(\hat{D},5)"] \arrow[loop below, "(\neg\hat{D},0)"] & u^2 \arrow[r, "(\hat{G},5)"] \arrow[loop below, "(\neg\hat{G},0)"] & u^3
\end{tikzcd}$$

Fig. 4. Reward machines for each user in Case Study II. u^3: goal state.

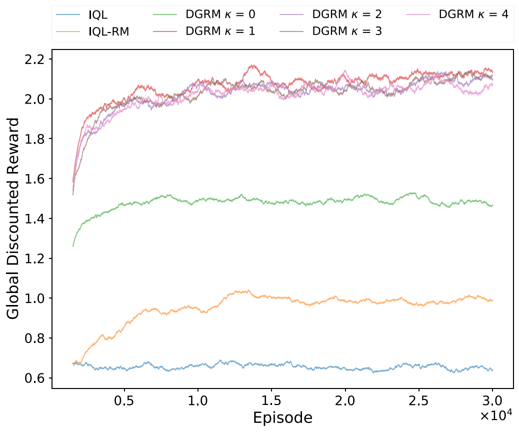
the performance of each algorithm in terms of global discounted rewards during training, as shown in Fig. 5(a). Additionally, we assess the average global discounted reward across 1000 testing episodes, as illustrated in Fig. 5(b). Our experimental results demonstrate that DGRM outperforms the baseline methods IQL and IQL-RM for all values of κ. Moreover, we observe that the performance of DGRM improves as κ increases when κ is less than 2. The policies learned by DGRM with $\kappa = 2$ and 3 achieve similar global discounted rewards to DGRM with $\kappa = 1$ during testing. These findings align with the exponential decay property of the truncated Q-function outlined in Theorem 2 and the convergence conclusion presented in Theorem 4. The increase in learning complexity associated with larger values of κ leads to a slight decrease in the average global discounted reward for DGRM with $\kappa = 4$.

5.3. Case study III: COVID-19 pandemic mitigation

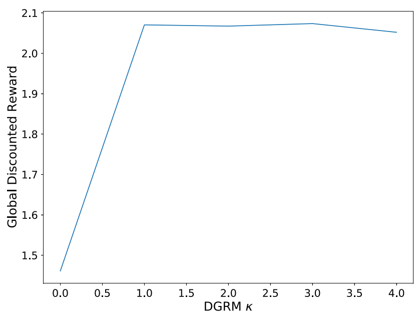
To the best of our knowledge, this case study provides the first attempt to use reward machines as specifications to mitigate epidemics and pandemics such as COVID-19.

We consider a network model of Italy [35], where Italy is modeled as a network of 20 regions and the parameters of each region's model are calculated from real data. Real data have been obtained from the public database of the Italian Civil Protection Agency. The total population is divided into five subgroups for each region i: susceptible $(\mathcal{S}_i)$, infected $(\mathcal{I}_i)$, quarantined $(\mathcal{Q}_i)$, hospitalized $(\mathcal{H}_i)$, recovered $(\mathcal{C}_i)$, and deceased $(\mathcal{D}_i)$. The detailed analysis of real data and the estimation of parameters of the pandemic model can be referred to Della Rossa et al. [35].

Della Rossa et al. [35] propose a bang–bang control strategy for each region according to the relative saturation level of its health system, mathematically, the ratio, $\tilde{r}_i$, between the number of hospitalized requiring care in ICU (estimated as $0.1\mathcal{H}_i$) over the number of available ICU beds $(\mathcal{T}_i^{\mathcal{H}})$ in the region. The regional lockdown is enforced when $\tilde{r}_i$ is larger than 0.5 and relaxed when $\tilde{r}_i$ is smaller than 0.2. During the lockdown, the region implements strict social distancing rules, and all fluxes in or out of the region are reduced to 70% of their original



(a) Global discounted rewards during training in Case Study II (denoised using a rolling window).



(b) Averaged global discounted rewards of 1000 episodes during testing for DGRM $\kappa = 0$ to 4 in Case Study II.

Fig. 5. Training and testing results of global discounted reward in Case Study II.

values. Mathematically, the social distancing parameter and fluxes are set as follows.

$$\rho_i = \begin{cases} \underline{\rho_i} & \frac{0.1\mathcal{H}_i}{\mathcal{T}_i^{\mathcal{H}}} \geq 0.5 \\ \min(1, 3\underline{\rho_i}) & \frac{0.1\mathcal{H}_i}{\mathcal{T}_i^{\mathcal{H}}} \leq 0.2 \end{cases} \tag{11}$$

where $\underline{\rho_i}$ is the minimum estimated value during the national lockdown and $\min(1, 3\underline{\rho_i})$ simulates the relaxation of social distancing rules.

$$\phi_{ij} = \begin{cases} 0.7\underline{\phi_{ij}} & \frac{0.1\mathcal{H}_i}{\mathcal{T}_i^{\mathcal{H}}} \geq 0.5 \\ \underline{\phi_{ij}} & \frac{0.1\mathcal{H}_i}{\mathcal{T}_i^{\mathcal{H}}} \leq 0.2, \end{cases} \forall i \neq j$$
$$\phi_{ji} = \begin{cases} 0.7\underline{\phi_{ji}} & \frac{0.1\mathcal{H}_i}{\mathcal{T}_i^{\mathcal{H}}} \geq 0.5 \\ \underline{\phi_{ji}} & \frac{0.1\mathcal{H}_i}{\mathcal{T}_i^{\mathcal{H}}} \leq 0.2, \end{cases} \forall i \neq j \tag{12}$$
$$\phi_{ii} = 1 - \sum_{i \neq j} \phi_{ij}$$

where $\underline{\phi_{ij}}$ is the original value of the flux from region i to region j before the pandemic, and $0.7\underline{\phi_{ij}}$ simulates the feedback flux control during a lockdown. We take the above bang–bang control as a baseline method for comparison with our proposed method.

In this case study, we aim to control the spread of COVID-19 among the 20 regions of Italy over a period of 4 weeks. A region is defined to be in a severe situation when the total hospital capacity is saturated, denoted by $\frac{0.1\mathcal{H}_i}{\mathcal{T}_i^{\mathcal{H}}} \geq 0.5$, which is the same as the saturation level when a lockdown is adopted in the bang–bang control. The control objective is that each region would not be in a severe situation for 2 consecutive weeks and avoid a long time lockdown of 2 consecutive weeks since the costs of continuing severe restrictions could be great relative to likely benefits in lives saved [38].

To conduct our experiment, we establish a labeled graph-based multi-agent MDP for the network model of Italy and a reward machine as task specification for each region. We define that region j is region i's neighbor if there are fluxes of people traveling between two regions. The simulation is conducted with a discretization step of 1 day and starts on a Monday. At the first step of the simulation, each region is set to be in a severe situation, $\mathcal{H}_i(0) = 6\mathcal{T}_i^{\mathcal{H}}$, the values of $\mathcal{I}_i(0)$, $\mathcal{Q}_i(0)$, $\mathcal{C}_i(0)$, and $\mathcal{D}_i(0)$ are the same as the experiment in Della Rossa et al. [35]. Each region has four possible actions at each time step: no restrictions, implementing social distancing ($\rho_i = \underline{\rho_i}$), controlling fluxes between neighboring regions ($\phi_{ij} = 0.7\underline{\phi_{ij}}, \forall i \neq j$), and adopting a lockdown. The local state of region i at day t, $s_i(t)$, includes the characterization of its population, high-level features of this week's situation, and the simulated time step,

$$s_i(t) = (\mathcal{S}_i(t), \mathcal{I}_i(t), \mathcal{R}_i(t), \mathcal{H}_i(t), \mathcal{Q}_i(t), \mathcal{D}_i(t), \tilde{v}_i, \tilde{l}_i, t) \tag{13}$$

where $\tilde{v}_i$ and $\tilde{l}_i$ are the numbers of days when the region is in a severe situation and the number of days when the region adopts a lockdown since this Monday, respectively. The transitions of $\mathcal{S}_i(t), \mathcal{I}_i(t), \mathcal{R}_i(t), \mathcal{H}_i(t)$, $\mathcal{Q}_i(t), \mathcal{D}_i(t)$ follow the pandemic model's dynamics in Della Rossa et al. [35]. The set of events is $\mathcal{P}_i = \{\epsilon_0, \epsilon_1, v^0l^0, v^{0.5}l^0, v^1l^0, v^0l^{0.5}, v^{0.5}l^{0.5}, v^1l^{0.5}, v^0l^1, v^{0.5}l^1, v^1l^1\}$, where ϵ_0 is an empty event when the current day is not Monday; ϵ_1 is the event leading to the goal state and indicates the current day is the last time step of the entire simulation; the superscript of v and l describe the level of the situation's severity and the frequency of lockdown implementation during the previous seven days (from Monday to Sunday), respectively. Mathematically,

$$v^k = \begin{cases} v^0 & \tilde{v} = 0 \\ v^1 & \tilde{v} = 7 \\ v^{0.5} & 0 < \tilde{v} < 7 \end{cases} \qquad l^k = \begin{cases} l^0 & \tilde{l} = 0 \\ l^1 & \tilde{l} = 7 \\ l^{0.5} & 0 < \tilde{l} < 7 \end{cases} \tag{14}$$

We note that the reward machine for each region is the same for simplicity. The Reward Machine gives each region guidance on the control objective, which is shown in Fig. 6. The agent starts at u^0, assuming that during the previous week, the region was in a severe situation for at least 1 day but less than 7 days and did not implement a lockdown. The RM states $(u^4, u^8, u^9, u^{10}, u^{11}, u^{12}, u^{15})$ are sink states which are not encouraged, indicating that the region has been in a severe situation or has implemented a lockdown for two consecutive weeks. u^{16} is the goal state and can be reached from all the states except the sink states. Due to the limited space in Fig. 6, the required events for RM state transitions which should be marked on the edges in Fig. 6 are listed in Table 1. For instance, during the first week, if the region was in a severe situation and implemented a lockdown for seven consecutive days, denoted by v^1l^1, the region is directed to u^5 on the second week's Monday. From Tuesday to Sunday, the agent would remain at the same RM state due to the empty event ϵ_0. The reward function σ_i is defined

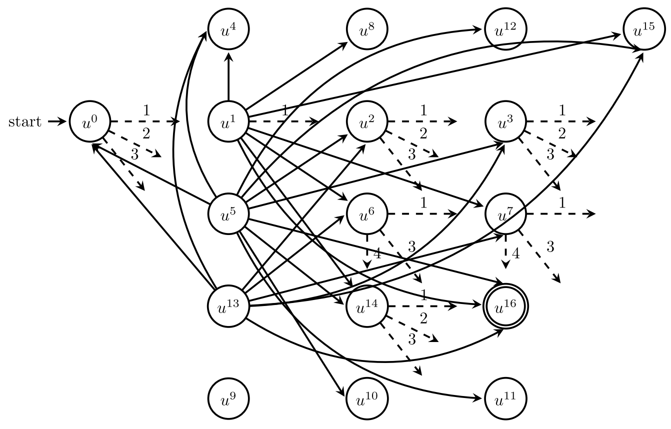


Fig. 6. Reward Machine structure for each agent in Case Study III. u^{16}: goal state; $\xrightarrow{1}$: pointing to the RM state set $\{u^0, u^1, u^2, u^3\}$; $\xrightarrow{2}$: pointing to the RM state set $\{u^5, u^6, u^7\}$; $\xrightarrow{3}$: pointing to the RM state set $\{u^{13}, u^{14}, u^{16}\}$; $\xrightarrow{4}$: pointing to the RM state set $\{u^9, u^{10}, u^{11}\}$. For simplicity, self-loops at $\{u^5, u^6, u^7, u^{13}\}$ are omitted. The required events for RM state transitions are listed in Table 1.

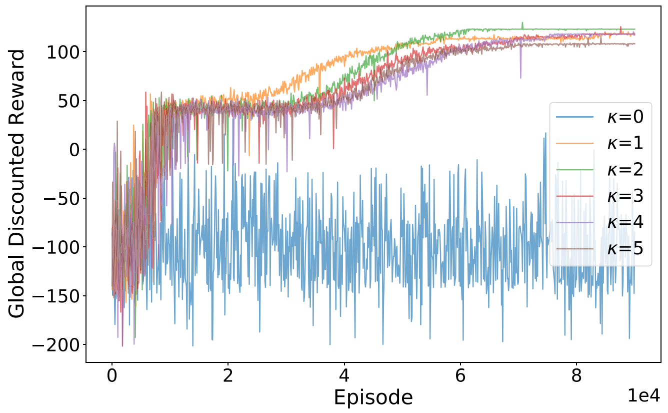


Fig. 7. Global discounted rewards during the training process in Case Study III (denoised using B-spline interpolation).

as follows.

$$\sigma_i(u_i, L_i(s_i, a_i, s_i'))$$
$$= \begin{cases} -600 & u_i = u^4, u^8, u^9, u^{10}, u^{11}, u^{12}, u^{15} \\ -250v - 50l + 400 & u_i' = u^3 \\ -250v - 50l - 300 & u_i' = u^4, u^8, u^9, u^{10}, u^{11}, u^{12}, u^{15} \\ -250v - 50l + 250 & L_i(s_i, a_i, s_i') = \epsilon_1 \\ 0 & L_i(s_i, a_i, s_i') = \epsilon_0 \\ -250v - 50l + 200 & \text{otherwise} \end{cases} \tag{15}$$

where $u_i' = \delta_i(u_i, L_i(s_i, a_i, s_i'))$.

In the experiment, we set $\gamma = 0.9$. We use two fully connected layers with [256, 128] units and ReLU activations for the actor and two fully connected layers with [256, 128] units and Tanh activation for the critic. We run the deep DGRM algorithm with $\kappa = 0$ up to $\kappa = 5$ and plot the global discounted rewards during the training process in Fig. 7. With the increase in κ, the global discounted reward increases when $\kappa \leq 2$. This is also demonstrated in Fig. 8, which shows the performance of the baseline (bang–bang control policy in Della Rossa et al. [35]) and the results of deep DGRM obtained by 20 independent simulation runs using the converged policy for each κ. The deep DGRM algorithm with $\kappa > 0$ outperforms $\kappa = 0$, which is the independent learner method in Tan [37]. The deep DGRM algorithm with $\kappa = 1$ improves the global discounted reward by 218% compared to $\kappa = 0$. The deep DGRM with

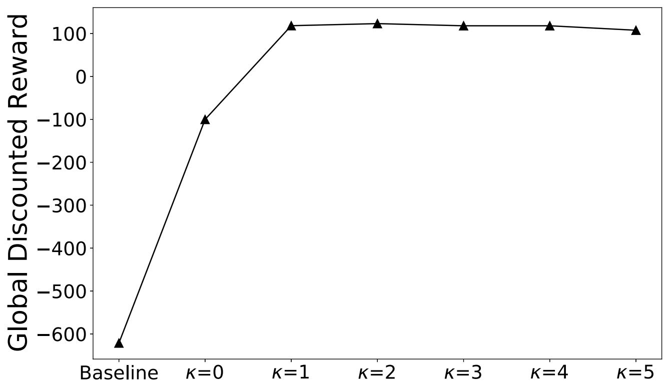


Fig. 8. Comparison of global discounted rewards in Case Study III.

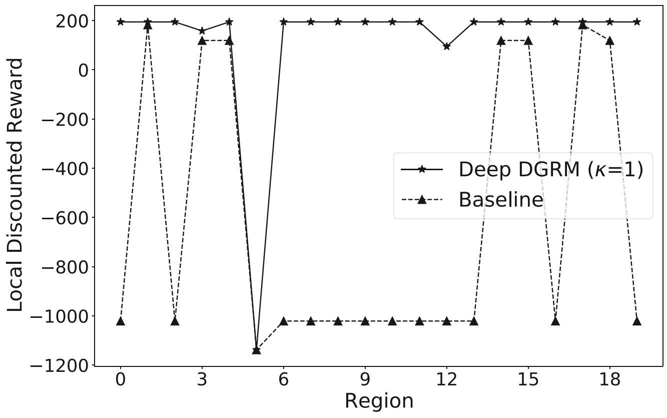


Fig. 9. Comparison of local discounted rewards of each region in Case Study III.

$\kappa = 2$ has limited improvement (4%) on the performance compared to $\kappa = 1$. When $\kappa > 2$, the global discounted reward stops growing due to the cascading of local interactions, which is consistent with the results in Case Study I. Moreover, the learning complexity grows with κ as discussed in Case Study I. As we use the deep neural networks in Case Study III, the model contains many parameters and the performance heavily depends on the learning rate. By spending much more effort in tuning, DGRM with a large κ has the potential to reach the same reward as DGRM with a small κ (e.g., $\kappa = 1, 2$). With the deep DGRM algorithm, information of the 1-hop neighborhood would be sufficient for good performance in this experiment, and the global information is unnecessary. Moreover, the deep DGRM with $\kappa = 1$ improves the global accumulated reward by 119% compared to the baseline.

To take a deeper look into the results, Fig. 9 shows the local discounted rewards of each region using the deep DRGM algorithm and the baseline method, with the reward function determined by the reward machine. From Fig. 9, it can be seen that only region 5 receives a large penalty and fails the task with the deep DGRM algorithm, while a total of 13 regions fail the task utilizing the baseline method. We also plot the result of $\frac{0.1H_i}{T_i^H}$ (the relative saturation level of its health system) during the entire simulation for each region with $\kappa = 1$ in Fig. 10. The result indicates our proposed deep DGRM algorithm can guide 19 among 20 regions to avoid being in a severe situation and adopting a lockdown for 2 consecutive weeks in a period of 4 weeks, which reduces the spread of disease while considering the economic factors.

6. Conclusion

In this work, reward machines are leveraged to express complex temporally extended tasks. A decentralized learning algorithm for

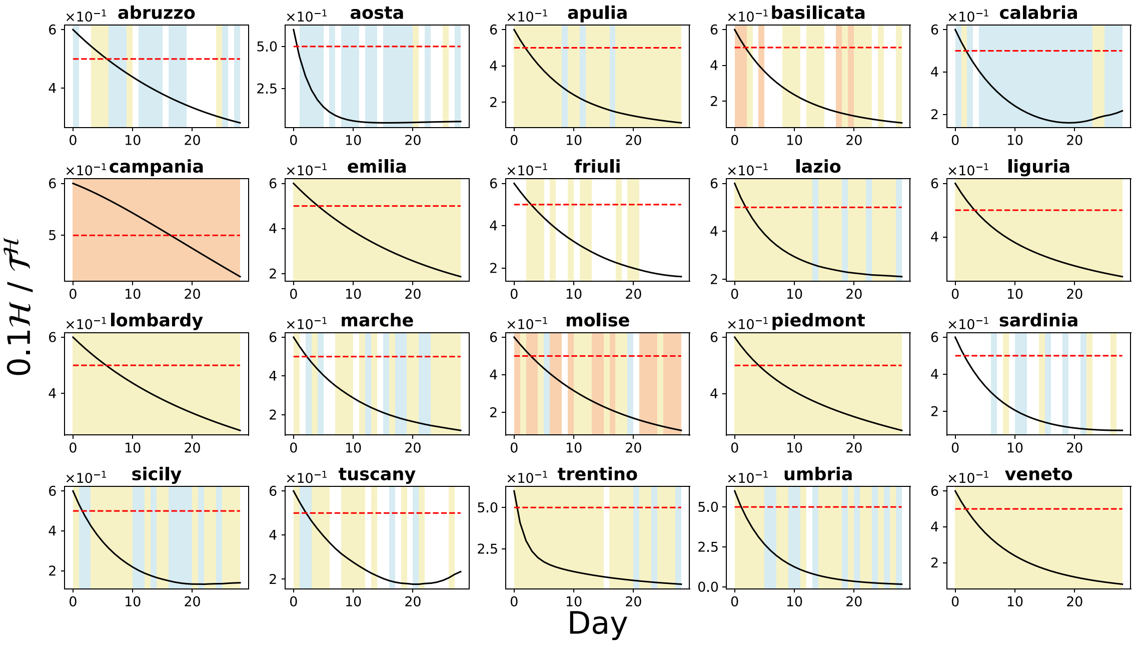


Fig. 10. Result of $\frac{0.1\mathcal{H}_i}{\mathcal{T}_i^{\mathcal{H}}}$ of each region with $\kappa = 1$ in Case Study III. Red dashed line: the threshold of 0.5 for $\frac{0.1\mathcal{H}_i}{\mathcal{T}_i^{\mathcal{H}}}$ when the region is in a severe condition. Background: orange: select to lockdown; yellow: select to practice social distancing; blue: select to control the flux.

Table 1
Required events for reward machine state transition in Case Study III.

u transition	event	u transition	event
$u_i \to u_0,\ i \in \{1,2,3,5,6,7,13,14\}$	$v^{0.5}l^0$	$u_0 \to u_0$	$\epsilon_0 \vee v^{0.5}l^0$
$u_i \to u_1,\ i \in \{0,2,3,6,7,14\}$	v^0l^1	$u_1 \to u_1$	$\epsilon_0 \vee v^0l^1$
$u_i \to u_2,\ i \in \{0,1,3,5,6,7,13,14\}$	$v^0l^{0.5}$	$u_2 \to u_2$	$\epsilon_0 \vee v^0l^{0.5}$
$u_i \to u_3,\ i \in \{0,1,2,5,6,7,13,14\}$	v^0l^0	$u_3 \to u_3$	$\epsilon_0 \vee v^0l^0$
$u_i \to u_5,\ i \in \{0,2,3,14\}$	v^1l^1	$u_5 \to u_5$	$\epsilon_0 \vee v^1l^1$
$u_i \to u_6,\ i \in \{0,1,2,3,13,14\}$	$v^1l^{0.5}$	$u_6 \to u_6$	$\epsilon_0 \vee v^1l^{0.5}$
$u_i \to u_7,\ i \in \{0,1,2,3,13,14\}$	v^1l^0	$u_7 \to u_7$	$\epsilon_0 \vee v^1l^0$
$u_i \to u_{13},\ i \in \{0,2,3,6,7,14\}$	$v^{0.5}l^1$	$u_{13} \to u_{13}$	$\epsilon_0 \vee v^{0.5}l^1$
$u_i \to u_{14},\ i \in \{0,1,2,3,5,6,7,13\}$	$v^{0.5}l^{0.5}$	$u_{14} \to u_{14}$	$\epsilon_0 \vee v^{0.5}l^{0.5}$
$u_i \to u_4,\ i \in \{1,5,13\}$	v^0l^1	$u_i \to u_8,\ i \in \{1,13\}$	v^1l^1
$u_i \to u_9,\ i \in \{6,7\}$	v^1l^1	$u_i \to u_{10},\ i \in \{5,6,7\}$	$v^1l^{0.5}$
$u_i \to u_{11},\ i \in \{5,6,7\}$	v^1l^0	$u_i \to u_{12},\ i \in \{5\}$	v^1l^1
$u_i \to u_{15},\ i \in \{1,5,13\}$	$v^{0.5}l^1$	$u_i \to u_{16},\ i \in \{0,1,2,3,5,6,7,13,14\}$	ϵ_1

graph-based MDP is introduced. We propose a truncated Q-function that uses the local information of the κ-hop neighborhood and can be computed efficiently. We also prove that the approximation error is bounded. After training, agents can implement the policies independently. We also combine the decentralized architecture and function approximation by deep neural networks to enable the application to large-scale MARL or continuous state problems. Furthermore, this is the first work to use reward machines to mitigate the COVID-19 pandemic through task specification, and we show performance improvement over the baseline method.

The proposed approach can be widely applied to various real-world large-scale sparse networked systems that can be formulated as a graph-based MDP, such as the power grid, communication systems, pandemic networks, and smart traffic or infrastructure systems. The current work assumes each agent has an individual reward machine and the reward machine is known, but it would be possible to learn reward machines from experience [24] and decompose a team-level task to a collection of reward machines [21]. We use a standard actor-critic algorithm, one possible research direction is to use more advanced RL algorithms such as Proximal Policy Optimization (PPO) [39] to allow for solving continuous action problems. Leveraging strategies such as Hindsight Experience Replay [40] to learn policies faster is also worth investigating.

CRediT authorship contribution statement

Jueming Hu: Conceptualization, Methodology, Software, Analysis, Validation, Writing – original draft. **Zhe Xu:** Conceptualization, Methodology, Analysis, Writing – review, Supervision. **Weichang Wang:** Methodology, Analysis, Writing – review. **Guannan Qu:** Analysis, Writing – review. **Yutian Pang:** Analysis, Writing – review. **Yongming Liu:** Writing – review, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

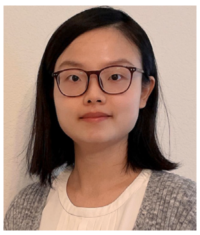
The research reported in this paper was supported by funds from NASA University Leadership Initiative program (Contract No. NNX17A J86A, PI: Yongming Liu, Technical Officer: Anupa Bajwa) and by funds from the National Science Foundation (Award: 2331781). The support is gratefully acknowledged.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.neucom.2023.126974.

References

[1] T. Dean, K. Kanazawa, A model for reasoning about persistence and causation, Comput. Intell. 5 (2) (1989) 142–150.
[2] C. Guestrin, D. Koller, R. Parr, S. Venkataraman, Efficient solution algorithms for factored MDPs, J. Artificial Intelligence Res. 19 (2003) 399–468.
[3] N. Forsell, R. Sabbadin, Approximate linear-programming algorithms for graph-based Markov decision processes, in: Proceedings of the 2006 Conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29–September 1, 2006, Riva Del Garda, Italy, 2006, pp. 590–594.
[4] Q. Cheng, Q. Liu, F. Chen, A.T. Ihler, Variational planning for graph-based MDPs, Adv. Neural Inf. Process. Syst. 26 (2013) 2976–2984.
[5] V.D. Blondel, J.N. Tsitsiklis, A survey of computational complexity results in systems and control, Automatica 36 (9) (2000) 1249–1274.
[6] P. Hernandez-Leal, B. Kartal, M.E. Taylor, A survey and critique of multiagent deep reinforcement learning, Auton. Agents Multi-Agent Syst. 33 (6) (2019) 750–797.
[7] M.L. Littman, Markov games as a framework for multi-agent reinforcement learning, in: Machine Learning Proceedings 1994, Elsevier, 1994, pp. 157–163.
[8] C. Claus, C. Boutilier, The dynamics of reinforcement learning in cooperative multiagent systems, AAAI/IAAI 1998 (746–752) (1998) 2.
[9] M.L. Littman, Value-function reinforcement learning in Markov games, Cogn. Syst. Res. 2 (1) (2001) 55–66.
[10] J. Hu, M.P. Wellman, Nash Q-learning for general-sum stochastic games, J. Mach. Learn. Res. 4 (Nov) (2003) 1039–1069.
[11] B. Silver, W. Frawley, G. Iba, J. Vittal, K. Bradford, ILS: A framework for multi-paradigmatic learning, in: Machine Learning Proceedings 1990, Elsevier, 1990, pp. 348–356.
[12] J. Hu, Y. Liu, UAS conflict resolution integrating a risk-based operational safety bound as airspace reservation with reinforcement learning, in: AIAA Scitech 2020 Forum, 2020, p. 1372.
[13] W. Wang, Y. Liu, R. Srikant, L. Ying, 3M-RL: Multi-resolution, multi-agent, mean-field reinforcement learning for autonomous UAV routing, IEEE Trans. Intell. Transp. Syst. (2021).
[14] K. Zhang, Z. Yang, H. Liu, T. Zhang, T. Basar, Fully decentralized multi-agent reinforcement learning with networked agents, in: International Conference on Machine Learning, PMLR, 2018, pp. 5872–5881.
[15] C. Guestrin, D. Koller, R. Parr, Multiagent planning with factored MDPs, in: NIPS, Vol. 1, 2001, pp. 1523–1530.
[16] M. Cubuktepe, Z. Xu, U. Topcu, Distributed policy synthesis of multi-agent systems with graph temporal logic specifications, IEEE Trans. Control Netw. Syst. (2021).
[17] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, S. Whiteson, Counterfactual multi-agent policy gradients, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, (1) 2018.
[18] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, 2017, arXiv preprint arXiv:1706.02275.
[19] G. Qu, A. Wierman, N. Li, Scalable reinforcement learning of localized policies for multi-agent networked systems, in: Learning for Dynamics and Control, PMLR, 2020, pp. 256–266.
[20] R.T. Icarte, T. Klassen, R. Valenzano, S. McIlraith, Using reward machines for high-level task specification and decomposition in reinforcement learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 2107–2116.
[21] C. Neary, Z. Xu, B. Wu, U. Topcu, Reward machines for cooperative multi-agent reinforcement learning, 2020, arXiv preprint arXiv:2007.01962.
[22] D. DeFazio, S. Zhang, Learning quadruped locomotion policies with reward machines, 2021, arXiv preprint arXiv:2107.10969.
[23] A. Shah, S. Li, J. Shah, Planning with uncertain specifications (Puns), IEEE Robot. Autom. Lett. 5 (2) (2020) 3414–3421.
[24] Z. Xu, I. Gavran, Y. Ahmad, R. Majumdar, D. Neider, U. Topcu, B. Wu, Joint inference of reward machines and policies for reinforcement learning, in: Proceedings of the International Conference on Automated Planning and Scheduling, Vol. 30, 2020, pp. 590–598.
[25] R. Toro Icarte, E. Waldie, T. Klassen, R. Valenzano, M. Castro, S. McIlraith, Learning reward machines for partially observable reinforcement learning, Adv. Neural Inf. Process. Syst. 32 (2019) 15523–15534.
[26] D. Furelos-Blanco, M. Law, A. Russo, K. Broda, A. Jonsson, Induction of subgoal automata for reinforcement learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 04, 2020, pp. 3890–3897.
[27] D. Furelos-Blanco, M. Law, A. Jonsson, K. Broda, A. Russo, Induction and exploitation of subgoal automata for reinforcement learning, J. Artificial Intelligence Res. 70 (2021) 1031–1116.
[28] M. Hasanbeig, N.Y. Jeppu, A. Abate, T. Melham, D. Kroening, DeepSynth: Automata synthesis for automatic task segmentation in deep reinforcement learning, in: The Thirty-Fifth {AAAI} Conference on Artificial Intelligence,{AAAI}, Vol. 2, No. 9, Baltimore, MD, USA, August 15–17, 27th {USENIX} Security Symposium,{USENIX} Security 18, 2021, p. 36.
[29] G. Rens, J.-F. Raskin, Learning non-Markovian reward models in mdps, 2020, arXiv preprint arXiv:2001.09293.
[30] R. Parr, S. Russell, Reinforcement learning with hierarchies of machines, in: Advances in Neural Information Processing Systems, Morgan Kaufmann Publishers, 1998, pp. 1043–1049.
[31] R.S. Sutton, D. Precup, S. Singh, Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning, Artif. Intell. 112 (1–2) (1999) 181–211.
[32] T.G. Dietterich, Hierarchical reinforcement learning with the MAXQ value function decomposition, J. Artif. Intell. Res. 13 (2000) 227–303.
[33] R.T. Icarte, T.Q. Klassen, R. Valenzano, S.A. McIlraith, Reward machines: Exploiting reward function structure in reinforcement learning, 2020, arXiv preprint arXiv:2010.03950.
[34] R.S. Sutton, D.A. McAllester, S.P. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: Advances in Neural Information Processing Systems, 2000, pp. 1057–1063.
[35] F. Della Rossa, D. Salzano, A. Di Meglio, F. De Lellis, M. Coraggio, C. Calabrese, A. Guarino, R. Cardona, P. DeLellis, D. Liuzza, et al., Intermittent yet coordinated regional strategies can alleviate the COVID-19 epidemic: a network model of the Italian case, 2020, arXiv preprint arXiv:2005.07594.
[36] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[37] M. Tan, Multi-agent reinforcement learning: Independent vs. cooperative agents, in: Proceedings of the Tenth International Conference on Machine Learning, 1993, pp. 330–337.
[38] D.K. Miles, M. Stedman, A.H. Heald, "Stay at Home, Protect the National Health Service, Save Lives": a cost benefit analysis of the lockdown in the United Kingdom, Int. J. Clin. Pract. 75 (3) (2021) e13674.
[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347.
[40] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, W. Zaremba, Hindsight experience replay, 2017, arXiv preprint arXiv:1707.01495.



Jueming Hu is a Ph.D. candidate in the School for Engineering of Matter, Transport & Energy at Arizona State University. She completed her master's degree at Arizona State University in 2018 and obtained her bachelor's degree from Southeast University in China in 2017. Her research interests include Reinforcement Learning, UAS traffic management, optimization, and operation research.

Zhe Xu received the B.S. and M.S. degrees in Electrical Engineering from Tianjin University, Tianjin, China, in 2011 and 2014, respectively. He received the Ph.D. degree in Electrical Engineering at Rensselaer Polytechnic Institute, Troy, NY, in 2018. He is currently an assistant professor in the School for Engineering of Matter, Transport, and Energy at Arizona State University. Before joining ASU, he was a postdoctoral researcher in the Oden Institute for Computational Engineering and Sciences at the University of Texas at Austin, Austin, TX. His research interests include formal methods, autonomous systems, control systems and reinforcement learning.

Weichang Wang received his Ph.D. degree and Master's degree in School of Electrical, Computer and Energy Engineering at Arizona State University. He obtained his Bachelor's degree in School of Electronic Information and Communications at Huazhong University of Science and Technology. His research interests include Wireless Communication Network, Resource Allocation in Stochastic Systems, Multi-agent Reinforcement Learning, and Internet of Things (IoT).

Guannan Qu is an assistant professor in the Electrical and Computer Engineering Department of Carnegie Mellon University. He received his B.S. degree in electrical engineering from Tsinghua University in Beijing, China in 2014, and his Ph.D. in applied mathematics from Harvard University in Cambridge, Massachusetts in 2019. He was a CMI and Resnick postdoctoral scholar in the Department of Computing and Mathematical Sciences at California Institute of Technology from 2019 to 2021. He is the recipient of Caltech Simoudis Discovery Award, PIMCO Fellowship, Amazon AI4Science Fellowship, and IEEE SmartGridComm Best Student Paper Award. His research interest lies in control, optimization, and machine/reinforcement learning with applications to power systems, multi-agent systems, Internet of things, and smart cities.

Yutian Pang is a Ph.D. candidate in mechanical and aerospace engineering at Arizona State University. He obtained his Bachelor's degree from Huazhong University of Science and Technology in 2017 and his master's degree from Arizona State University in 2018. His research interests include data-driven intelligent decision support for aviation systems, Bayesian machine learning for uncertainty quantification, and robust autonomy.

Yongming Liu is a professor of aerospace and mechanical engineering with the School for Engineering of Matter, Transport and Energy at Arizona State University. He heads the Prognostic Analysis and Reliability Assessment Laboratory (PARA). His research interests range from fatigue and fracture of engineering materials and structures, probabilistic computational mechanics, risk assessment and management to multi-physics damage modeling and structural durability, multi-scale uncertainty quantification and propagation, imaging-based experimental testing, diagnostics and prognostics.