ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9414597

MAKF-SR: MULTI-AGENT ADAPTIVE KALMAN FILTERING-BASED SUCCESSOR REPRESENTATIONS

Mohammad Salimibeni[†]*, Parvin Malekzadeh*[‡]*, Arash Mohammadi*[†]
Petros Spachos[††], *Konstantinos N. Plataniotis*[‡]
[†]Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada
[††] School of Engineering, University of Guelph, Guelph, ON, Canada
[‡]Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada

ABSTRACT

The paper is motivated by the importance of the Smart Cities (SC) concept for future management of global urbanization and energy consumption. Multi-agent Reinforcement Learning (RL) is an efficient solution to utilize large amount of sensory data provided by the Internet of Things (IoT) infrastructure of the SCs for city-wide decision making and managing demand response. Conventional Model-Free (MF) and Model-Based (MB) RL algorithms, however, use a fixed reward model to learn the value function rendering their application challenging for ever changing SC environments. Successor Representations (SR)-based techniques are attractive alternatives that address this issue by learning the expected discounted future state occupancy, referred to as the SR, and the immediate reward of each state. SR-based approaches are, however, mainly developed for single agent scenarios and have not yet been extended to multi-agent settings. The paper addresses this gap and proposes the Multi-Agent Adaptive Kalman Filtering-based Successor Representation (MAKF-SR) framework. The proposed framework can adapt quickly to the changes in a multi-agent environment faster than the MF methods and with a lower computational cost compared to MB algorithms. The proposed MAKF-SR is evaluated through a comprehensive set of experiments illustrating superior performance compared to its counterparts.

***Index Terms*—** Reinforcement Learning, Successor Representations, Kalman Temporal Difference.

1. INTRODUCTION

It is expected that in near future Smart Cities (SCs) [1] flourish across the globe aiming at efficient management of global urbanization and energy consumption [2] to assist in having more environmental friendly societies [3]. The concept of Internet of Things (IoT) [4–6] plays a vital role in development of SCs by presenting an interactive infrastructure that can provide a large amount of sensory data in a real time fashion to monitor, manage, and control the SC. On one hand, the large volume of data provided by IoT devices can assist in implementation of predictive analytic in SCs for policy design and city-wide decision making. On the other hand, information obtained from IoT data sources can assist in managing demand response for coordination of energy resources especially in smart buildings. The key underlying challenge here is to process such large amounts of IoT data and perform efficient actions. Reinforcement Learning (RL) techniques [7–9] are attractive solutions for this challenge (to process IoT data) as RL approaches can learn and adapt to the environmental changes and can directly learn from historical data in the SC environments where having a unified model is too expensive to develop. Although multi-agent RL-based approaches have been used for smart public governance of SCs [10] and to control building energy systems [11], this field is still in its infancy.

Literature Review: Generally speaking, RL is a class of Machine Learning (ML) algorithms enabling autonomous agents to learn the optimal control (action) policy by using trial and error based on the feedback received from the environment after each action [7]. Unlike supervised learning, RL approaches do not need labeled data, which is difficult to acquire in the context of SCs. Traditionally, RL algorithms are categorized into two main classes: (i) Model-Free (MF) methods [7–9], which learn the value function using sample trajectories, and; (ii) Model-Based (MB) methods [12] that estimate transition and reward functions through search trees or dynamic programming. Algorithms belonging to the former category (MF algorithms), typically, fail to rapidly adjust an agent to localized changes in the reward function. The MB algorithms can quickly adjust to the environmental changes but with a high computational cost [13, 14]. As a remedy to aforementioned adaptation problems, Successor Representations (SR) methods [15, 16] are proposed recently as an alternative class of RL algorithms. The SR methods provide computational efficiency, comparable to that of MF algorithms, concurrently with the flexibility of MB algorithms. The SR-based algorithms learn the expected immediate reward received after each action together with the expected discounted future state occupancy (i.e., the SR). SR-based approaches are mainly developed for single agent scenarios and have not yet been extended to multi-agent scenarios. The paper addresses this gap for potential applications to predictive analytic and demand response in SCs.

When it comes to multi-agent RL, conventional approaches developed for single agent scenarios such as Q-Learning or policy gradient are not readily suited for adoption. A key issue is that, typically, the environment becomes non-stationary from the perspective of an individual agent given that each agent's policy is changing as training progresses. Within the context of deep Q-learning, this results in stabilizing issues due to difficulties in proper use of past localized experience. Within the context of policy gradient methods, typically, high variance is observed for coordination of multiple agents. While MB solutions can be used, this is challenging due to the need for construction of a differentiable model of the underlying physical dynamics and imposing restrictive assumptions on the interactions between agents. Utilization of DNNs has enabled RL methods [17–21] to drive optimal policies for sophisticated multi-agent scenarios. While DNN approaches like Deep Q-Networks (DQN) and Deep Deterministic Policy Gradient (DDPG) have reached considerable results, but they suffer from some key drawbacks including the overfitting issue; high sensitivity to parameter selection; sample

This Project was partially supported by the Department of National Defence's Innovation for Defence Excellence and Security (IDEaS) program, Canada.

inefficiency, and; the need for a large number of episodes to achieve acceptable results. An alternative for these approximations is to use a set of weighted local estimators and convert the approximation problem into a weight estimation problem using Radial Basis Functions (RBFs) [22]. Among various local estimators , RBFs are more suitable for systems with continuous states, and their performance is comparable to those using Fourier basis [23]. Finally, RBFs can represent gradual-continuous transitions as such are considered here for the SR and reward function estimation within multi-agent environments. Parameters of the RBFs are initialized for each agent, and are adapted after observing the system's transition based on cross entropy and gradient descent methods.

Contributions: Capitalizing on the above discussions, the paper proposes a SR-based framework, referred to as Multi-Agent Adaptive Kalman Filtering-based Successor Representation (MAKF-SR). The proposed framework can adapt quickly to the changes in a multi-agent environment faster than the MF methods and with the lower computational cost compared to MB algorithms. Within the proposed MAKF-SR framework, we model the SR learning procedure into a filtering problem using Kalman Temporal Difference (KTD) formulation [24].Intuitevely speaking, the idea is to benefit from intrinsic advantages of Kalman filtering, i.e., online second order learning, uncertainty estimation and non-stationarity handling. RBF estimators are then utilized within the MAKF-SR framework to project continuous states into feature vectors and to represent the reward function as a linear function of extracted feature vectors. Then to estimate the localized reward functions, we resorted to Multiple Model Adaptive Estimation (MMAE) techniques. More specifically, MMAE is used to tackle the lack of information about the measurement mapping function and measurement noise covariance of the reward weight. The proposed MAKF-SR framework is evaluated through a comprehensive set of experiments and simulations illustrating superior performance compared to its counterparts.

2. PROBLEM FORMULATION

Within the context of single-agent RL algorithms, the agent selects potential actions from the action set $\mathcal{A}$ by following an optimal policy in such a way that its cumulative reward would be maximized over time. More specifically, at time step k, the agent takes an action $a_k \in \mathcal{A}$ based on the policy π_k given its current state of $\boldsymbol{s}_k \in \mathcal{S}$. The environment then responds to this action by taking the system to state $\boldsymbol{s}_{k+1} \in \mathcal{S}$ with the transition probability of $P(\boldsymbol{s}_{k+1}|\boldsymbol{s}_k, a_k) \in \mathcal{P}_a$ and returning a reward $r_k \in \mathcal{R}$ to the agent. The agent starts at an initial state $\boldsymbol{s}_0$ and continues its interactions until the terminal state $\boldsymbol{s}_T$ is reached. The rewards are discounted using discount factor $\gamma \in (0, 1)$ to control the importance of the future rewards versus the immediate ones. The 5-tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}_a, \mathcal{R}, \gamma\}$ defines Markov Decision Process (MDP), which provides a mathematical framework for modeling the RL tasks. The objective in a RL problem is to learn the policy, which maps states into actions while maximizes the expected sum of discounted rewards over the future states. This is known as the optimal policy denoted by π^* [25]. In this regard, the state-action value function, $Q_\pi(\boldsymbol{s}, a)$, is typically used, i.e.,

$$Q_\pi(\boldsymbol{s}, a) = \mathbb{E}\left\{\sum_{k=0}^{T} \gamma^k r_k | \boldsymbol{s}_0 = \boldsymbol{s}, a_0 = a, a_k = \pi(\boldsymbol{s}_k)\right\}, \quad (1)$$

where $\mathbb{E}\{\cdot\}$ represents the expectation function. Temporal Difference (TD) learning is obtained via recursive upgrade of state-action value function, i.e.,

$$Q_\pi^{\text{new}}(\boldsymbol{s}_k, a_k) = Q_\pi^{\text{old}}(\boldsymbol{s}_k, a_k) + \alpha\Big(r(\boldsymbol{s}_k, a_k) + \gamma\, Q_\pi(\boldsymbol{s}_{k+1}, a_{k+1}) - Q_\pi^{\text{old}}(\boldsymbol{s}_k, a_k)\Big), \quad (2)$$

where $0 < \alpha \leq 1$ represents the learning rate. The current policy is used to select actions at the learning stage. After convergence, the optimal policy can be used as $a_k = \arg\max_{a \in \mathcal{A}} Q_{\pi^*}(\boldsymbol{s}_k, a)$. In the multi-agent scenarios, each agent has its localized state, action, and observations at each time. Consider a multi-agent scenario with N number of agents. We use superset $\mathbb{S} = \{\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(N)}\}$ to represent set of local states. Similarly, $\mathbb{A} = \{\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(N)}\}$ represents a superset consisting of local action sets $\mathcal{A}^{(i)}$, for $(1 \leq i \leq N)$. Finally, we define superset $\mathbb{Z} = \{\mathcal{Z}^{(1)}, \ldots, \mathcal{Z}^{(N)}\}$, which includes local observation sets. Consider the case where Agent i, for $(1 \leq i \leq N)$, uses stochastic policy $\pi^{(i)} : \mathcal{Z}^{(i)} \times \mathcal{A}^{(i)} \rightarrow [0, 1]$ to take actions within the environment. Each agent makes decisions based on the transition function $T : \mathbb{S} \times \mathcal{A}^{(1)} \times, \ldots, \times \mathcal{A}^{(N)} \rightarrow \mathbb{S}^2$ and according to its actions makes a new observation and receives a reward $r^{(i)} : \mathbb{S} \times \mathcal{A}^{(i)} \rightarrow \mathbb{R}$. The objective of each agent is to maximize the expected return $R^{(i)}$ defined as $R^{(i)} = \sum_{t=0}^{T} \gamma^t (r^{(i)})^t$, where T is the termination time and γ is the discount factor.

2.1. Multi-Agent Successor Representation (SR)

The SR approaches estimate the expected discounted future state occupancy of state $\boldsymbol{s}'^{(i)}$ given the initial state $\boldsymbol{s}^{(i)}$, and initial action $a^{(i)}$ according the current policy $\pi^{(i)}$, i.e.,

$$\boldsymbol{M}_{\pi^{(i)}}(\boldsymbol{s}^{(i)}, \boldsymbol{s}'^{(i)}, a^{(i)}) = \mathbb{E}\left[\sum_{k=0}^{T} \gamma^k \mathbb{1}[\boldsymbol{s}_k^{(i)} = \boldsymbol{s}'^{(i)}] | \boldsymbol{s}_0^{(i)} = \boldsymbol{s}^{(i)}, a_0^{(i)} = a^{(i)}\right], \quad (3)$$

where $\mathbb{1}\{\cdot\} = 1$ if $\boldsymbol{s}_k^{(i)} = \boldsymbol{s}'^{(i)}$ and 0 otherwise. In a discrete state space, SR is a $N_{\boldsymbol{s}^{(i)}} \times N_{\boldsymbol{s}^{(i)}}$ matrix, where $N_{\boldsymbol{s}^{(i)}}$ is the number of sates of the Agent i, for $(1 \leq i \leq N)$. Similar to Eq. (2), the SR can be updated in an on-policy recursive form as follows

$$\boldsymbol{M}_{\pi^{(i)}}^{\text{new}}(\boldsymbol{s}_k^{(i)}, \boldsymbol{s}'^{(i)}, a_k^{(i)}) = \boldsymbol{M}_{\pi^{(i)}}^{\text{old}}(\boldsymbol{s}_k^{(i)}, \boldsymbol{s}'^{(i)}, a_k^{(i)}) + \alpha\Big(\mathbb{1}[\boldsymbol{s}_k^{(i)} = \boldsymbol{s}'^{(i)}] + \gamma \boldsymbol{M}_{\pi^{(i)}}(\boldsymbol{s}_{k+1}^{(i)}, \boldsymbol{s}'^{(i)}, a_{k+1}^{(i)}) - \boldsymbol{M}_{\pi^{(i)}}^{\text{old}}(\boldsymbol{s}_k^{(i)}, \boldsymbol{s}'^{(i)}, a_k^{(i)})\Big) \quad (4)$$

Once the SR has been estimated at each step, the state-action value function in Eq. (1) can be expressed as the inner product of the SR and the estimated immediate reward as follows

$$Q_{\pi^{(i)}}(\boldsymbol{s}_k^{(i)}, a_k^{(i)}) = \sum_{\boldsymbol{s}'^{(i)} \in \mathcal{S}^{(i)}} \boldsymbol{M}(\boldsymbol{s}_k^{(i)}, \boldsymbol{s}'^{(i)}, a_k^{(i)}) R^{(i)}(\boldsymbol{s}'^{(i)}, a_k^{(i)}), \quad (5)$$

Eq. (5) demonstrates the most important feature of SR-based frameworks, which is reconstructing the state-action value function straightforwardly given changes in the reward function. Next, we present the proposed MAKF-SR framework based on the developed multi-agent formulation discussed in this section.

3. THE PROPOSED MAKF-SR FRAMEWORK

Exact computation of the SR and the reward function is, typically, not possible within the multi-agent settings as we are dealing with a large number of states. In addition, in applications such as smart public governance of SCs and demand-response to control building energy systems, the state-space is continuous. Therefore, we consider approximating the SR and reward function via basis functions by mapping State $\boldsymbol{s}^{(i)}$ to a N-dimensional state feature vector $\boldsymbol{\phi}(\boldsymbol{s}^{(i)})$. Given a pair $(\boldsymbol{s}^{(i)}, a^{(i)})$, the state feature vector $\boldsymbol{\phi}(\boldsymbol{s}^{(i)})$ is used in the corresponding slot for that action a while setting basis function values for the rest of the actions to zero. The generated vector is referred to as state-action feature vector $\boldsymbol{\phi}(\boldsymbol{s}^{(i)}, a^{(i)})$, where $\boldsymbol{\phi} : \mathcal{A}^{(i)} \times \mathcal{S} \rightarrow \mathbb{R}^{N \times D^{(i)}}$), and $D^{(i)}$ is the total number of actions for the i^{th} agent. For the state-action feature vector $\boldsymbol{\phi}(\boldsymbol{s}^{(i)}, a^{(i)})$, a

feature-based SR, which encodes the expected feature occupancy of the features, is defined as follows

$$\boldsymbol{M}_{\pi^{(i)}}(\boldsymbol{s}^{(i)},:,a^{(i)})=\mathbb{E}\left[\sum_{k=0}^{T}\gamma^{k}\boldsymbol{\phi}(\boldsymbol{s}_k^{(i)},a_k^{(i)})|\boldsymbol{s}_0^{(i)}=\boldsymbol{s}^{(i)},a_0^{(i)}=a^{(i)}\right] \tag{6}$$

We consider that the immediate reward function for pair $(\boldsymbol{s}^{(i)},a^{(i)})$ can be linearly factorized as

$$R^{(i)}(\boldsymbol{s}_k^{(i)},a_k^{(i)})\approx\boldsymbol{\phi}(\boldsymbol{s}_k^{(i)},a_k^{(i)})^T\boldsymbol{\theta}_k^{(i)}, \tag{7}$$

where $\boldsymbol{\theta}_k^{(i)}$ is the reward weight vector. The state-action value function (Eq. (5)), therefore, can be computed as follows

$$Q(\boldsymbol{s}_k^{(i)},a_k^{(i)})=\boldsymbol{\theta}^{(i)}{}_k^{T}\boldsymbol{M}(\boldsymbol{s}_k^{(i)},:,a_k^{(i)}). \tag{8}$$

The SR matrix $\boldsymbol{M}(\boldsymbol{s}_k^{(i)},:,a_k^{(i)})$ can be approximated as a linear function of the same feature vector as follows

$$\boldsymbol{M}_{\pi^{(i)}}(\boldsymbol{s}_k^{(i)},:,a_k^{(i)})\approx\boldsymbol{M}_k\,\boldsymbol{\phi}(\boldsymbol{s}_k^{(i)},a_k^{(i)}). \tag{9}$$

The TD learning of the SR then can be learned as follows

$$\begin{aligned}\boldsymbol{M}^{\text{new}}_{\pi^{(i)}}(\boldsymbol{s}_k^{(i)},:,a_k^{(i)})=\boldsymbol{M}^{\text{old}}_{\pi^{(i)}}(\boldsymbol{s}_k^{(i)},:,a_k^{(i)})+\alpha\big(\boldsymbol{\phi^{(i)}}(\boldsymbol{s}_k^{(i)},a_k^{(i)})\\ +\gamma\boldsymbol{M}_{\pi^{(i)}}(\boldsymbol{s}_{k+1}^{(i)},:,a_{k+1}^{(i)})-\boldsymbol{M}^{\text{old}}_{\pi^{(i)}}(\boldsymbol{s}_k^{(i)},:,a_k^{(i)})\big).\end{aligned} \tag{10}$$

Once the approximation structure of the SR and reward function have been defined, a suitable algorithm should be chosen to learn (estimate) the reward's weight vector $\boldsymbol{\theta}^{(i)}$ and the SR's weight matrix $\boldsymbol{M}$ for agent i. The proposed multi-agent MAKF-SR framework consists of the two main components, i.e., KTD-based weight SR learning and radial basis function update, which are detailed below.

3.1. KTD-based Weight SR Learning

The SR can be approximated from its one-step approximation using the TD method of Eq. (10). In this regard, the state-action feature vector at time step k can be considered as a noisy measurement as

$$\hat{\boldsymbol{\phi}}(\boldsymbol{s}_k^{(i)},a_k^{(i)})=\boldsymbol{M}^{\text{new}}(\boldsymbol{s}_k^{(i)},:,a_k^{(i)})-\gamma\boldsymbol{M}(\boldsymbol{s}_{k+1}^{(i)},:,a_{k+1}^{(i)})+\boldsymbol{n}_k^{(i)}, \tag{11}$$

where $\boldsymbol{n}_k^{(i)}$ is supposed to be a zero-mean Gaussian noise with the covariance of $\boldsymbol{R}_{\boldsymbol{M}}^{(i)}$. By putting Eqs. (9) and (11) together, the feature vector $\boldsymbol{\phi}(\boldsymbol{s}_k^{(i)},a_k^{(i)})$ can be approximated as

$$\hat{\boldsymbol{\phi}}(\boldsymbol{s}_k^{(i)},a_k^{(i)})=\boldsymbol{M}_k\underbrace{\left[\boldsymbol{\phi}(\boldsymbol{s}_k^{(i)},a_k^{(i)})-\gamma\boldsymbol{\phi}(\boldsymbol{s}_{k+1}^{(i)},a_{k+1}^{(i)})\right]}_{\boldsymbol{g}_k^{(i)}}+\boldsymbol{n}_k^{(i)} \tag{12}$$

We map matrix $\boldsymbol{M}_k$ to the column vector $\boldsymbol{m}_k^{(i)}$ by stacking its columns of $\boldsymbol{M}_k$. Based on the vec-trick property of Kronecker product, Eq. (12) can be rewritten as

$$\hat{\boldsymbol{\phi}}(\boldsymbol{s}_k^{(i)},a_k^{(i)})=(\boldsymbol{g}^{(i)}{}_k^{T}\otimes\boldsymbol{I})\boldsymbol{m}_k^{(i)}+\boldsymbol{n}_k^{(i)}, \tag{13}$$

where $\otimes$ denotes the Kronecker product and $\boldsymbol{I}$ is the identity matrix. Eq. (13) defines the observation of the system $(\boldsymbol{\phi}(\boldsymbol{s}_k^{(i)},a_k^{(i)}))$ as a linear function of vector $\boldsymbol{m}_k^{(i)}$, which is to be estimated. We consider $\boldsymbol{m}_{k+1}^{(i)}=\boldsymbol{m}_k^{(i)}+\boldsymbol{\mu}_k^{(i)}$ to form a complete state-space model for Kalman Filter (KF) implementation. In the dynamic model the forcing term $\boldsymbol{\mu}_k^{(i)}$ is supposed to be a zero-mean Gaussian noise with the variance of $\boldsymbol{Q}_M$. At each iteration, $\boldsymbol{m}_k^{(i)}$ and its covariance matrix $\boldsymbol{P}^{(i)}_{\boldsymbol{m}^{(i)},k}$ are estimated and updated using the actual measurement of the system. Matrix $\boldsymbol{M}_k$ is then reconstructed via reshaping the estimated vector $\boldsymbol{m}_k^{(i)}$ into a $(L\times L)$ matrix. The state-action value function for each pair $(\boldsymbol{s}_k^{(i)},a_k^{(i)})$, therefore, is computed through Eq. (8).

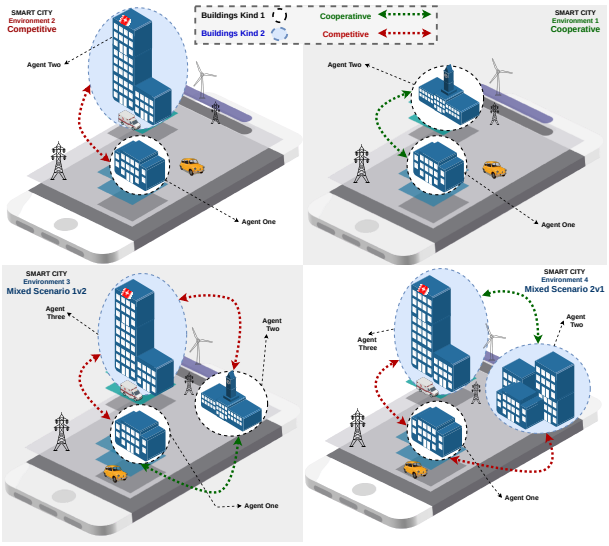

Fig. 1. Four scenarios in multi-agent OpenAI gym environment.

3.2. Radial Basis Function Update

As stated previously, the measurement mapping function needs to be properly set up in KF-based estimations as one of the most important parameters. Such a priori knowledge, however, is usually not available, and consequently it has to be adapted to its correct value. The vector of basis functions $\boldsymbol{\phi}(\boldsymbol{s}_k)$ is formed as follows

$$\boldsymbol{\phi}(\boldsymbol{s}_k^{(i)})=\left[\phi_1(\boldsymbol{s}_k^{(i)}),\phi_2(\boldsymbol{s}_k^{(i)}),\ldots,\phi_{N-1}(\boldsymbol{s}_k^{(i)}),\phi_N(\boldsymbol{s}_k^{(i)})\right]^T \tag{14}$$

Each basis function is selected as a radial basis function, i.e., $\phi_n(\boldsymbol{s}_k^{(i)})=\exp\{\frac{-1}{2}(\boldsymbol{s}_k^{(i)}-\boldsymbol{\mu}_n^{(i)})^T{\boldsymbol{\Sigma}_n^{(i)}}^{-1}(\boldsymbol{s}_k^{(i)}-\boldsymbol{\mu}_n^{(i)})\}$, where $\boldsymbol{\mu}_n^{(i)}$ and $\boldsymbol{\Sigma}_n^{(i)}$ are the mean and covariance of the radial basis function. In this work, the state-action feature vector $\boldsymbol{\phi}(\boldsymbol{s}_k^{(i)},a_k^{(i)}=a_d^{(i)})$, for $(1\leq d\leq D)$, is assumed to be generated from $\boldsymbol{\phi}(\boldsymbol{s}_k^{(i)})$ by placing this state feature vector in the corresponding spot for action $a_k^{(i)}$ while the feature values for the rest of the actions are set to zero, i.e., $\boldsymbol{\phi}(\boldsymbol{s}_k^{(i)},a_k^{(i)})=[0,\ldots 0,\phi_1(\boldsymbol{s}_k^{(i)}),\ldots,\phi_N(\boldsymbol{s}_k^{(i)}),0,\ldots 0,]^T$. The restricted gradient descent (RGD) method proposed in [26] is adopted to adapt these parameters. The goal is to minimize the loss function $(L_k^{(i)})$, i.e., $L_k^{(i)}=(\boldsymbol{\phi}^T(\boldsymbol{s}_k^{(i)},a_k)\,\boldsymbol{\theta}_k^{(i)}-r_k^{(i)})^2$. The gradient of the loss function with respect to the parameters of the RBFs is calculated using the chain rule resulting in the following formulations for the mean and covariance of RBFs

$$\boldsymbol{\mu}_n^{(i)}=\boldsymbol{\mu}_n^{(i)}-2\lambda_{\boldsymbol{\mu}^{(i)}}\left(L_k^{(i)}\right)^{\frac{1}{2}}\boldsymbol{\theta}_k^{(i)^T}\boldsymbol{\phi}(\boldsymbol{\Sigma}^{(i)})^{-1}(\boldsymbol{s}_k^{(i)}-\boldsymbol{\mu}_n^{(i)}) \tag{15}$$

$$\begin{aligned}\boldsymbol{\Sigma}_n^{(i)}=\boldsymbol{\Sigma}_n^{(i)}-2\lambda_{\boldsymbol{\Sigma}^{(i)}}\left(L_k^{(i)}\right)^{\frac{1}{2}}\boldsymbol{\theta}_k^{(i)^T}\boldsymbol{\phi}(\boldsymbol{\Sigma}^{(i)})^{-1}\\ \times(\boldsymbol{s}_k^{(i)}-\boldsymbol{\mu}_n^{(i)})(\boldsymbol{s}_k^{(i)}-\boldsymbol{\mu}_n^{(i)})^T{\boldsymbol{\Sigma}_n^{(i)}}^{-1},\end{aligned} \tag{16}$$

where $\lambda_{\boldsymbol{\mu}^{(i)}}$ and $\lambda_{\boldsymbol{\Sigma}^{(i)}}$ are the adaptation rates.

4. EXPERIMENTAL RESULTS

In this section, the proposed MAKF-SR framework is evaluated and compared with the most popular state-of-the-art multi-agent

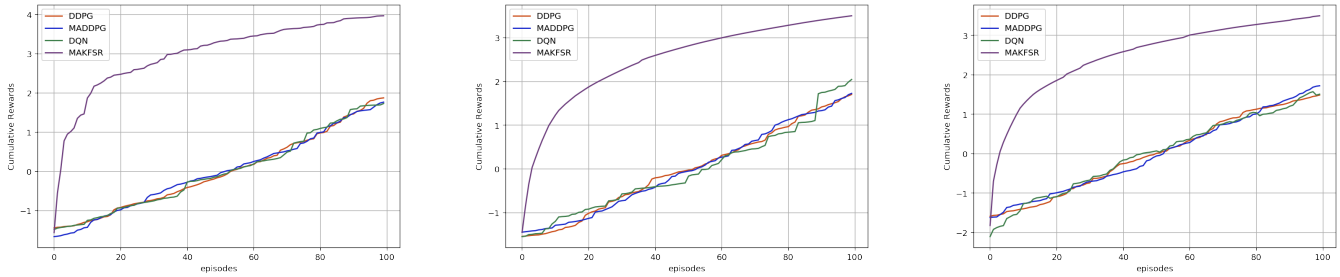

Fig. 2. Average rewards after 100 episodes: (a) Cooperative Scenario. (b) Competitive Scenario. (c) Predator-Prey Scenario (i).

Table 1. Average loss per episode for different environments and platforms.

Environment	MAKF-SR	MADDPG	DDPG	DQN
Simple Cooperation	8.93	9649.84	10561.16	10.93
Simple Competition	0.43	10158.18	10710.37	107.39
Predator-Prey 1v2	0.0005	6816.34	6884.33	8.21
Predator-Prey 2v1	8.87	7390.18	6882.2	10.24

Table 2. Average received reward by the agents per episode.

environment	MAKF-SR	MADDPG	DDPG	DQN
Simple Cooperation	-16.76	-69.28	-66.29	-39.96
Simple Competition	-0.778	-63.30	-61.34	-1.49
Predator-Prey 1v2	-0.0916	-46.17	-20.53	-3.451
Predator-Prey 2v1	-0.081	-55.69	-49.41	-4.32

RL algorithms, i.e., DQN, DDPG, and MADDPG Algorithms. To demonstrate the efficacy of the proposed MAKF-SR framework in the learning process and its rapid adaptation to the reward changes, the following different environments are considered: *(a) Cooperative Scenario*, where two cooperative agents work together in the SC to prevent interception and maximize their cumulative rewards and the number of steps achieved in an episode; *(b) Competitive Scenario*, where two competitive agents work against each other, and; *(c) Predator-Prey Scenario*: Two more complex predator-prey environments are designed, i.e., (i) Predator-Prey 1v2, where one slower predator (adversary) is perusing two faster preys, and; (ii) Predator-Prey 2v1, one faster prey is escaping from two slower predators. These environments are created in the multi-agent extension of the OpenAI gym benchmark, which is a two-dimensional world with continuous space [27]. There is no stable equilibrium in the environment, pushing the agents to become smarter despite their present level of intelligence. In these environments, agents take actions in each step, and the platform provides information, including given rewards and the observation of the agents on the new states. In all of these environments, agents' observations consist of local information and the relative positions and velocities with other agents allowing them to know other agents' behaviors to maximize the returned reward. Fig. 1 shows examples of the environment used to evaluate the results and performance of the proposed MAKF-SR.

In the multi-agent scenarios, agents' observation size differs based on the type (i.e., Predator and Prey) and number of agents in the environment. Feature vector size, based on the number of the actions (five different actions considered in the experiments) is considered to be 50 regardless of the agent's observation size (the number of local and global information each agent is receiving). In all the environments, mean and covariance of the RBFs for all the agents are initialized randomly. For instance, in the Predator-Prey scenario, while 2 preys are playing with one predator (adversary), the size of the mean for the predator is 9×12 while this size of the other agents is 9×10. 12 and 10 are the sizes of the observation (state size) of each agent, respectively, and 9 is the number of RBFs used along with the bias parameters to build the feature vector. The initial values of $\lambda_{\boldsymbol{\mu}}$ and $\lambda_{\boldsymbol{\Sigma}}$ are selected as 200 and 100, respectively, to keep the system stable. The discount factor is selected as 0.95, (i.e., $\gamma = 0.95$). The covariance of the noise is defined to be a small value ($\boldsymbol{Q}_k = 10^{-7}\boldsymbol{I}_{50}$), and the initial value of the transition matrix is set to be $\boldsymbol{F} = \boldsymbol{I}_{50}$. The measurement noise variance candidates for the reward's weight estimation is selected from the following set, $R^{(i)} \in \{0.01, 0.1, 0.5, 1, 5, 10, 50, 100\}$. Initial weights also defined to be zero, i.e., $\boldsymbol{\theta}_0 = \mathbf{0}_{50}$ and the error covariance for all the environments are selected to be $\boldsymbol{P}_{\boldsymbol{\theta},0} = 10\boldsymbol{I}_{50}$. Each experiment starts randomly with the aforementioned initial parameters, and all the experiments are performed for more than 100 episodes in the learning stage and 100 episodes during the test stage. Tables 1-3 compare the results achieved from different scenarios. The results are averaged over multiple realizations leveraging Monte Carlo sampling. As it can be seen in Table 1, the average loss in the proposed MAKF-SR is significantly better than that of the other approaches. Such an excellent performance can be seen across all the scenarios as shown in Fig. 2. As can be seen, other approaches cannot provide that level of performance and stability achieved by MAKF-SR with the minimal number of training episodes in these experiments. It is worth mentioning that the other three main approaches can achieve better efficiency with a much higher amount of training experience, which leads to consuming a large amount of memory to save the batches of the information. Table 2, represents the returned immediate rewards for all scenarios and approaches deployed in this work. It is worth mentioning that the average number of steps taken by all the agents in the defined environments is also represented in Table 3, showing remarkable performance of the MAKF-SR framework in contrast to its counterparts. Finally, the computational cost of the MAKF-SR is much lower than its counterparts.

Table 3. Average steps taken by agents per episode.

environment	MAKF-SR	MADDPG	DDPG	DQN
Simple Cooperation	14.03	6.43	6.78	12.06
Simple Competition	17.59	7.36	7.18	11.98
Predator-Prey 1v2	14.78	6.21	7.69	10.02
Predator-Prey 2v1	9.94	6.25	7.12	8.46

5. CONCLUSION

The paper proposed a novel KF-based framework, referred to as the MAKF-SR, to learn goal reaching behavior in multi-agent RL problems. The MAKF-SR estimates (learns) the value function by computing the inner product of the SR and immediate reward's weight estimates. The proposed MAKF-SR algorithm was evaluated based on four environments in the multi-agent extension of the OpenAI gym benchmark. Based on the achieved results, the proposed algorithm outperformed DQN, DDPG and MADDPG methods in terms of cumulative reward, speed of the value function convergence to reward changes, sample efficiency, and cumulative steps of the agents in the environment.

6. REFERENCES

[1] P. Spachos and K. Plataniotis, "BLE Beacons in the Smart City: Applications, Challenges, and Research Opportunities," *IEEE Internet of Things Magazine*, vol. 3, no. 1, pp. 14-18, March 2020,

[2] Z. Nagy, J. Y. Park, and J. R. Vazquez-Canteli. "Reinforcement Learning for Intelligent Environments: A Tutorial," *Handbook of Sustainable and Resilient Infrastructure*, 2018.

[3] J.R.Vazquez-Canteli, *et al.* , "Fusing TensorFlow with Building Energy Simulation for Intelligent Energy Management in Smart Cities," *Sustainable Cities Society 45*, 243-257, 2019.

[4] S. Sadowski, P. Spachos and K. N. Plataniotis, "Memoryless Techniques and Wireless Technologies for Indoor Localization with the Internet of Things," *IEEE Internet of Things Journal*, 2020. In Press.

[5] P. Spachos and K. N. Plataniotis, "BLE Beacons for Indoor Positioning at an Interactive IoT-Based Smart Museum," *IEEE Systems Journal*, vol. 14, no. 3, pp. 3483-3493, Sept. 2020.

[6] P. Malekzadeh, A. Mohammadi, M. Barbulescu and K. N. Plataniotis, "STUPEFY: Set-Valued Box Particle Filtering for Bluetooth Low Energy-Based Indoor Localization," *IEEE Signal Processing Letters*, vol. 26, no. 12, pp. 1773-1777, Dec. 2019.

[7] P. Malekzadeh, M. Salimibeni, A. Mohammadi, A. Assa and K. N. Plataniotis, "MM-KTD: Multiple Model Kalman Temporal Differences for Reinforcement Learning," *IEEE Access*, vol. 8, pp. 128716-128729, 2020.

[8] H. Hu, S. Song and C. L. P. Chen, "Plume Tracing via Model-Free Reinforcement Learning Method," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 8, pp. 2515-2527, Aug. 2019.

[9] H. K. Venkataraman, and P. J. Seiler, "Recovering Robustness in Model-Free Reinforcement Learning," *American Control Conference (ACC)*, Philadelphia, PA, USA, 2019, pp. 4210-4216.

[10] K. Kolomvatsos, and C. Anagnostopoulos, "Reinforcement Learning for Predictive Analytics in Smart Cities," *Informatics*, vol. 4, no. 3, 2017.

[11] J. R. Vazquez-Canteli, and Z. Nagy, "Reinforcement Learning for Demand Response: A Review of Algorithms and Modeling Techniques," *Applied Energy, 235, 1072–1089*, 2019.

[12] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information Theoretic MPC for Model-based Reinforcement Learning," *International Conference on Robotics and Automation (ICRA)*, 2017.

[13] E. Vértes,and M. Sahani, "A Neurally Plausible Model Learns Successor Representations in Partially Observable Environments," *Advances in Neural Information Processing Systems 32*, pp.13714-13724, 2019.

[14] S. Blakeman, and D. Mareschal, "A Complementary Learning Systems Approach to Temporal Difference Learning," *Neural Networks*, vol. 122, 2020, pp. 218-230.

[15] A. Ducarouge, O. Sigaud, "The Successor Representation as a Model of Behavioural Flexibility," *Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes (JFPDA)*, 2017.

[16] J. P. Geerts, K. L. Stachenfeld, and N. Burgess, "Probabilistic Successor Representations with Kalman Temporal Differences," *arXiv preprint arXiv:1910.02532*, 2019.

[17] Kulkarni, Tejas D., Ardavan Saeedi, Simanta Gautam, and Samuel J. Gershman, "Deep Successor Reinforcement Learning," *arXiv preprint*, arXiv:1606.02396, 2016.

[18] M. Riedmiller, "Neural Fitted Q Iteration-first Experiences with a Data Efficient Neural Reinforcement Learning Method," *European Conference on Machine Learning*, Springer, 2005, pp. 317-328.

[19] Y. Tang, H. Guo,T. Yuan, X. Gao, X. Hong, Y. Li, J. Qiu, Y. Zuo, and J. Wu, "Flow Splitter: A Deep Reinforcement Learning-Based Flow Scheduler for Hybrid Optical-Electrical Data Center Network," *IEEE Access*, vol. 7, pp.129955-129965, 2019.

[20] M. Kim, S. Lee, J. Lim, J. Choi, and S.G. Kang, "Unexpected Collision Avoidance Driving Strategy Using Deep Reinforcement Learning," *IEEE Access*, vol. 8, pp. 17243-17252, 2020.

[21] J. Xie, Z. Shao, Y. Li, Y. Guan, and J. Tan, "Deep reinforcement learning with optimized reward functions for robotic trajectory planning," *IEEE Access*, vol. 7, pp. 105669-105679, 2019.

[22] S. Haykin, "Neural Networks: A Comprehensive Foundation," *Prentice Hall PTR*, 1994.

[23] G. Konidaris, S. Osentoski, and P. S. Thomas, "Value Function Approximation in Reinforcement Learning using the Fourier Basis," *AAAI*, vol.6, 2011, p. 7.

[24] M. Geist and O. Pietquin, "Kalman Temporal Differences," *Journal of Artificial Intelligence Research*, vol. 39, pp. 483-532, 2010.

[25] R. S. Sutton, A. G. Barto, F. Bach *et al.*, "Reinforcement Learning: An Introduction," *MIT Press*, 1998.

[26] A. d. M. S. Barreto and C. W. Anderson, "Restricted Gradient-descent Algorithm for Value-function Approximation in Reinforcement Learning," *Artificial Intelligence*, vol. 172, no. 4-5, pp. 454-482, 2008.

[27] I. Mordatch, P. Abbeel, "Emergence of grounded compositional language in multi-agent populations." *Proc. AAAI Conference of Artificial Intelligence*, (2018).