Information Sciences 608 (2022) 63–80

Contents lists available at ScienceDirect

Information Sciences

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

Graph convolutional recurrent networks for reward shaping in reinforcement learning

Hani Sami [a], Jamal Bentahar [a], Azzam Mourad [b,c], Hadi Otrok [d], Ernesto Damiani [d,*]

[a] Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada
[b] Department of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon
[c] Division of Science, New York University, Abu Dhabi, United Arab Emirates
[d] Center of Cyber-Physical Systems (C2PS), Department of EECS, Khalifa University, Abu Dhabi, United Arab Emirates

ARTICLE INFO

Article history:
Received 3 February 2022
Received in revised form 12 April 2022
Accepted 11 June 2022
Available online 18 June 2022

Keywords:
Reinforcement Learning
Reward Shaping
GCRN
Augmented Krylov
Look-Ahead Advice
Atari
MuJoCo

ABSTRACT

In this paper, we consider the problem of low-speed convergence in Reinforcement Learning (RL). As a solution, various potential-based reward shaping techniques were proposed to form the potential function. Learning a potential function is still challenging and comparable to building a value function from scratch. In this work, our main contribution is proposing a new scheme for reward shaping, which combines (1) the Graph Convolutional Recurrent Networks (GCRN), (2) augmented Krylov, and (3) look-ahead advice to form the potential function. We propose an architecture for GCRN that combines Graph Convolutional Networks (GCN) to capture spatial dependencies and Bi-Directional Gated Recurrent Units (Bi-GRUs) to account for temporal dependencies. Our definition of the loss function of GCRN incorporates the message passing technique of the Hidden Markov Models (HMM). Since the transition matrix of the environment is hard to compute, we use the Krylov basis to estimate the transition matrix, which outperforms the existing approximation bases. Unlike existing potential functions that only rely on states to perform reward shaping, we use both the states and actions through the look-ahead advice mechanism to produce more precise advice. Our evaluations conducted on the Atari 2600 and MuJoCo games show that our solution outperforms the state-of-the-art that utilizes GCN as the potential function in most games in terms of the learning speed while reaching higher rewards.

1. Introduction

A Reinforcement Learning (RL) agent recognizes its position in the environment through a state defined by a Markov Decision Process (MDP). From a given state, the agent decides on the action that best maximizes the cumulative rewards. The action selection is based on a value function, which either considers states only, or states and actions. The objective of the value function is to maximize the cumulative rewards by selecting the best action for each state. The reward function is defined through the MDP and is used to evaluate the actions taken. The value function is updated iteratively using an RL algorithm, such as Q-learning, SARSA(λ), Deep Q-Network (DQN), etc [18]. RL techniques are usually time consuming; therefore, speeding the learning process is crucial for practical consideration of RL solutions in various applications. For this purpose, reward shaping techniques were invented to alter the original reward function definition, thus helping the agent

* Corresponding author.

https://doi.org/10.1016/j.ins.2022.06.050
0020-0255/

reach the optimal policy [17]. Different techniques exist for designing a reward shaping function. In this work, we focus on potential-based reward shaping [17] because it guarantees invariance with respect to the optimal policy for solving an MDP. Building the potential function is not straightforward for complex environments [1]. Therefore, a potential-based reward shaping solution is still an open problem.

Our first step in this paper is defining the potential function using the probabilistic inference view of RL. Using the message passing technique of Hidden Markov Models (HMM) [31], we are able to calculate the probability of the agent belonging to an optimal trajectory given the state and action [12], which we consider as an effectiveness signal for accelerating the learning. We also claim that predicting the reward shaping value at the next timestep helps accelerate further the learning. Therefore, we propose using a Recurrent Neural Network (RNN) that takes as input the state transitions from time $t = 1$ to a timestep T and predicts the reward shaping value at $T + 1$. The loss function of the RNN uses the actual labels of the reward shaping values that are calculated using the message passing technique.

The message passing technique is used to calculate the forward and backward messages. As discussed in Section 2.3, calculating these messages is computationally expensive, especially when dealing with a large graph of states and transitions. To overcome this issue, we propose using a GCRN model that combines a Graph Convolutional Network (GCN) and an RNN, where the GCN is responsible for computing the message passing, while the RNN predicts the next reward shaping value. In other words, the GCRN is capable of studying the spatio-temporal dependencies using GCN and RNN respectively. Due to the recursive nature of GCN, it is straightforward to perform message passing by propagating information about rewarding states between the neighbors using a filter matrix. In various research proposals that use GCN, the graph Laplacian corresponds to the filter that represents the connection of nodes in the graph or approximates the transition matrix. For instance, the authors in [12] use the graph Laplacian to perform reward shaping using GCN, while assuming that the value function is smooth over the induced MDP graph. This smoothness is defined using the Sobolov norm [3]. This assumption results in a margin of error for approximating the value function that can be reduced by computing the Krylov basis [19]. In this paper, we also argue that if the graph Laplacian affects the accuracy of the value function approximation (VFA), it definitely affects using GCN for reward shaping with the filter of the graph Laplacian. We elaborate on the drawbacks of using the graph Laplacian instead of the Krylov basis in Section 2.6.

In this work, we propose using GCRN with the Krylov basis as the filter of GCN to produce reward shaping values. The actual labels in the GCRN training are resulted from calculating the forward and backward messages. Due to the computation complexity induced for calculating the message passing, as well as the difficulty of retrieving the full transition matrix of the environment, we train GCRN on a sample of the agent transitions. When using a sub-graph of transitions, the learning accuracy of GCN is not affected, because GCN propagates information in the graph and was commonly used to perform semi-supervised learning [11]. In order to further improve the performance of GCRN, we also propose adapting an *optional* look-ahead advice mechanism to the training process. By using the look-ahead advice, the potential function is a function of the state and action, instead of only the state. Adapting this mechanism helps produce more precise advice at the action level [36].

The full process of our proposed GCRN scheme is presented in Fig. 1. We collect a sample of the agent transitions for each episode in the environment. Using these transitions, we form a sub-graph as shown in Fig. 1 (the red sub-graph). Afterwards, the input to GCN $X(S,A)$ is formed from the states $S \subset \mathscr{S}$ and actions $A \subseteq \mathscr{A}$ of the MDP, which considers incorporating the look-ahead advice mechanism. In addition, we build the Krylov basis K (i.e. the approximated transition matrix) using the augmented Krylov algorithm presented in Section 2.6. The input $X(S,A)$ and the approximated transition matrix K are employed to perform the GCN training. The output from GCN up to T, referred to as GC, is passed to an RNN network composed of Bi-directional Gated Recurrent Units (Bi-GRUs) to predict the reward shaping value at $T + 1$. Our choice of the GRUs over LSTM is due to the superior performance of GRUs in our evaluations. Moreover, we propose using bi-directional networks to predict the agent decision from the future as well as the post. Therefore, we claim that bi-directional networks further contribute to the novelty of our proposed scheme.

As a summary, our contributions in this paper are:

1. A novel scheme for potential-based reward shaping using a GCRN architecture that combines GCN and Bi-GRUs and benfits from the probability inference view of RL.
2. Train GCRN on a sample of transitions that predicts the reward shaping value at the next timestep.
3. Use the Krylov basis as a filter for the GCRN, which outperforms the graph Laplacian.
4. Adapt an optional look-ahead advice to produce more precise advice for the agent.

The rest of this paper is organized as follows. In Section 2, we present the background required to understand the core of our approach, and focus on the motivations and rationals behind our decisions for building GCRN. In Section 3, we present our proposed reward shaping solution that utilizes GCRN with the Krylov basis and look-ahead advice. The evaluations conducted on the Atari and MuJoCo games are presented in Section 4 compared to various baselines. Section 5 presents the related work. We conclude the paper in Section 6 with an open discussion about the limitations and future plan.

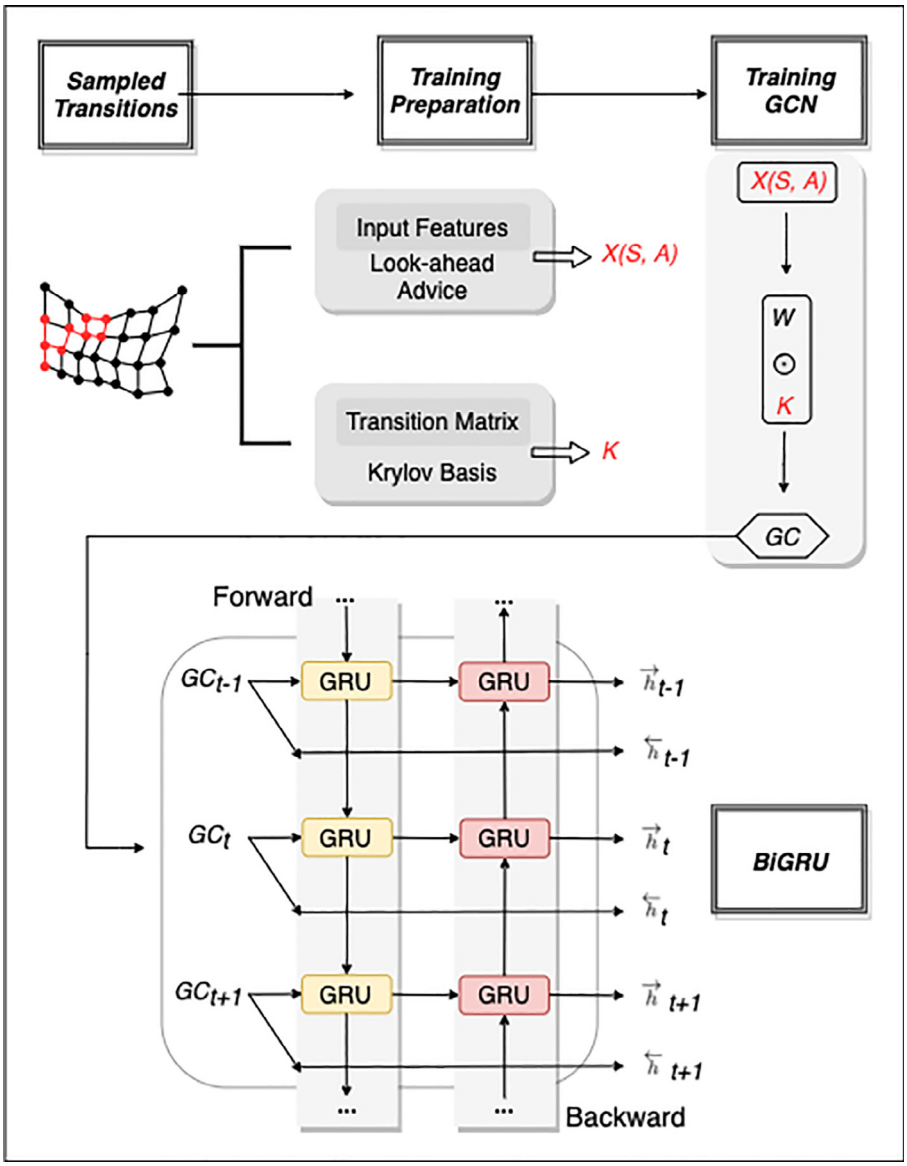


Fig. 1. Proposed Scheme Using GCRN.

2. Background and Motivation

After introducing the used techniques, we focus in this section on the rationals and motivations behind our design choices.

The RL environment is an MDP. MDP is defined by the tuple $(\mathscr{S}, \mathscr{A}, P, r, \gamma)$, where $\mathscr{S}$ is the state space, $\mathscr{A}$ is the action space, P is the probability transition matrix, r is the reward function, such that $r : \mathscr{S} \times \mathscr{A} \rightarrow \mathbb{R}$, and γ is the discount factor. RL uses an MDP to form a policy that maximizes the future discounted rewards induced by the action of each state. The policy is formed using the value function that can be calculated as follows [20]: $v = (I - \gamma P)^{-1} R = \sum_{i=0}^{\infty} (\gamma P)^i R$, where I is the identity matrix of an appropriate size, and R is the reward vector. The right side of the equality is extracted from the Neumann series expansion. The value function is calculated by iteratively adding the vectors extracted from the Neumann series.

2.1. Reward Shaping

The idea behind reward shaping is to apply expert knowledge for directing the reward function. This is done by appending scaler values, from the shaping function F, to the reward function so that it takes the following form:

$$R(S_t, A_t, S_{t+1}) = r(S_t, A_t) + F(S_t, S_{t+1}) \tag{1}$$

In this equation, $F(S_t, S_{t+1})$ is the shaping function that takes as input the states at t and $t+1$ from $\mathscr{S}$. Because it is important to preserve the policy when applying reward shaping, F is written as [36]:

$$F(S_t, S_{t+1}) = \gamma\phi(S_{t+1}) - \phi(S_t) \tag{2}$$

where ϕ is a potential-shaping function that outputs a scaler value. Existing solutions that form shaping functions either rely on human intervention [8], or cannot scale well in complex environments [6]. A more detailed explanation of previous approaches proposing shaping functions is described in Section 5. On the other hand, in [12], the authors propose the use of GCN as the shaping function, which resolves the issue of scalability. However, [12] does not consider the drawback of using the graph Laplacian as the GCN filter and neglects the importance of studying the temporal dependencies between states of the graph. Moreover, it is important to consider the option of adding the look-ahead feature when computing the reward shaping values using the potential function [36].

2.2. Look Ahead Advice

The look-ahead advice is a reward shaping technique proposed in [36] without altering the optimal policy. The shaping function usually takes the form of Eq. 2, allowing the agent to send feedback after observing the rewards of the states only. A more rigorous advice is given by building the reward shaping on both states and actions. Following the look-ahead advice proposed in [36], the shaping function takes the following form:

$$F(S_t, A_t, S_{t+1}, A_{t+1}) = \gamma\phi(S_{t+1}, A_{t+1}) - \phi(S_t, A_t) \tag{3}$$

where ϕ is a function of the state from $\mathscr{S}$ and action from $\mathscr{A}$ that produces a scaler value. Hence, the reward shaping function is expressed as follows:

$$R(S_t, A_t, S_{t+1}, A_{t+1}) = r(S_t, A_t) + F(S_t, A_t, S_{t+1}, A_{t+1}) \tag{4}$$

The look-ahead advice in the reward shaping decisions could further speed the learning process by augmenting the action values and affecting the action selection [36]. Therefore, we propose adding an optional look-ahead mechanism when training our GCRN.

2.3. HMM and Message Passing

The probability inference view of RL can be used to build a reward shaping function [12]. To apply probability inference on an MDP structure, the binary optimality variable O is introduced, where $O_t = 1$ means the state $S_t \in \mathscr{S}$ is optimal, and $O_t = 0$ otherwise. The distribution over the optimality variable is written as: $p(O_t = 1|S_t, A_t) = f(r(S_t, A_t))$, where f is a function that maps rewards to a probability value. This structure is presented in Fig. 2, and is analogous to HMMs. Thus, the VFA is obtained using the backward message passing function of the form: $\beta(S_t, A_t) = p(O_{t:T}|S_t, A_t)$ [31,37]. In this equation, $O_{t:T}$ represents the observation variables from time t until the end of the episode or the final state at T. The measure is proven to represent the probability inference view of RL, which is a generalization of the optimal control problem [21]. We use this quantity as a signal to speed the learning in the form of a potential function.

On the other hand, forward messages (α) are used to look back at the trajectory from $t = 0$ to $t - 1$. Thus, combining forward and backward messages allows the agent to view the outcome of the whole trajectory and induce its optimality, which is relevant for forming an effective reward shaping function. The forward message has the following form: $\alpha(S_t, A_t) = p(O_{t_0:t-1})|S_t, A_t)p(S_t, A_t)$. Thus, the combined messages is expressed as:

$$p(O_t|S_t, A_t) \simeq \alpha(S_t, A_t)\beta(S_t, A_t) \tag{5}$$

Based on [12], the potential function is expressed as $\phi_{\alpha,\beta} = \alpha(S_t, A_t)\beta(S_t, A_t)$.

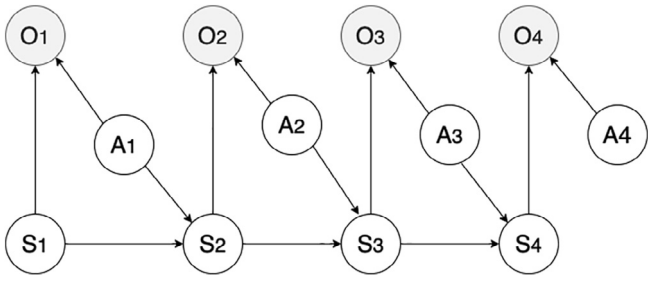


Fig. 2. Graphical Model From An MDP in RL.

2.4. Bi-Directional Gated Recurrent Units

The purpose of reward shaping is to speed the learning. Thus, predicting the reward shaping value of the next state can further improve the effectiveness of any potential function. To do this, we propose in this paper adding a bi-directional RNN that processes the reward shaping values from $t = 0$ to T to predict the shaping value at $T + 1$. In other words, we use the bi-directional RNN to learn the temporal dependencies between the states and transform it to a shaping value. The loss function in this network relies on using the message passing to compute the actual labels. In the sequel, we describe the mechanism behing the Bi-GRUs.

LSTM and GRUs are RNN variants that use the gated mechanism to expand the memory capabilities. Both LSTM and GRU excel in different domains. LSTMs are more complex and require more training time; however, GRUs have simpler structures with less number of parameters and less time to train. For these reasons, GRUs are more useful in the context of reward shaping. Besides, Bi-GRUs contain hidden layers for studying the sequences in forward and backward directions, which is better suited for our problem. The structure of the Bi-GRU layers is our scheme is shown in Fig. 1.

2.5. Graph Convolutional Networks (GCN)

Forward and backward messaging is only possible for some MDP problems where the state space and the size of trajectories are small. To make the message passing scalable, GCN is proposed as a function of reward shaping in [12]. GCN is first used to perform semi-supervised learning benefiting from its recursive nature and information propagation of labeled data to neighbors [11]. Transforming an MDP to a graph structure and applying GCN to perform message passing for reward shaping is possible. A basic form of two layers GCN is written as:

$$\phi_{GCN}(X) = Softmax(\mathrm{T}\, ReLu(\mathrm{T}XW_1)W_0) \tag{6}$$

where *ReLu* and *Softmax* are the activation functions, T is the GCN filter used to define the graph connections, X is the input matrix, and W_0, W_1 are the weights of each layer. As shown in Eq. 6, T is used in the GCN calculations for propagating information from the neighbors. In the context of reward shaping, T is an approximation of the transition matrix. In [12], the graph Laplacian is used as an approximate of the transition matrix. This is motivated by the proto-value function framework in the RL literature, which uses the graph Laplacian as a surrogate of the transition matrix due to its smooth approximate to the value function of an MDP following the spectral graph theory [3,14].

As argued in [19], the graph Laplacian is guaranteed to form an approximate estimate of the value function when assuming that the latter is smooth over the induced MDP graph. In order to resolve the issue of smoothness for the value function, diffusion wavelets can be used. However, diffusion wavelets require a matrix inverse operation, which is very expensive to compute. In addition, the work in [19] derived that using the actual transition matrix instead of the adjacency matrix leads to a smaller margin of error for forming the VFA. Henceforth, the use of the Krylov basis instead of the graph Laplacian for approximating the value function is a better option. The Krylov basis is formed using the augmented Krylov algorithm. In this paper, we use the Krylov basis for approximating the transition matrix for the GCN propagation. To predict the future reward shaping value, we propose the use of GCRN that combines GCN and Bi-GRUs to study the spatio-temporal dependencies between the nodes of the graph. In the next subsection, we provide more details about the augmented Krylov algorithm and the Krylov basis.

2.6. Augmented Krylov

In [19], the authors analyze the performance of the graph Laplacian for computing the function approximation. A normalized graph Laplacian is widely used and expressed as $L_c = I - \mathfrak{D}^{\frac{-1}{2}}\mathfrak{A}\mathfrak{D}^{\frac{-1}{2}}$, where $\mathfrak{D}$ and $\mathfrak{A}$ are the degree matrix and the adjacency matrix respectively, and I is the identity matrix. Besides, L_c is adapted in the original GCN work to resemble a filter connecting the nodes of the graph. Furthermore, GCN with L_c is used to generate the potential function for reward shaping in RL, where the adjacency matrix forming L_c is extracted from the graph of states [12]. In the same context, we argue in this work that the vectors extracted from the augmented Krylov algorithm, which we refer to as Krylov basis, outperforms the use of L_c as the GCN filter for forming the potential function. The results of the analysis in [19] show that the graph Laplacian forms an effective approximation when assuming that the adjacency matrix is symmetric (i.e., the value function is smooth over the graph of states). Such an approximation can be defined by considering the bottom eigenvectors of the graph Laplacian to approximate smooth functions with a low Sobolev norm. The final step is to aggregate these vectors to form the VFA.

According to [19], using the adjacency matrix to approximate the value function is not well motivated. In fact, using the actual transition matrix P^π leads to a smaller margin of errors when approximating the function compared to using the graph Laplacian. Using the spectral approximation of VFA, we can compute the top eigenvectors of P^π, which are considered as the approximation vectors. The algorithm for computing these vectors is referred to as the *weighted spectral method*. There is also another motivated base choice, which is using m vectors from the Neumann series, where m is the degree of the minimal polynomial of $(I - \gamma P)$ [10]. These extracted vectors form the Krylov space to approximate the VFA, denoted as $\mathscr{K}(P, r)$.

A practical implementation of the *weighted spectral method* faces issues in regards to the complexity of computing the eigenvectors. In addition, Jordan-decomposition should be calculated when dealing with a non-diagonalizable transition

matrix, which is time consuming, and there might be complex numbers in the computed eigenvalues and eigenvectors. In order to overcome these problems, the augmented Krylov algorithm is used, which combines the benefits of both approximations. In particular, the Krylov space captures the short-term behavior, while the top eigenvectors of the transition matrix capture the long-term behavior [19]. These features are well suited for our proposed GCRN for propagating the short and long term impacts of the messages in GCN. In augmented Krylov, the Krylov basis is built using the top eigenvectors of P followed by m vectors of the Neumann series.

In this paper, we argue that using the Krylov basis generated by the augmented Krylov algorithm generates a closer estimate to P compared to the graph Laplacian for training our GCRN. As stated earlier, the Augmented Krylov requires the transition matrix P, which is not accessible in our case. Thus, we use the transition matrix of a subset of the MDP graph to retrieve the Krylov basis and construct our approximation. We provide more details about this approximation in Section 3. To the best of our knowledge, there is no similar approach used in the context of approximating a transition matrix in the literature.

3. Proposed Scheme

In this section, we list the steps for constructing the potential function of our reward shaping solution using GCRN. We also present an algorithm to obtain the Krylov basis to approximate the transition matrix, and another algorithm showing the training of our proposed GCRN for potential-based reward shaping.

3.1. GCRN Configuration

In this paper, we propose GCRN, which combines GCN and RNN as shown in Fig. 1. GCRN learns both the complex spatial dependencies and dynamic temporal dependencies in the graph of states. Our GCRN network is composed of GCN followed by Bi-GRU layers, where the vanilla GRU is used and the hidden layers are unchanged. However, the input to the RNN is the output of the preceding GCN. Noting that the input layer of the GCN accepts states and actions as part of implementing the look-ahead advice mechanism. Furthermore, the output of our GCRN is a probability distribution considered as the potential function output. The outputs of a GRU, reset, and update gates are computed as follows:

$$\begin{aligned}
&\mathbf{r}_t = \sigma(W_r X_t + U_r h_{t-1} + b_r)\\
&g_t = \sigma(W_g X_t + U_g h_{t-1} + b_g)\\
&\tilde{h}_t = tanh[W_h X_t + U_h + \mathbf{r}_t \odot h_{t-1}) + b_h]\\
&h_t = (1 - g_t) \odot h_{t-1} + g_t \odot \tilde{h}_t
\end{aligned} \tag{7}$$

In these equations, $\mathbf{r}, g$, and h are the reset, update, and the output gates respectively. In addition, X_t is the input at time t, σ is the logistic sigmoid activation, and $\odot$ is the Hadamard product. W_r, W_g, W_h, U_r, U_g, and U_h are the weight matrices. b_r, b_g, and b_h are the synthesis of bias vectors for the input X.

A Bi-GRU is composed of forward backward GRUs. The output vectors of the forward and backward GRUs are concatenated to get the final result. In our implementation, a forward pass in GCRN is expressed as follows:

$$\begin{aligned}
&GCN(X_t) = \mathrm{K}\, ReLu(\mathrm{K} X_t W_1) W_0\\
&\vec{h_t} = \mathrm{GRU}_{fwd}\left(GC(X_t), \vec{h_{t-1}}\right)\\
&\overleftarrow{h_t} = \mathrm{GRU}_{bwd}\left(GC(X_t), \overleftarrow{h_{t+1}}\right)\\
&\phi_{GCRN}(X_t) = LogSoftmax\left(\vec{h_t} \oplus \overleftarrow{h_t}\right)
\end{aligned} \tag{8}$$

where GRU_{fwd} and GRU_{bwd} are computed following the steps in Eq. 7. In this formulation, the output of GCRN as a potential function is expressed as ϕ_{GCRN}. X_t is the input matrix of size $(||S_t|| + ||A_t||)$, where $||S_t||$ and $||A_t||$ are the number of features in the state and action respectively; and K is the Krylov basis, which is computed through the augmented Krylov. The output of GCRN resembles the scaler value that gets appended to the original reward value of the transition being studied.

3.2. Loss Function

Updating GCRN for reward shaping entails using the message passing technique for calculating the network loss using the predicted and actual labels. The standard GCRN loss function is composed of the base and recursive cases in order to reflect the message passing mechanism as follows:

$$\mathscr{L} = \mathscr{L}_0(\mathbf{S}, \mathbf{A}) + \eta \mathscr{L}_{rec}(S, A) \tag{9}$$

where $\mathbf{S}$ and $\mathbf{A}$ are the lists of base case states and actions respectively, and S and A are the states and actions obtained from the sampled transitions. The base states and actions have a reward different from zero for the current episode, where each action is assigned to its state. It is important to propagate information in GCN from rewarding states and actions only. The

actual labels for this loss are calculated using the forward and backward messages defined in Eq. 5. Therefore, the base loss is calculated as follows:

$$\mathscr{L}_0 = H(p(O|\mathbf{S},\mathbf{A}), \phi_{GCRN}(\mathbf{S},\mathbf{A})) = \sum_{s,a\in\mathbf{S},\mathbf{A}} p(O|s,a) log(\phi_{GCRN}(s,a)) \tag{10}$$

where H represents the cross entropy loss between the actual and predicted values by the GCRN. In this approach, the input to the network considers both the state and action to benefit from the look-ahead advice described in Section 2.2.

For the recursive case, the loss function takes the following form:

$$\mathscr{L}_{rec} = \sum_{i=1}^{||d||}\sum_{j=1}^{||e||} \mathfrak{A}_{i,j} ||\phi_{GCRN}(S_i,A_i) - \phi_{GCRN}(S_j,A_j)||^2 \tag{11}$$

where d and e are sets of identifiers for the states and their corresponding neighbors respectively. Furthermore, A_i and A_j are the actions taken at states S_i and S_j respectively, $\mathfrak{A}$ is the adjacency matrix, and $\phi_{GCRN}(S_i,A_i)$ is the output of the shaping function for state S_i while selecting action A_i. Noting that the propagation model in GCRN uses the approximated transition matrix K to aggregate messages. Thus, a message in GCRN is written as $\mathrm{m}_i = \sigma\left(\sum_{j=1}^{||e||} \mathrm{K}_{i,j}\mathrm{m}_j\right)$, where m_j is a message from the neighbor j.

3.3. Computing the Krylov Basis

In our GCRN, a sample from the MDP is converted to a graph structure, where each node corresponds to a state. Furthermore, an edge resembles the transition between states for a given action, as part of the look-ahead advice mechanism. Because constructing the whole graph iteratively is expensive when calculating the message passing, we consider using a sub-graph. The extracted sub-graph from the transition samples is sufficient to construct the shaping function [12]. To compute the Krylov basis K, we apply the augmented Krylov algorithm on the sub-graph. The resulted vectors from the augmented Krylov algorithm are appended to form the Krylov basis K as an approximation of the transition matrix. The full pseudo-code for computing K using the Krylov space and the *weighted spectral method* is presented in Algorithm 1.

Algorithm 1: Transition Matrix Approximation Using Augmented Krylov To Construct K

Input: $P\prime$ - sampled transition matrix, r, e - number of eigenvectors from $P\prime$, n - number of sampled transitions
Output: K - Estimate of the transition matrix P^π
Compute top e eigenvectors of $P\prime$: $\{q_0, q_1, \dots q_e\}$
$q_{e+1} = r$
for $i = 1, \dots, n+e$ **do**
  if $i > e+1$ **then**
    $q_i = P\prime q_{i-1}$
  end if
  for $j = 1, \dots, (i-1)$ **do**
    $q_i = q_i - \left(q_j . q_i\right) q_j$
  end for
end for
$\mathrm{K} = [q_0, q_1, \dots, q_e, \dots, q_n]$
return K

3.4. Training GCRN

A sample of the transition matrix $P\prime$ is taken to form a sub-graph for training the GCRN every n steps. Once $P\prime$ is retrieved, the augmented Krylov algorithm is applied to build the estimate of transitions matrix. The loss function of GCRN is applied following Eqs. (9)–(11). The combined value function with reward shaping takes the form of $Q^\pi_{comb}(s,a) = \alpha Q^\pi(s,a) + (1-\alpha)Q^\pi_\phi(s,a)$, where $Q^\pi_\phi(s,a) = \mathbf{E}_{(s,a)}\left[\sum_t \gamma^t r(S_t,A_t) + \gamma\phi_{GCRN}(S_{t+1},A_{t+1}) - \phi_{GCRN}(S_t,A_t)\right]$. Moreover, α is a hyperparameter indicating the amount of reward shaping decision used in the global value function.

Algorithm 2: Training GCRN

Input: Sets of transitions
Output: A potential function (A trained GCRN)
1: Create empty graph G
2: **for** Episode = 0,1,2, ... **do**
3: **for** $t = 1, 2, \ldots, T$ **do**
4: Store transition $(S_{t-1}, A_{t-1}, S_t, A_t)$
5: Build a graph of transitions in G
6: **end for**
7: **if** mod(Episode, N) **then**
8: Build the sampled transition matrix $P\prime$ from G
9: Construct Krylov basis K using Algorithm 1 from $P\prime$
10: Update GCRN using Eq. (9)
11: **end if**
12: $Q^{\pi}_{comb} = \alpha Q^{\pi} + (1-\alpha)Q^{\pi}_{\phi}$
13: Train DRL to maximize $E_{\pi}\left[\nabla log\pi(A_t|S_t)Q^{\pi}_{comb}(S_t, A_t)\right]$
14: Reset G to empty graph (optional)
15: **end for**

4. Experiments

In this section, we provide a set of experiments for evaluating the performance of GCRN compared to: (1) Actor Critic (A2C) [30], (2) Proximal Policy Optimization (PPO) [27], (3) Random Network Distillation (RND), (4) Intrinsic Curiosity Module (ICM), (5) Learning Intrinsic Rewards for Policy Gradient (LIRPG), and (6) using GCN as the shaping function with the graph Laplacian L_c as the filter [12]. RND and ICM are reward shaping solutions that improve exploration of actions. Furthermore, LIRPG seeks reward shaping through improving the agent performance, but it does not guarantee invariance with respect to the optimal policy. On the other hand, GCN and the proposed GCRN are potential based, scalable, and capable of improving the agent performance. LIRPG does not support continuous action spaces, thus its performance is studied in environments with discrete controls.

In the sequel, we first study the complexity of the proposed GCRN compared to the different baselines in the four rooms and four rooms traps games. Afterwards, we evaluate the performance of GCRN compared to the baselines in the Atari and MuJoCo games for discrete and continuous control.

4.1. Complexity

In order to study the runtime of each reward shaping method, we measure the number of frames processed per second (FPS) for each solution in Atari games. The number of FPS is a measure of the runtime for each method because it indicates the speed of learning the policy and the potential function in the environment. For every episode in GCRN, eigenvector decomposition on the sampled transition matrix is performed to compute the Krylov basis K. To avoid the expensive computation to retrieve the eigenvectors, we utilize the Singular Value Decomposition (SVD) for extracting the top eigenvectors from the sampled transition $P\prime$ [33]. The time complexity for using SVD to calculate the top eigenvectors of a matrix is $O(mde)$, where m is the mean vector of the input, d is dimension, and e is the number of eigenvectors to compute. Furthermore, training GCRN consumes additional time every episode due to training the GCN and GRU layers. Therefore, GCRN has a slightly slower execution time compared to PPO and GCN. Despite this, GCRN is faster compared to RND, ICM, and LIRPG and has the best overall performance. In Table 1, we provide a comparison of the frame processing time (FPS) between PPO, GCN, and our proposed GCRN for potential-based reward shaping.

Table 1
FPS - Atari.

Method	FPS
PPO	1122
GCRN	1028
GCN	1054
RND	1002
ICM	896
LIRPG	274

4.2. Performance Evaluation

To evaluate the performance of the proposed reward shaping approach, we study the impact of each of the proposed techniques for building GCRN as a potential function. Thus, we analyze the impact of (1) ϕ_{kGCRN}: GCRN using Krylov basis and look-ahead advice; (2) ϕ_{kGCN}: GCRN with Krylov basis; and (3) ϕ_{GCRN}: GCRN with look-ahead advice, compared to A2C and ϕ_{GCN} of [12] in tabular learning. We evaluate the performance of these proposed techniques compared to PPO, RND, ICM, LIRPG, and ϕ_{GCN} in 20 Atari for discrete control. Furthermore, additional experiments are performed on four Mujoco games for continuous control compared to PPO and ϕ_{GCN}.

4.2.1. ***Tabular***

We developed two versions of the Four Rooms game to compare the learning speed and analyze the impact of $\phi_{kGCRN}, \phi_{kGCN}$, and ϕ_{GCRN} compared to A2C, ϕ_{GCN}, and $\phi_{\alpha\beta}$ as baselines, where $\phi_{\alpha\beta}$ is using the pure message passing. The two games are Four Rooms and its variant Four Rooms Traps, where negative rewards are scattered through the room as traps. We use tabular learning in the form of A2C, where the critic uses λ-return. In such games, it is possible to calculate the results of the message passing because the environments are small. In addition, random actions are selected with a probability of 0.1. The results showing the cumulative steps to reach the goal are presented in Figs. 3 and 4. As shown in these figures, the performance of $\phi_{kGCRN}, \phi_{kGCN}$, and ϕ_{GCRN} outperforms the rest of the baselines. Furthermore, the best performance can vary between the proposed mechanisms depending on the hyperparameters used. These simple games show the importance of appending RNN to the GCN architecture. The BiGRU layer makes use of the message passing results that require a memory for the forward and backward passes in the environment. Therefore, predicting the shaping value at the next time-steps while considering what happened in the past, contributes to speeding the learning. In the next experiment, we perform our evaluations on the Atari 2600 games, where message passing cannot be directly computed.

4.2.2. ***Atari 2600***

Atari 2600 games are sufficient for comparing the performance between the proposed ϕ_{kGCRN} and ϕ_{GCN} in terms of discrete action space. The Atari games offer a range of environments, which we use to show the importance of the combined solution, rather than using the separate components of augmented Krylov, look-ahead advice, and RNN. In order to add the action to the state vector, we utilize the one-hot-encoding and concatenate the state and action vectors. This step is required to evaluate the look-ahead advice mechanism in our proposed solution.

The performance of $\phi_{kGCRN}, \phi_{GCRN}, \phi_{kGCN}, \phi_{GCN}$, and PPO is evaluated on all Atari games. The same parameters were used for all the learning solutions for fair comparison. The parameters for the Atari games are shown in Table 2. Using GYM python dependency, the pixel rows are passed to a CNN for feature extraction. The input to GCRN is the output of the last hidden layer of the CNN. The experiments were executed for ten million steps for each game.

In Fig. 5, we show the improvement achieved by ϕ_{kGCRN}, ϕ_{GCN}, RND, ICM, and LIRPG compared to PPO in log scale. Following the results displayed in Fig. 5a, ϕ_{kGCRN} learns faster compared to PPO in all the games except RoadRunner. This implies that ϕ_{kGCRN} achieves the best results in terms of the number of games where an improvement is achieved over PPO as baseline.

Furthermore, to show the importance of using GCN combined with the use of augmented Krylov, look-ahead advice, and BiGRUs, we study the improvement of each of the three proposed techniques ($\phi_{kGCRN}, \phi_{kGCN}$, and ϕ_{GCRN}) compared to ϕ_{GCN}. The

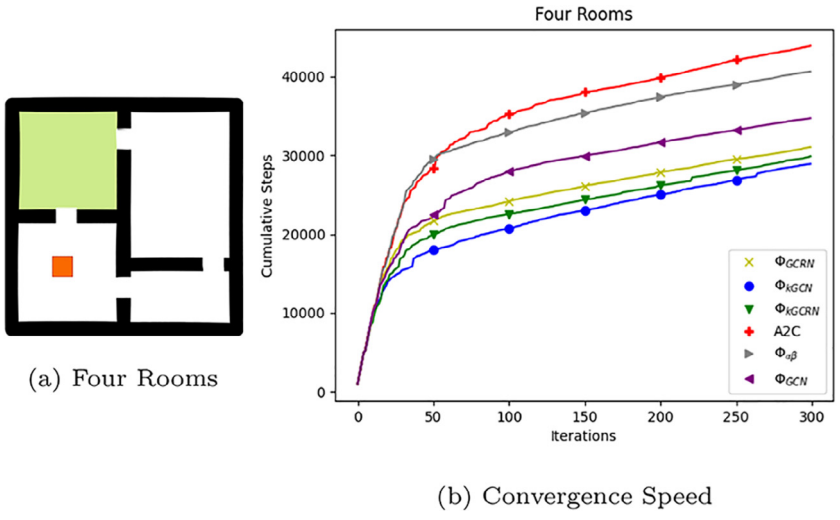


Fig. 3. Convergence speed of different solutions in the Four Rooms game.

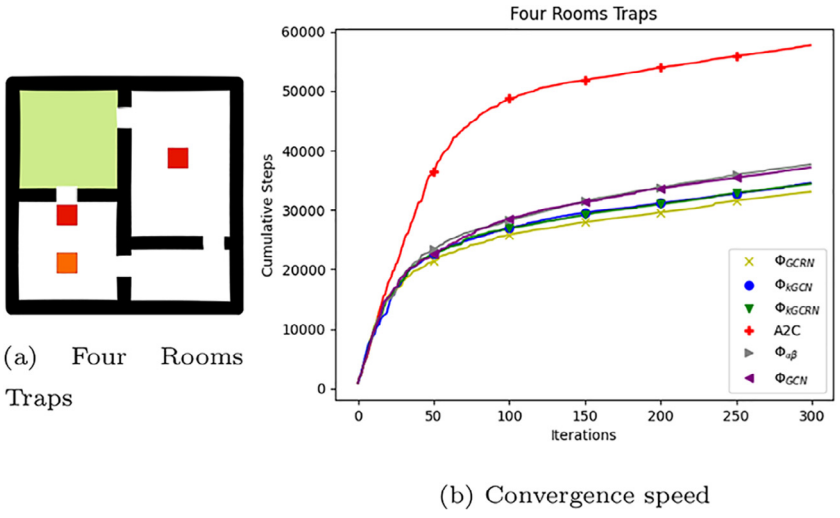


Fig. 4. Convergence speed of different solutions in the Four Rooms Traps game.

Table 2
Model configuration for the Atari games.

Hyperparameter	Value
Learning rate	2.5e-4
γ	0.99
λ	0.95
Entropy Coefficient	0.01
PPO steps	128
PPO Clipping Value	0.1
# of minibatches	4
# of processes	8
GCRN: α	0.9
GCRN: η	1e1
GCN: α	0.9
GCN: η	1e1

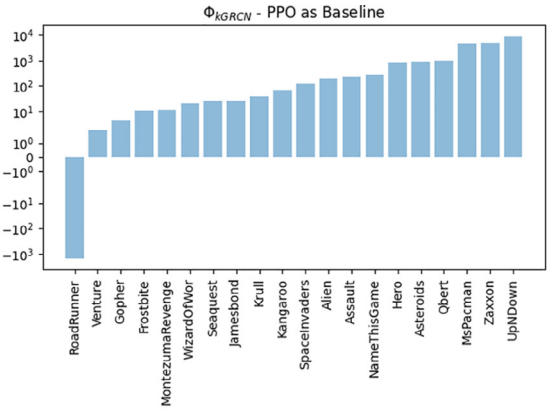
results are presented in Fig. 6 in log scale compared with ϕ_{GCN}. We chose to compare with ϕ_{GCN} as baseline because it is the most related to our proposed solution, and it achieves significant advancement in the domain of reward shaping.

Comparing Fig. 6a with 6b and 6c, we notice the importance of using the combined solution. Considering for example the MsPacman and UpNDown, we can see that ϕ_{kGCRN} improves in terms of convergence speed and maximum reward reached compare to ϕ_{GCN}, ϕ_{kGCN}, and ϕ_{GCRN}. In contrast, taking the example of the RoadRunner game, we can see that ϕ_{GCN}, ϕ_{GCRN}, and ϕ_{kGCN} performed better. Besides, ϕ_{kGCRN} does not always provide the best performance compared to ϕ_{GCN}. Similar to the tabular learning case, the selection of the hyperparameters also affects the performance of each solution. Therefore, studying the spatial and temporal dependencies for the Atari games by using GCRN as the potential function improves the overall learning quality compared to ϕ_{GCN}. Moreover, adding augmented Krylov, look-ahead advice, or both, can also result in improving the learning speed and maximum reward achieved. As a conclustion, there is no one best solution for all the Atari games, thus $\phi_{kGCRN}, \phi_{GCRN}, \phi_{kGCN}$, and ϕ_{GCN} should be tried when tested on similar environments. In Fig. 7, the average reward of each of the solutions for different games is shown.

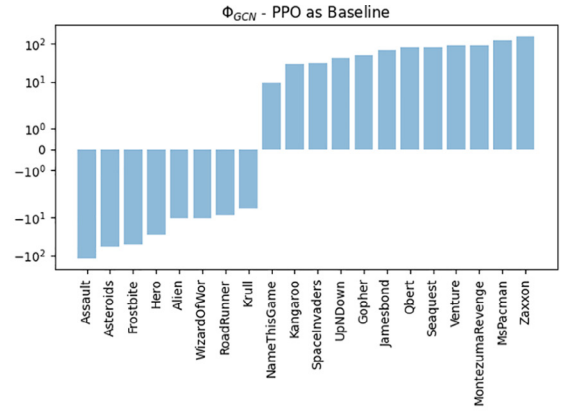
4.2.3. ***Mujoco***

To further investigate the performance of our proposed model compared to ϕ_{GCN} and PPO, we perform experiments on continuous action space using the Mujoco environments. We also use these environments to illustrate the advantage of combining the spatial and temporal learning through GCRN. We evaluate the performance of the three different proposed reward shaping techniques: $\phi_{kGCRN}, \phi_{GCRN}$, and ϕ_{kGCN}. The results comparing the performance of each game and technique, in terms of average rewards, are shown in Fig. 8. We use the same parameters for evaluating the different techniques compared to ϕ_{GCN} for fair comparison. The experiments were executed for three million steps for each game. The model parameters for MuJoCo evaluation are provided in Table 3.

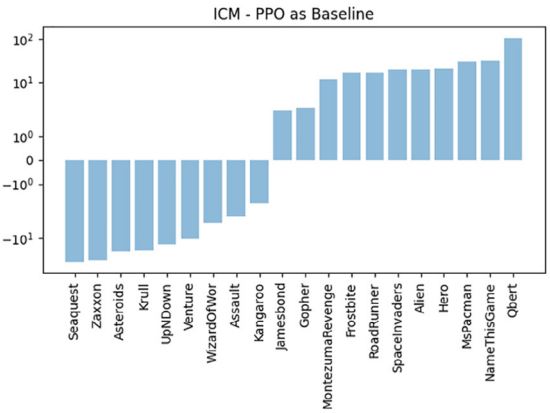
Using our approach, learning on continuous action spaces is possible; however, the evaluation should be done for the different models with varying hyperparameters. It is not guaranteed that ϕ_{kGCRN} always achieves the best performance. Similar



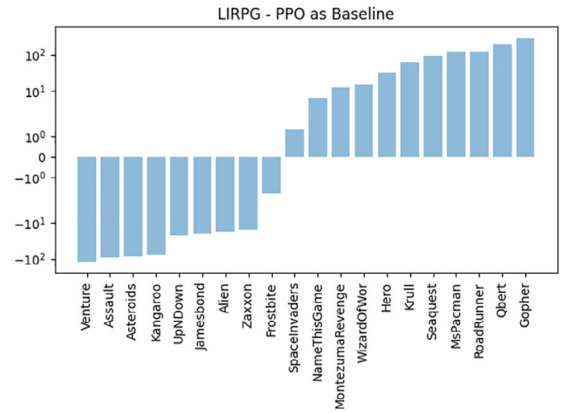
(a) ϕ_{kGCRN} performance different Atari games in log scale over PPO



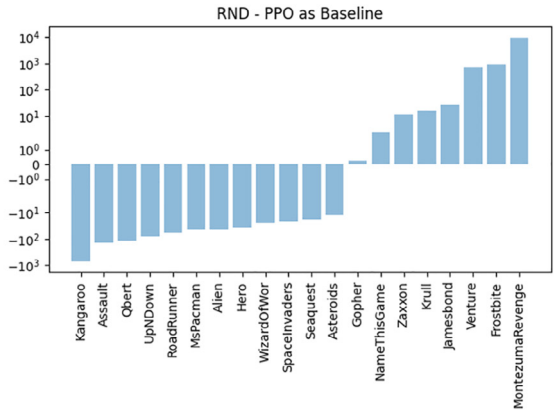
(b) GCN performance different Atari games in log scale over PPO



(c) ICM performance different Atari games in log scale over PPO



(d) LIRPG performance different Atari games in log scale over PPO



(e) RND performance different Atari games in log scale over PPO

Fig. 5. Performance comparison between the proposed ϕ_{kGCRN} and different baselines in Atari games.

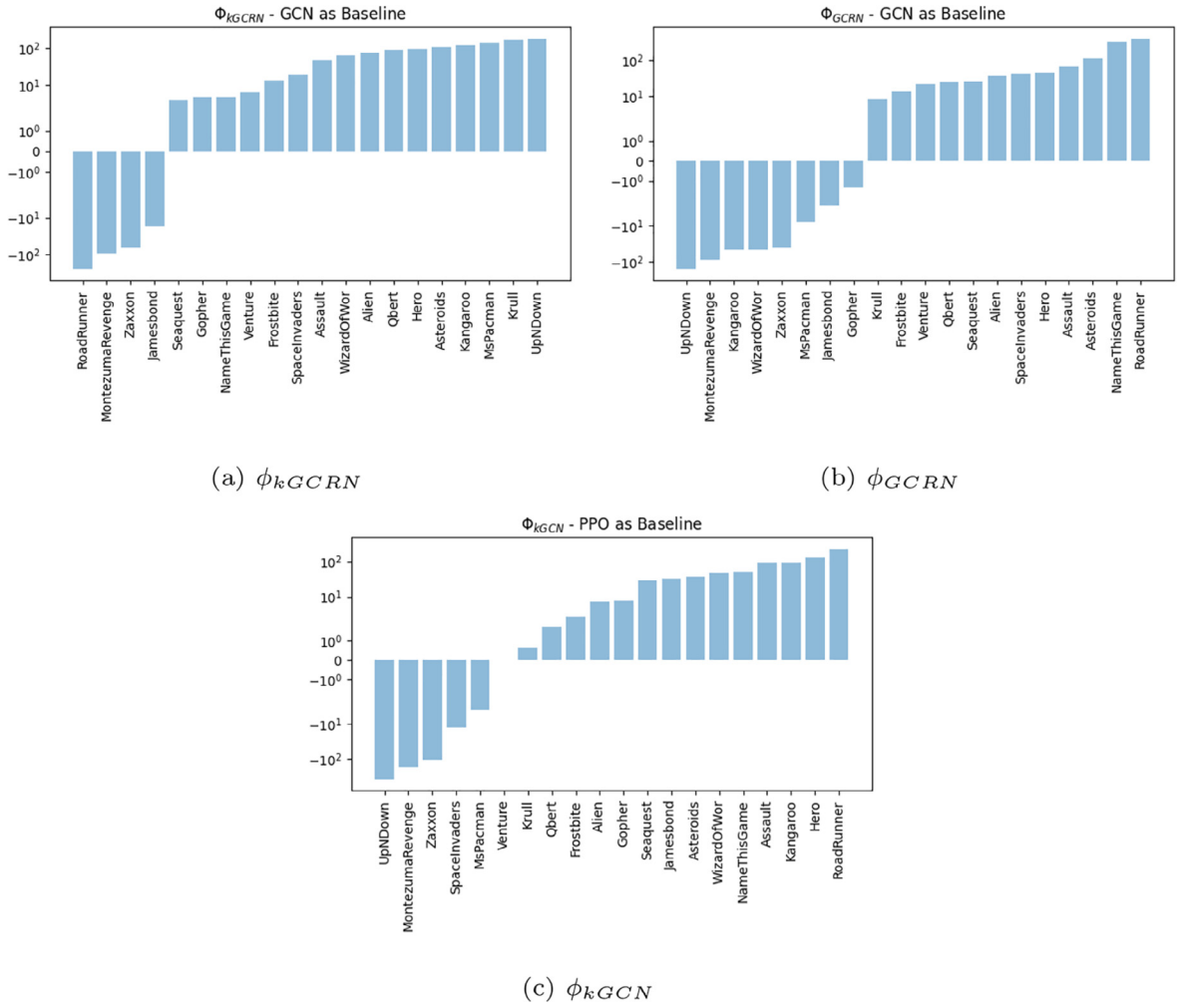


Fig. 6. Performance comparison of the improvement achieved in different Atari games in log scale over ϕ_{GCN}.

to the Atari games, ϕ_{kGCRN} offers the best performance among the remaining approaches. These results further highlight the importance of combining the different techniques to form ϕ_{kGCRN}. In the games of Ant, HalfCheetah, and Hopper, ϕ_{kGCN} is offering a better performance compared to ϕ_{GCN}, thus showing the advantage of using the augmented Krylov instead of graph Laplacian to approximate the transition matrix.

5. Related Work

Reward shaping has gained more attention in the past years due to the importance of speeding the learning process in RL, especially in applications requiring real-time feedback [25] and after proposing the Deep Reinforcement Learning [16]. In this section, we cover some the most recent research proposals in the field of reward shaping for RL.

In [38], the authors are proposing a task distribution reward shaping with metadata. In other words, a meta-learning framework is proposed to learn reward shaping functions from the environment. The advantage of the solution is adapting the reward shaping function to different action spaces while the state space is shared. Our solution works on improving the potential function itself, so combining our solution with [38] could produce state-of-the art results in learning task distribution rewards. In [9], the authors consider the utilization of the reward shaping as a bi-level optimization problem, where the lower level optimizes the policy using the shaping function, while the upper part optimizes the weights of the shaping function to adapt to different settings in the environment. In the same context, our GCRN solution also performs two level optimization. However, using deep learning makes our approach more scalable in large environments thanks to a scalable deep learning architecture. The authors in [5] propose a reward shaping solution based on natural language instructions that are passed to the reinforcement learning solution to speed learning. This approach requires expert knowledge and human inputs, which is costly and time consuming. In [32], the authors designed a reward shaping solution that is based on improv-

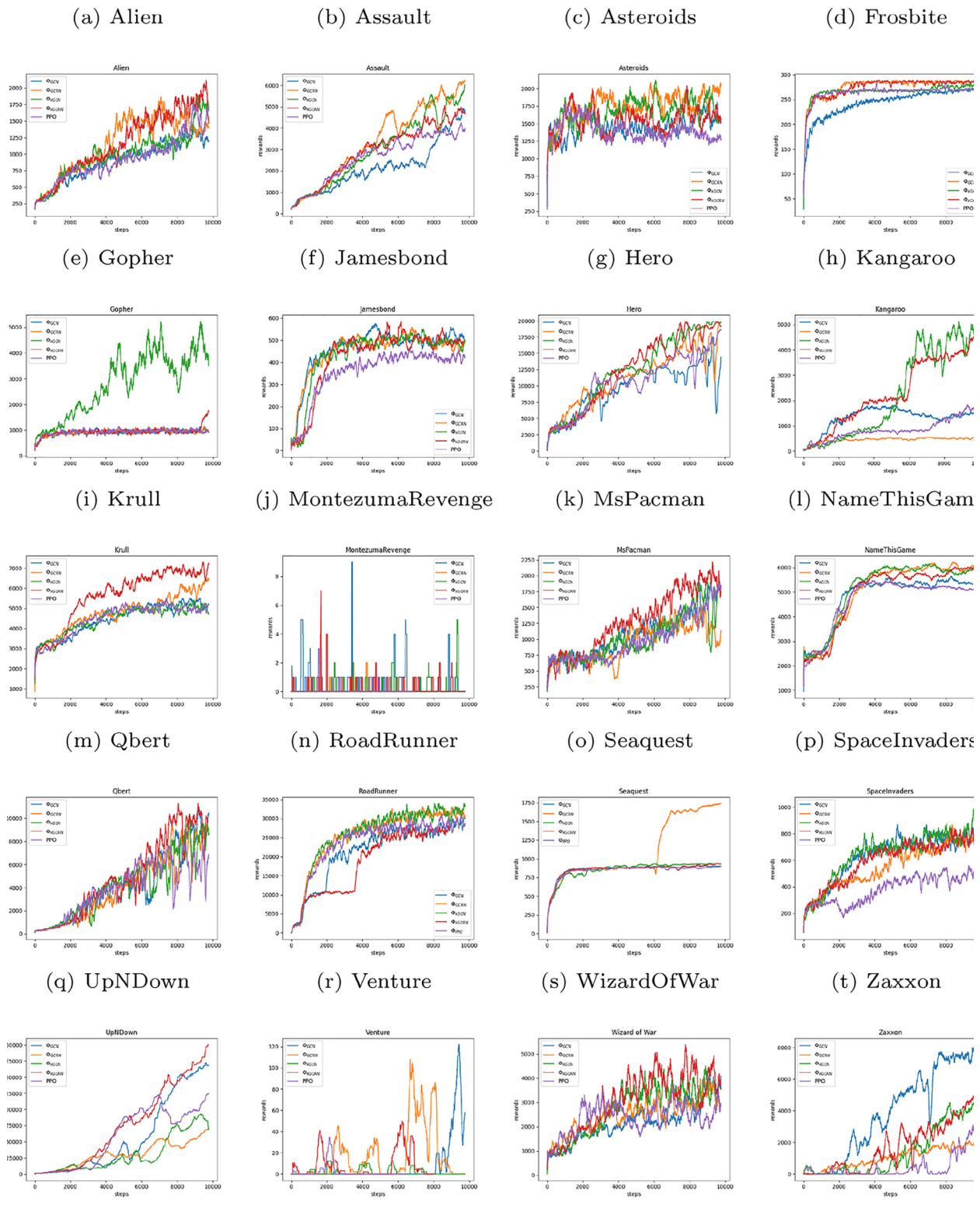


Fig. 7. Results on 20 Atari games comapring the performance of ϕ_{CNN} to ϕ_{GCN} and PPO.

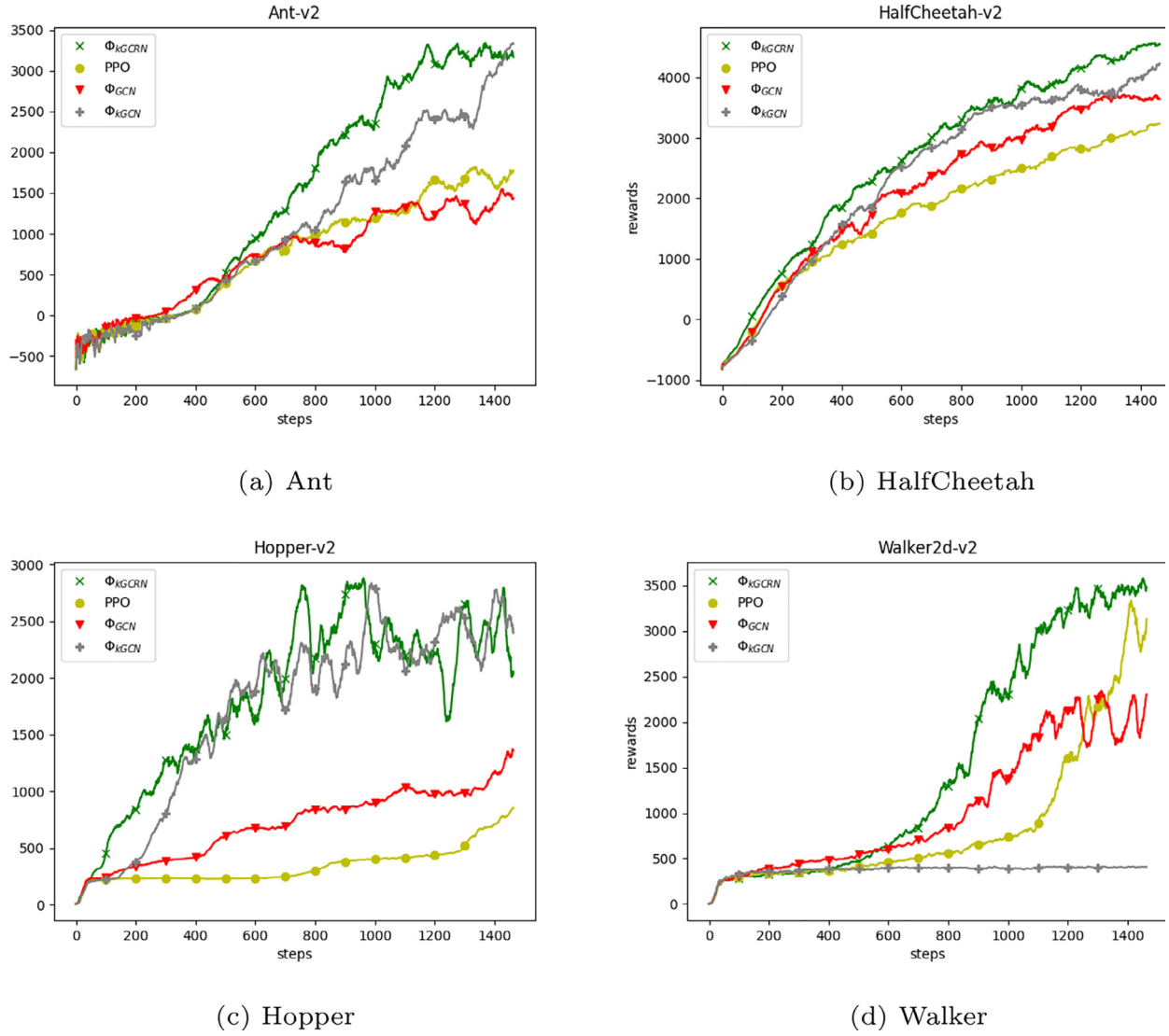


Fig. 8. Performance comparison between different reward shaping mechanisms and PPO in Mujoco environments.

Table 3
Model configuration for the MuJoCo games.

Hyperparameter	Value
Learning rate	3e-4
γ	0.99
λ	0.95
Entropy Coefficient	0.0
PPO steps	2048
PPO Clipping Value	0.1
# of minibatches	32
# of processes	1
GCRN/GCN (Walker and Ant): α	0.6
GCRN/GCN (Hopper and HalfCheetah): α	0.6
GCRN/GCN: η	1e1

ing action exploration using distance-based calculations to the goal. Reward shaping based on improving action exploration is different compared to GCRN, where our focus is engineering the reward function itself to improve the learning. Besides, not all environments support distance-based calculations to the goal.

On the other hand, there is an increased interest in using the graph Laplacian for supporting RL tasks such as option discovery or reward shaping. There are various existing approximators for the graph Laplacian in the literature that use spectral graph drawings, which results in indefinite global minimizers. In [35], the authors proposed an improved solution based on generalized graph drawing, which provides a unique global minimizer and can further improve the RL performance especially when used for reward shaping. As discussed earlier, we avoid using the graph laplacian as an approximation for the transition matrix when forming the potential function, because it is assumed that the value function is smooth over the induced MDP graph. Therefore, using graph Laplacian cannot generalize to all MDPs. Moreover, we use the augmented Krylov in our solution to solve this problem.

Furthermore, there are various potential-based reward shaping solutions in the literature that mainly suffer from the problem of scalability [6], or requiring human intervention [8]. Therefore, we focus in this work on comparing with the recent scalable solution of using GCN as the shaping function, which overcomes the aforementioned issues. Recently, [12] were able to achieve a major advancement in the field of potential-based reward shaping by benefiting from the recursive nature of GCN to perform message passing. State of the art results are achieved for some games. The main drawback of this work is using the graph Laplacian for approximating the transition matrix. In this paper, we use the augmented Krylov algorithm for the first time to estimate the transitin matrix, which offers a smaller margin of error compared to graph Laplacian when performing VFA. Furthermer, [12] does not study the temporal dependencies between nodes of the graph, which helps in speed the learning further. Thus, we append an RNN network to GCN for predicting the reward shaping value of the next timestep. We also propose using the look-ahead advice for advice at the action level.

For many years, improvements were proposed for reward shaping, such as applying the concept of look-ahead advice or improving the transition matrix approximation. look-ahead advice does not alter the optimal policy and augments action selection for more precise advice [36]. Several work utilized the look-ahead concept in the context of reward shaping for speeding RL [13,2,15]. In this work, we apply the look-ahead advice for the first time using GCRN with augmented Krylov for reward shaping.

Several methods were proposed to construct bases for VFA such as using the graph Laplacian. It is proven that the graph Laplacian can only produce effective VFA in specific cases and cannot generalize to all MDPs [19]. Thus, the augemented Krylov method is used to overcome the Laplacian limitations. Several work extend and use the augemented Krylov method for improving the VFA, while others employ it in real applications [29,34]. In this work, we propose for the first time the use of the Krylov basis as an estimate to the transition matrix.

The work published in [28] is one of the first papers that propose GCRN by combining GCN and RNN. GCRN is proven to acquire spatial and temporal information in graph structures. For instance, traffic prediction is an example of a problem that requires retrieving information from neighboring roads, as well as traffic change over time. Using GCRN, traffic forecasting in complex road structures is possible with more robust models [4]. In our work, we propose the use of GCRN for the first time in the context of RL.

Summary: The research in the area of reward shaping is divided into non-potential and potential based. The non-potential based solutions do not guarantee invariance towards the optimal policy. Some solutions focus on improving the action exploration and may require expert knowledge. These factors lead to performance instability and increase in the efforts required to learn and adapt the shaping function. Recent scalable potential-based solutions suffer from the issue of approximating the transition matrix. Using graph Laplacian to approximate the transition matrix cannot generalize to all MDP environments.

In this work, we overcome the above limitations by proposing a potential-based scalable reward shaping solution named GCRN. GCRN combines GCN and RNN to form the potential function, which offers the ability to learn the spatial and temporal dependencies. Our solution utilizes the augmented Krylov algorithm to build the Krylov basis as a better estimate to the transition matrix. In addition, our solution embeds the look-ahead advice mechanism to further boost the learning speed.

6. Conclusion and Discussion

Our work proposes a novel GCRN scheme for potential-based reward shaping, which guarantees invariance in the optimal policy. The shaping function of GCRN combines layers of GCN followed by RNN to capture spatio-temporal dependencies between the sampled states. The training of GCRN is performed on a sample of transitions. Our solution embeds the look-ahead advice methodology and uses the augmented Krylov algorithm to estimate the transition matrix. Computing the actual labels of our GCRN is inspired by the probabilistic view of RL to perform message passing. The proposed GCRN excels in terms of convergence speed compared to existing potential-based reward shaping solutions. ϕ_{kGCRN} achieves state of the art learning speed in some of the games. Besides, ϕ_{GCRN} and ϕ_{kGCN} also outperform other solutions in particular environments. Therefore, we recommend trying the three techniques we experimented with when testing the agent performance including the use of the Krylov basis and look-ahead advice.

The computational complexity of training the ϕ_{kGCRN} varies depending on the size of sampled transitions. In addition, we believe that the sub-graph selection can be improved by capturing more important information from the trajectories traversed, to be included in future samples. Furthermore, we aim to augment our solution to be deployed in multi-agent systems while trying to optimize multiple objectives.

Later on, we believe that the proposed GCRN can be applied to a wider range of applications utilizing RL for solving time-sensitive problems. The results achieved improve the learning speed, which has direct impact on the feasibility of RL in various applications. For example, fast decisions are essential for autonomous driving [7,23], resource management [26], task scheduling [22], trust-driven reinforcement selection for federated learning [24] and health-related applications.

CRediT authorship contribution statement

Hani Sami: Formal analysis, Investigation, Software. **Jamal Bentahar:** Conceptualization, Formal analysis, Validation. **Azzam Mourad:** Conceptualization, Formal analysis, Validation. **Hadi Otrok:** Conceptualization, Formal analysis, Validation. **Ernesto Damiani:** Conceptualization, Formal analysis, Methodology.

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.

References

[1] Amodei, D., Olah, C., Steinhardt, J., Christiano, P.F., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. CoRR, abs/1606.06565..
[2] T. Brys, A. Harutyunyan, H.B. Suay, S. Chernova, M.E. Taylor, A. Nowé, Reinforcement learning from demonstration through shaping, in: IJCAI, AAAI Press, 2015, pp. 3352–3358.
[3] F.R. Chung, F.C. Graham, Spectral graph theory, 92., American Mathematical Soc, 1997.
[4] Z. Cui, K. Henrickson, R. Ke, Y. Wang, Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting, IEEE Transactions on Intelligent Transportation Systems 21 (2019) 4883–4894.
[5] P. Goyal, S. Niekum, R.J. Mooney, Using natural language for reward shaping in reinforcement learning, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, 2019, pp. 2385–2391.
[6] M. Grześ, D. Kudenko, Online learning of shaping rewards in reinforcement learning, Neural Networks 23 (2010) 541–550.
[7] A. Hammoud, H. Sami, A. Mourad, H. Otrok, R. Mizouni, J. Bentahar, AI, blockchain, and vehicular edge computing for smart and secure IoV: Challenges and directions, IEEE Internet of Things Magazine 3 (2020) 68–73.
[8] Harutyunyan, A., Brys, T., Vrancx, P., & Nowé, A. (2015). Shaping mario with human advice. In AAMAS (pp. 1913–1914)..
[9] Y. Hu, W. Wang, H. Jia, Y. Wang, Y. Chen, J. Hao, F. Wu, C. Fan, Learning to utilize shaping rewards: A new approach of reward shaping, Advances in Neural Information Processing Systems 33 (2020) 15931–15941.
[10] I.C. Ipsen, C.D. Meyer, The idea behind krylov methods, The American mathematical monthly 105 (1998) 889–899.
[11] Kipf, T.N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. CoRR, abs/1609.02907..
[12] M. Klissarov, D. Precup, Reward propagation using graph convolutional networks, NeurIPS (2020).
[13] Knox, W.B., & Stone, P. (2012). Reinforcement learning from simultaneous human and mdp reward. In AAMAS (pp. 475–482)..
[14] S. Mahadevan, M. Maggioni, Value function approximation with diffusion wavelets and laplacian eigenfunctions, NeurIPS 18 (2006) 843.
[15] D.K. Misra, J. Langford, Y. Artzi, Mapping instructions and visual observations to actions with reinforcement learning, in: EMNLP 2017, Association for Computational Linguistics, 2017, pp. 1004–1015.
[16] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M.A. (2013). Playing atari with deep reinforcement learning. CoRR, abs/1312.5602..
[17] A.Y. Ng, D. Harada, S. Russell, Policy invariance under reward transformations: Theory and application to reward shaping, In Icml 99 (1999) 278–287.
[18] J. Oh, M. Hessel, W.M. Czarnecki, Z. Xu, H.P. van Hasselt, S. Singh, D. Silver, Discovering reinforcement learning algorithms, NeurIPS 33 (2020).
[19] Petrik, M. (2007). An analysis of laplacian methods for value function approximation in mdps. In IJCAI (pp. 2574–2579)..
[20] M.L. Puterman, Markov decision processes: discrete stochastic dynamic programming, John Wiley & Sons, 2014.
[21] L. Rabiner, B. Juang, An introduction to hidden markov models, IEEE ASSP Magazine 3 (1986) 4–16.
[22] G. Rjoub, J. Bentahar, O.A. Wahab, A.S. Bataineh, Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems, Concurrency and Computation: Practice and Experience 33 (2021).
[23] Rjoub, G., Wahab, O.A., Bentahar, J., & Bataineh, A.S. (2021b). Improving autonomous vehicles safety in snow weather using federated YOLO CNN learning. In J. Bentahar, I. Awan, M. Younas, & T. Grønli (Eds.), Mobile Web and Intelligent Information Systems - 17th International Conference, MobiWIS 2021, Virtual Event, August 23–25, 2021, Proceedings (pp. 121–134). Springer volume 12814 of Lecture Notes in Computer Science..
[24] G. Rjoub, O.A. Wahab, J. Bentahar, A.S. Bataineh, Trust-driven reinforcement selection strategy for federated learning on IoT devices, Computing (2022), in press.
[25] H. Sami, A. Mourad, H. Otrok, J. Bentahar, Demand-driven deep reinforcement learning for scalable fog and service placement, IEEE Transactions on Services Computing (2021), in press.
[26] H. Sami, H. Otrok, J. Bentahar, A. Mourad, AI-based resource provisioning of IoE services in 6G: A deep reinforcement learning approach, IEEE Transactions on Network and Service Management 18 (2021) 3527–3540.
[27] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. CoRR, abs/1707.06347..
[28] Y. Seo, M. Defferrard, P. Vandergheynst, X. Bresson, in: I.C.O.N.I.P. In (Ed.), Structured sequence modeling with graph convolutional recurrent networks, Springer, 2018, pp. 362–373.
[29] A. Somani, N. Ye, D. Hsu, W.S. Lee, Despot: Online pomdp planning with regularization, NIPS 13 (2013) 1772–1780.
[30] Sutton, R.S., McAllester, D.A., Singh, S.P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In Advances in neural information processing systems (pp. 1057–1063)..
[31] Toussaint, M., & Storkey, A. (2006). Probabilistic inference for solving discrete and continuous state markov decision processes. In ICML (pp. 945–952)..
[32] A. Trott, S. Zheng, C. Xiong, R. Socher, Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards, Advances in Neural Information Processing Systems 32 (2019).
[33] M.E. Wall, A. Rechtsteiner, L.M. Rocha, Singular value decomposition and principal component analysis, in: A practical approach to microarray data analysis, Springer, 2003, pp. 91–109.
[34] K. Wampler, E. Andersen, E. Herbst, Y. Lee, Z. Popović, Character animation in two-player adversarial games, ACM Transactions on Graphics (TOG) 29 (2010) 1–13.
[35] K. Wang, K. Zhou, Q. Zhang, J. Shao, B. Hooi, J. Feng, Towards better laplacian representation in reinforcement learning with generalized graph drawing, in: International Conference on Machine Learning, PMLR, 2021, pp. 11003–11012.
[36] Wiewiora, E., Cottrell, G.W., & Elkan, C. (2003). Principled methods for advising reinforcement learning agents. In ICML (pp. 792–799)..
[37] Ziebart, B.D., Maas, A.L., Bagnell, J.A., & Dey, A.K. (2008). Maximum entropy inverse reinforcement learning. In AAAI (pp. 1433–1438). AAAI Press..

[38] H. Zou, T. Ren, D. Yan, H. Su, J. Zhu, Learning task-distribution reward shaping with meta-learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 2021, pp. 2–9.

Hani Sami is currently pursuing his Ph.D. at Concordia University, Institute for information Systems Engineering (CIISE). He received his M.Sc. degree in Computer Science from the American University of Beirut and completed his B.S. and worked as Research Assistant at the Lebanese American University. The topics of his research are Fog Computing, Vehicular Fog Computing, and Reinforcement Learning. He is a reviewer of several prestigious conferences and journals.

Jamal Bentahar received the Ph.D. degree in computer science and software engineering from Laval University, Canada, in 2005. He is a Professor with Concordia Institute for Information Systems Engineering, Concordia University, Canada. From 2005 to 2006, he was a Postdoctoral Fellow with Laval University, and then NSERC Postdoctoral Fellow at Simon Fraser University, Canada. He is an NSERC Co-Chair for Discovery Grant for Computer Science (2016–2018). His research interests include the areas of computational logics, model checking, multi-agent systems, services computing, game theory, and deep learning.

Azzam Mourad received his M.Sc. in CS from Laval University, Canada (2003) and Ph.D. in ECE from Concordia University, Canada (2008). He is currently a Professor of Computer Science with the Lebanese American University, a Visiting Professor of Computer Science with New York University Abu Dhabi and an Affiliate Professor with the Software Engineering and IT Department, Ecole de Technologie Superieure (ETS), Montreal, Canada. He published more than 100 papers in international journal and conferences on Security, Network and Service Optimization and Management targeting IoT, Cloud/Fog/Edge Computing, Vehicular and Mobile Networks, and Federated Learning. He has served/serves as an associate editor for IEEE Transactions on Services Computing, IEEE Transactions on Network and Service Management, IEEE Network, IEEE Open Journal of the Communications Society, IET Quantum Communication, and IEEE Communications Letters, the General Chair of IWCMC2020, the General Co-Chair of WiMob2016, and the Track Chair, a TPC member, and a reviewer for several prestigious journals and conferences. He is an IEEE senior member.

Hadi Otrok received his Ph.D. in ECE from Concordia University. He holds a Full Professor position in the department of Electrical Engineering and Computer Science (EECS) at Khalifa University. Also, he is an Affiliate Associate Professor in the Concordia Institute for Information Systems Engineering at Concordia University, Montreal, Canada, and an Affiliate Associate Professor in the Electrical department at Ecole de Technologie Superieure (ETS), Montreal, Canada. His research interests include the domain of blockchain, reinforcement learning, crowd sensing and sourcing, ad hoc networks, and cloud security. He co-chaired several committees at various IEEE conferences. He is also an Associate Editor at IEEE Transactions on Services Computing, IEEE Transactions on Network and Service Management (TNSM), Ad-hoc networks (Elsevier), and IEEE Network. He also served from 2015 to 2019 as an Associate Editor at IEEE Communications Letters.

Ernesto Damiani is currently a Full Professor at the Department of Computer Science, Universita degli Studi di Milano, where he leads the Secure Service-oriented Architectures Research (SESAR) Laboratory. He is also the Founding Director of the Center for Cyber-Physical Systems, Khalifa University, United Arab Emirates. He received an Honorary Doctorate from Institute National des Sciences Appliquees de Lyon, France, in 2017, for his contributions to research and teaching on big data analytics. He is the Principal Investigator of the H2020 TOREADOR project on Big Data as a Service. He serves as Editor in Chief for IEEE Transactions on Services Computing. His research interests include cyber-security, big data, and cloud/edge processing, and he has published over 600 peer-reviewed articles and books. He is a Distinguished Scientist of ACM and was a recipient of the 2017 Stephen Yau Award.