2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
17-20 October, 2021. Melbourne, Australia

Accelerating Lifelong Reinforcement Learning via Reshaping Rewards*

Kun Chu[1], Xianchao Zhu[1], William Zhu[1]

2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC) | 978-1-6654-4207-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/SMC52423.2021.9659064

Abstract— **The reinforcement learning (RL) problem is typically formalized as the Markov Decision Process (MDP), where an agent interacts with the environment to maximize the long-term expected reward. As an important branch of RL, Lifelong RL requires the agent to consecutively solve a series of tasks modeled as MDPs, each of which is drawn from some distribution. A crucial issue in Lifelong RL is how best to utilize the knowledge from the previous tasks for improving the performance in the current task. As a pioneering work in this field, MaxQInit takes the maximum over action-values learned from previous tasks' environmental rewards as the initial action-value of the current task. In this way, MaxQInit improves the initial performance in the current task and reduces the sample complexity of learning. However, the rewards obtained in the learning process are usually delayed and sparse, dramatically decreasing the learning efficiency. In this paper, we propose a new method, Shaping Rewards for Lifelong RL (SR-LLRL), to speed up the lifelong learning process by shaping timely and informative rewards for each task. Critically, we construct the Lifetime Reward Shaping (LRS) function based on the knowledge of optimal trajectories collected in previous tasks, providing additional reward information for the current task. Compared with MaxQInit, our method exhibits higher learning efficiency and superior performance in Lifelong RL experiments.**

I. Introduction

The reinforcement learning (RL) [1] problem is usually formulated as the Markov Decision Process (MDP) [2], in which an agent interacts with the environment to find a policy that maximizes the long-term expected reward. An increasingly valuable sub-area of RL is Lifelong RL, where the agent solves a sequence of tasks, each of which is drawn from some distribution over a finite set of MDPs [3], [4]. The key question in this setting is how previously acquired knowledge can be exploited to improve the performance in the current task [5], [6]. Some progress has been made in this field, such as incorporating the experience samples previously collected to reduce the number of samples required for learning [7], [8], [9], reusing the past learned policies to bias the learning process [10], [11], [12], selecting related value function from the set of source tasks to accelerate learning in the target task [13], [14], [15] and more. Although these methods have attained impressive results, the knowledge they utilize is usually dependent on specific tasks, rather than can be shared among all tasks.

*This work is supported in part by The National Nature Science Foundation of China under Grant Nos. 61772120.

[1]Kun Chu, Xianchao Zhu, and William Zhu are with Institute of Fundamental and Frontier Sciences, University of Electronic Science and Technology of China, Chengdu, Sichuan, China. e-mail: kun_chu@outlook.com, xczhuiffs@163.com, wfzhu@uestc.edu.cn

As a pioneering work in Lifelong RL, MaxQInit [16] injects the knowledge represented by the maximum over action-values (Q-values) learned from previous tasks into the current task to improve the learning performance. In particular, MaxQInit first learns the Q-value from the environmental rewards of each sampled task. Then, when the amount of learned tasks is over some threshold, MaxQInit takes the maximum of the learned Q-value over these tasks as the initial value of the Q-function in the current task. In this way, MaxQInit obtains a better initial policy than other algorithms, achieving good performance immediately. Moreover, MaxQInit reduces the number of samples required for learning the near-optimal policy. However, the rewards obtained in the learning process are usually sparse and delayed, leading to a dramatic decrease in learning efficiency.

To address the above problem, we propose a new algorithm, Shaping Rewards for Lifelong RL (SR-LLRL), to accelerate lifelong learning through shaping timely and informative rewards in each task. To this end, we utilize the knowledge among previous tasks to provide shaped rewards, which are augmented to the original rewards in the current task. Intuitively, state-action pairs and states with higher visit-counts are more helpful for reaching the goal state [17], [18]. Therefore, we collect the visit-counts of state-action pairs and states in the optimal trajectory, which is the trajectory with the highest cumulative rewards in each learned task. Then, based on Reward Shaping [19], [20], we construct the Lifetime Reward Shaping (LRS) function to reshape the rewards in the current task according to previously collected visit-counts. With additional reward information, our method effectively speeds up the convergence process of the algorithm. Extensive experiments in several classical grid-world domains illustrate that SR-LLRL can achieve higher learning efficiency and better performance than MaxQInit.

The rest of this paper is organized as follows. In Section II, we briefly review RL and Lifelong RL. In Section III, based on Reward Shaping, we construct the LRS function through the optimal trajectories collected in previous tasks. Then, we propose the SR-LLRL algorithm based on the LRS function, so as to speed up the lifelong learning process. In Section IV, we empirically demonstrate that our algorithm has higher learning efficiency and superior performance compared to MaxQInit on benchmark experiments. Finally, we conclude in Section V.

II. Background

The reinforcement learning (RL) [1] problem is usually modeled as a finite Markov Decision Process (MDP, [21]) consisting in a five-tuple: $\langle \mathcal{S}, \mathcal{A}, \gamma, \mathcal{T}, \mathcal{R} \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\gamma \in [0,1]$ is the discount factor, $\mathcal{T}$: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is the state transition probability function and $\mathcal{R}$: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, R_{\max}]$ is the reward function, where rewards all lie between 0 and $R_{\max}$. A policy $\pi : \mathcal{S} \to \mathcal{A}$ instructs which action should be taken given the current state. Given a fixed policy π and a start state $s_0 = s$, the value function $V_M^\pi(s)$ in MDP M is given by:

$$V_M^\pi(s) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t \mid s_0 = s, \pi\right] \tag{1}$$

where r_t is a real reward received in time step t. Similarly, the action-value (Q-value) function $Q_M^\pi(s,a)$ of the policy π in MDP M is given by:

$$Q_M^\pi(s,a) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t \mid s_0 = s, a_0 = a, \pi\right] \tag{2}$$

where $(\{s_t, a_t\}_{t \geq 0})$ is a trajectory generated by taking action $a_0 = a$ in state $s_0 = s$ and following π thereafter, i.e., $a_t = \pi(s_t)$ (for $t \geq 1$). The optimal policy is denoted as π^* and its corresponding value functions are $V_M^*(s)$ and $Q_M^*(s,a)$, respectively. Since the reward function is uniformly bounded by $R_{\max} > 0$, we let $V_{\max}$ be a known upper bound on the value that a policy can have in any state, which is at most $R_{\max}/(1-\gamma)$ but can be much smaller.

Of growing interest in RL literature is Lifelong RL [3], [4], in which an agent solves a sequence of tasks, each of which is sampled independently from some distribution over a finite set of MDPs. During the agent's lifetime, tasks are sampled consecutively and a classical RL question takes place each time. Naturally, a key question in this setting is how best to utilize the previously learned knowledge to improve the performance on the current task [5], [6]. Several algorithms have been proposed to address this question. For example, TIMBREL [8] helps to construct a target task model through experience instances from the source task. Policy Reuse [10] is a technique to probabilistically bias the exploration given past learned policies. Recently, [22] introduce a similarity measure between tasks, which helps select related policies to instruct learning in the new task. Similarly, [14], [15] propose new metrics to select value functions from appropriate source tasks for accelerating learning in the new task. Although these encouraging works have shown improved performance, the knowledge they utilize is usually dependent on specific tasks, rather than can be shared among all tasks. In other words, the knowledge they leverage may be inapplicable to the current task, or even impair the task's performance.

As a pioneering work in Lifelong RL, MaxQInit [16] provides provable upper bounds on Q-functions across the task space with high probability, reducing the number of samples required for learning near-optimal policies. In other words, MaxQInit provides a type of knowledge that can be shared across the MDPs drawn from the same distribution, provably reducing the sample complexity of learning. In particular, MaxQInit first learns the Q-value from each sampled task's environmental rewards. Then, when the number of learned tasks is greater than some threshold, MaxQInit takes the maximum of the Q-value over the set of learned tasks, as the initial value of the Q-function in the current task. In this way, MaxQInit obtains a better initial policy than re-learning from scratch, achieving good performance immediately in the current task. Moreover, MaxQInit improves the speed of learning the near-optimal policy. However, the rewards obtained in the learning process are usually delayed and sparse, where a long sequence of uninformed actions is required to reach a reward state, dramatically decreasing the learning efficiency.

III. Shaping Rewards for Lifelong Reinforcement Learning

In this section, we propose the SR-LLRL algorithm to address the above-mentioned question by shaping timely and informative rewards in each task. Specifically, based on Reward Shaping, we design the Lifetime Reward Shaping (LRS) function, which reshapes the rewards in the current task based on the knowledge accumulated by solving previous tasks.

A. Reward Shaping

In this part, we provide a brief survey about Reward shaping, which is an effective way to speed up the learning process by introducing additional reward functions representative of prior knowledge [23], [24], [19], [20]. Normally, a reward shaping function $\mathcal{F}$: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ modifies the original reward function to a new one,

$$\mathcal{R}'(s,a,s') = \mathcal{R}(s,a,s') + \mathcal{F}(s,a,s'). \tag{3}$$

It transforms the original MDP M: $\langle \mathcal{S}, \mathcal{A}, \gamma, \mathcal{T}, \mathcal{R} \rangle$ into another shaped MDP M': $\langle \mathcal{S}, \mathcal{A}, \gamma, \mathcal{T}, \mathcal{R}' = \mathcal{R} + \mathcal{F} \rangle$. The Q-function in the new MDP can converge to an optimal value faster by designing an appropriate function $\mathcal{F}(s,a,s')$.

Several works adopted diverse methods to design reward shaping functions for better performance. [20] introduced potential-based reward shaping (PBRS), where the reward shaping function is expressed as the difference of the potential function applied to the states. Then, there are some works that extended PBRS to more flexible cases [25], [26], [27]. Recently, [28] presented a Bayesian reward shaping method to provide shaping rewards directly based on the full transition information. Moreover, some works proposed to take natural language [29], [30], human feedback [31] or human demonstrations [32] as heuristic knowledge to construct reward shaping functions.

B. Lifetime Reward Shaping function

Based on Reward Shaping, we design the LRS function to provide additional rewards in the current task through the previously learned knowledge. Specifically, LRS contains two processes in the construction. In the first process, we collect useful knowledge from previous tasks as the basis for the LRS function. The trajectory is an intuitive and important data accumulated during the agent's interaction with the environment. Thus, we store the trajectories and corresponding cumulative rewards in terms of two-tuple during the learning process in previous tasks. Formally, after m episodes' learning in task k, we obtain the data as follows,

$$\mathcal{Z}_k : \{(\tau_{k,i}, G_{k,i})\} \ (1 \leq i \leq m, \ 1 \leq k \leq n-1) \tag{4}$$

where $\tau_{k,i} = \{(s_t, a_t)_{t \geq 0}\}$ is the generated trajectory from episode i, and $G_{k,i} = \sum_{t \geq 0} r_t$ is the cumulative rewards obtained along with trajectory $\tau_{k,i}$. Generally, the trajectory with higher cumulative rewards will be more instructive for the learning. Therefore, we only select the optimal trajectory τ_k^* as the knowledge we require from task k, which is corresponding to the highest cumulative rewards, $G_k^* \leftarrow \max\{G_{k,1}, G_{k,2}, \cdots, G_{k,m}\}$.

In the latter process, we utilize the optimal trajectories to construct the LRS function. Specifically, LRS function forms the rewards based on the visit-counts of states and state-actions collected in the optimal trajectories. There is an intuition that the state-action pairs and states with higher visit-counts are normally more helpful for the agent to reach the goal state [17], [18]. From this point of view, we construct the LRS function $\mathcal{F}_n$ in task n as below,

$$\forall_{s,a,s'} : \mathcal{F}_n(s,a,s') = (1-\gamma)V_{\max} \frac{\mathcal{C}_{n-1}(s,a,s')}{\mathcal{C}_{n-1}(s)} \tag{5}$$

where $\mathcal{C}_{n-1}(s,a,s')$ and $\mathcal{C}_{n-1}(s)$ are the cumulative visit-counts of state-action-state pair (s,a,s') and state s collected from the trajectories $\tau_1^*, \cdots, \tau_{n-1}^*$, respectively. They are formally expressed as below,

$$\forall_{s,a,s'} : \mathcal{C}_{n-1}(s,a,s') = \sum\nolimits_{k=1}^{n-1} \mathcal{N}_k(s,a,s') \tag{6}$$

$$\forall_{s,a,s'} : \mathcal{C}_{n-1}(s) = \sum\nolimits_{k=1}^{n-1} \mathcal{N}_k(s) \tag{7}$$

where $\mathcal{N}_k(s,a,s')$ and $\mathcal{N}_k(s)$ are the visit-counts of (s,a,s') and s in the trajectory τ_k^*, respectively. When $n = 1$, for $\forall_{s,a,s'}$, we have $\mathcal{C}_0(s,a,s') = 0$ and $\mathcal{C}_0(s) = 0$, i.e. $\mathcal{F}_1(s,a,s') = 0$. When $n > 1$, since $\forall_{s,a,s'} : 0 \leq \mathcal{N}_k(s,a,s') \leq \mathcal{N}_k(s)$ in any trajectory τ_k^*, we can get $0 \leq \mathcal{C}_{n-1}(s,a,s')/\mathcal{C}_{n-1}(s) \leq 1$ for task n. Therefore, we obtain $\forall_{s,a,s'} : 0 \leq \mathcal{F}_n(s,a,s') \leq (1-\gamma)V_{\max} \leq R_{\max}$ for each task n. Noted that for each visited state-action pair in the current task, LRS will provide it with a shaped reward based on its visit-counts in previous tasks. In this manner, the LRS function provides timely and informative rewards to effectively accelerate the learning process, especially when the environmental rewards are sparse and delayed. At the same time, the LRS function restricts the additional rewards in an appropriate range, without too much intervention to the agent.

Based on the above elaboration about the LRS function, we attain a new algorithm namely SR-LLRL, which is listed in Algorithm 1. As shown in line 3, we let the Q-function be initialized to $V_{\max}$ in each new sampled MDP, which is equivalent to a particular PBRS function [33], [34], [16]. As shown in line 6, we shape the reward function in each new sampled MDP after updating the LRS function. With this combination, our algorithm can avoid the problem of positive-reward cycles [19]. By only utilizing the optimal trajectories, SR-LLRL motivates the agent's learning without too much extra computation.

Algorithm 1 Shaping Rewards for Lifelong RL

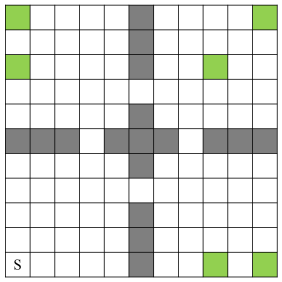
```
Input: task distribution $D$, initial value $V_{\max}$, learning algorithm $\mathscr{A}$, maximum iteration steps $I_{\max}$
1:  Initialize: $\forall_{s,a,s'} : \mathcal{C}_0(s,a,s'),\ \mathcal{C}_0(s) = 0$
2:  for $t = 1, 2, \ldots$ do
3:      Initialize: $\forall_{s,a,s'} : Q_t(s,a) = V_{\max},\ \mathcal{N}_t(s,a,s') = 0,\ \mathcal{N}_t(s) = 0;\ \mathcal{Z}_t = \emptyset$
4:      $\langle \mathcal{S}, \mathcal{A}, \gamma, \mathcal{T}_t, \mathcal{R}_t \rangle \leftarrow$ sample($D$)
5:      Compute $\mathcal{F}_t(s,a,s')$ using Equation (5)
6:      $Q_t(s,a),\ \mathcal{Z}_t \leftarrow \mathscr{A}(\langle \mathcal{S}, \mathcal{A}, \gamma, \mathcal{T}_t, \mathcal{R}_t + \mathcal{F}_t \rangle, I_{\max})$
7:      Select optimal trajectory $\tau_t^*$ from $\mathcal{Z}_t$
8:      Get $\mathcal{N}_t(s,a,s')$ and $\mathcal{N}_t(s)$ from $\tau_t^*$
9:      Compute $\mathcal{C}_t(s,a,s')$ and $\mathcal{C}_t(s)$ using Equations (6) and (7), respectively
10: end for
```

IV. Experiments

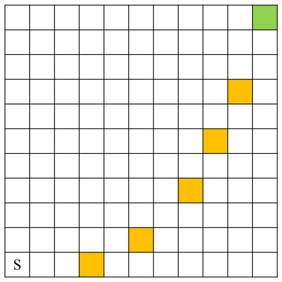
In this section, we evaluate SR-LLRL with MaxQInit [16] and an "Ideal" case in three types of classical grid-world environments. The "Ideal" case is from the evaluation experiments of MaxQInit, in which the initial value for the agent in each sampled task is fixed with the highest value over the (near) optimal Q-values learned from all the tasks of the distribution. In other words, the agent in the "Ideal" case knows the (near) optimal Q-value of every MDP in the true distribution. Our extensive experiments illustrate that SR-LLRL can effectively accelerate the convergence process and achieve superior performance since the very early of lifelong learning. Among them, we take Q-Learning [35] and Delayed Q-Learning [33] (Delayed-Q for short) as baseline learning algorithms in each experiment, respectively.

A. Environment Setup

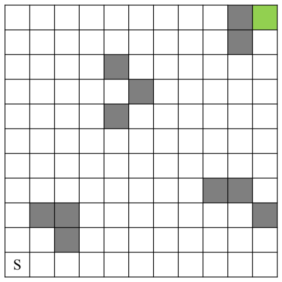
We experiment with three 11×11 grid-world domains: Four Room, Lava, and Maze. Illustrations of environments are plotted in Fig. 1(a), where the agent has four possible actions: "Up", "Down", "Left", and "Right". Four Room is a classical grid-world from [1]. The start state is in the bottom left corner, which is displayed in "S". Each time the task distribution samples a new MDP, with the goal randomly sampled from a target distribution (green cells). The wall permutation (gray cells) remains fixed across all tasks. All rewards are set to 0.0 apart from the transition into the goal state, which yields 1.0. Lava is a distribution with each reward function specifying diverse lava states. The figure plotted in Fig. 1(b) is a task instance of Lava. Not like Four Room, Lava maintains the goal state and adds a reward to each state with lava (orange cells), which is set to 0.01. Maze is a more challenging setting. As plotted in Fig. 1(c), we specified diverse wall locations but fixed start and goal locations. The difference between Maze and Four Room is



(a) Four Room

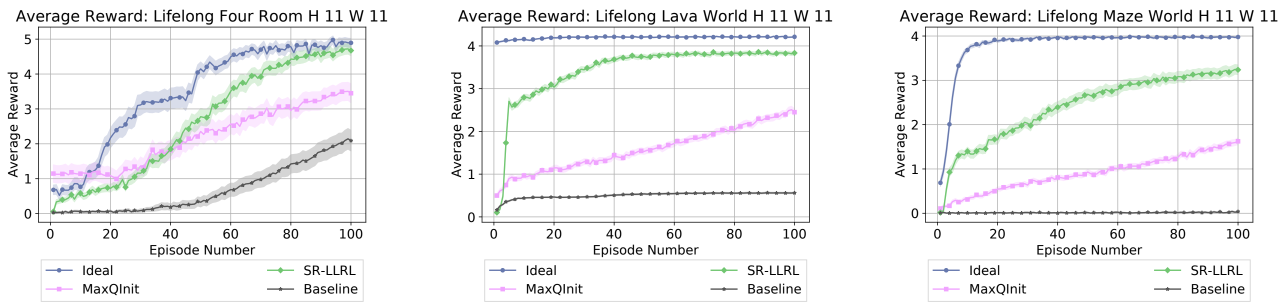


(b) Lava

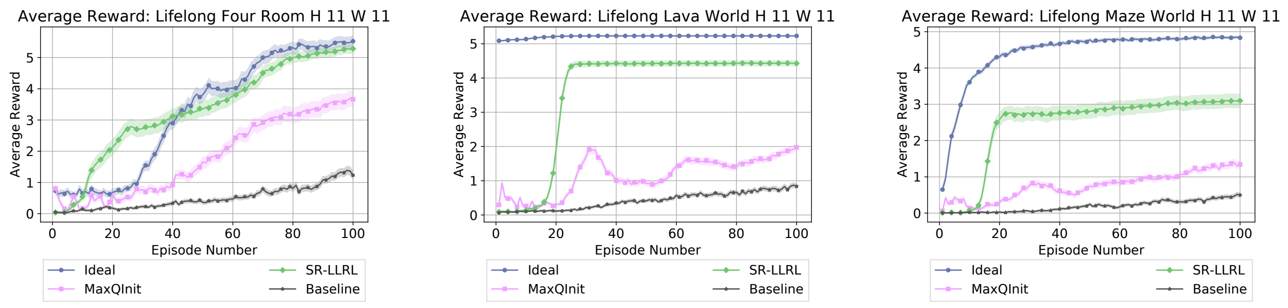


(c) Maze

Fig. 1. Illustration of Four Room, Lava, and Maze grid-world environments. The start state is the cell with "S" displayed in the center. The green, orange and gray cells denote the locations of the targets, lavas, and walls, respectively.



(a) Average reward for each episode (Based on Q-Learning)



(b) Average reward for each episode (Based on Delayed-Q)

Fig. 2. Results of the first experiment. Plots showed reward averaged over 100 tasks for each episode in three different environments.

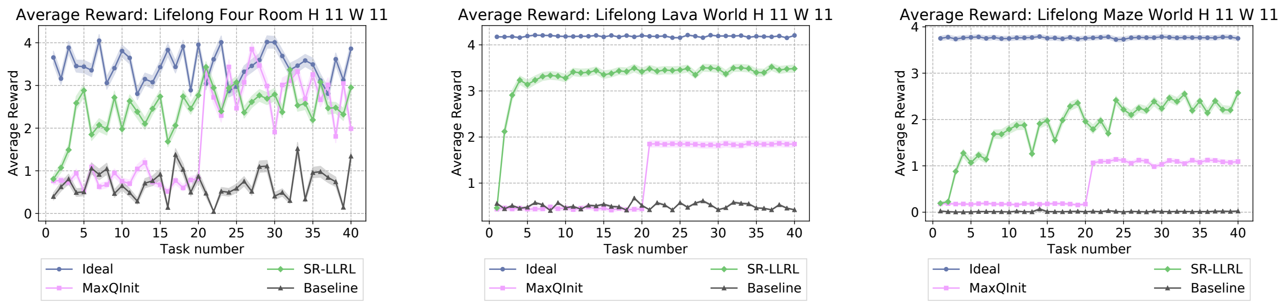
that Four Room randomly assigns a goal while the other elements remain the same, whereas Maze randomly assigns a wall permutation while the goal remains the same.

B. Experiment Results

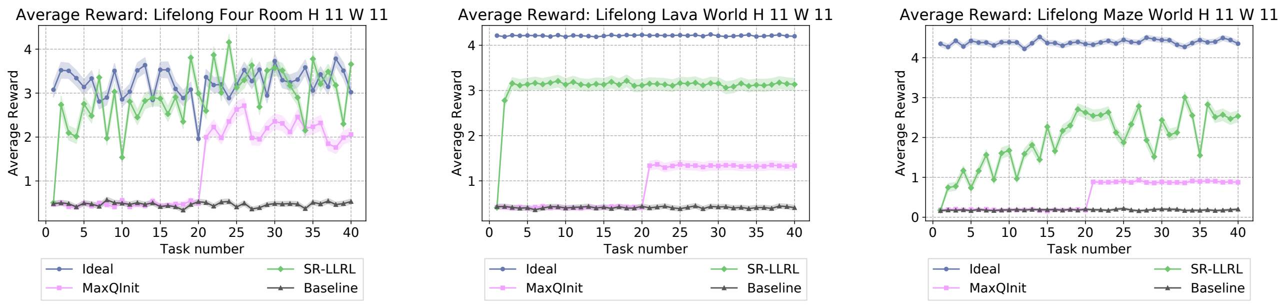
In all the experiments, we set the discount factor $\gamma = 0.95$. During the lifelong learning process, each time the agent interacts with the randomly sampled MDP for 100 episodes with 100 steps per episode. Experimental results are plotted with 95% confidence intervals. When the learner is the Q-Learning agent, we use ϵ-greedy action selection with $\epsilon = 0.1$ and set learning rate $\alpha = 0.1$. When the learner is the Delayed-Q agent, we set the number of experiences m before an update is allowed as 5 and the constant exploration bonus $\epsilon_1 = 0.001$. As for the MaxQInit algorithm, we set the number of tasks required before updating the initial Q-value as 20 in all the experiments.

In the first experiment, we evaluate the relative performance of different methods across 100 tasks. In addition, we take the agent with no transfer as "Baseline" in each experiment. The results are reported in Fig. 2, which displays the reward averaged across all the tasks for each episode. In all the plots, "Ideal" and "Baseline" serve as the upper and lower bounds on performance, respectively. For both Q-Learning and Delayed-Q, SR-LLRL achieved the closest performance to Ideal's, compared to MaxQInit. SR-LLRL has higher learning efficiency and faster convergence speed, which exhibits superior scores in all environments, even in a complex one like Maze.

In the next experiment, we sample 40 tasks for each agent and plot the reward averaged across 100 episodes for each task. The results are presented in Fig 3. We can note



(a) Average reward for each task (Based on Q-Learning)



(b) Average reward for each task (Based on Delayed-Q)

Fig. 3. Results of the second experiment. Plots show reward averaged over 100 episodes for each task in three different environments.

that SR-LLRL can achieve better performance at early of lifelong learning. Across all plots, SR-LLRL has achieved very effective results as early as the second task, especially in Four Room and Lava. Moreover, this good performance is maintained until the end of the experiment. Conversely, MaxQInit does not perform well at first as it requires collecting the knowledge from the first 20 tasks. After that, the rewards on subsequent tasks achieved by MaxQInit are still lower than ours.

In summary, the results have shown that SR-LLRL achieves better learning efficiency and performance compared to MaxQInit. SR-LLRL provides more instructive and informative rewards in new tasks, hence accelerates the convergence process. Moreover, the early performance increase of SR-LLRL showcases the effectiveness of the reward information we extracted from the previous tasks.

V. Conclusion

A key question in Lifelong RL is utilizing the knowledge from previous tasks while solving new tasks. As a pioneering work in this field, MaxQInit [16] initializes the Q-value in the current task as the maximum of Q-values over previously learned tasks, achieving better initial performance and lowering sample complexity than other algorithms. However, delayed and sparse rewards obtained in the learning process dramatically slow down the learning process. In this paper, we propose the SR-LLRL algorithm to address the above question. By constructing additional reward information across tasks, our method effectively speeds up the learning process. Critically, we design a new reward shaping function namely LRS function, to provide timely and informative rewards in the current task based on the optimal trajectories obtained from previous tasks. We showcase our algorithm in Lifelong RL experiments and demonstrate empirically its significantly improved learning efficiency and performance. In the future, we hope to scale our approach to more general environments, like high-dimensional or continuous state-action space.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[2] A. G. Barto, R. S. Sutton, and C. J. C. H. Watkins, *Learning and Sequential Decision Making*. MIT Press, 1989, pp. 539–602.

[3] A. Wilson, A. Fern, S. Ray, and P. Tadepalli, "Multi-task reinforcement learning: a hierarchical bayesian approach." in *Proc. 24th Int. Conf. Mach. Learn.*, Jan. 2007, pp. 1015–1022.

[4] D. Silver, Q. Yang, and L. Li, "Lifelong machine learning systems: Beyond learning algorithms," in *2013 AAAI Spring Symposium Series.*, Mar. 2013, p. 49.

[5] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, no. 10, pp. 1633–1685, 2009.

[6] A. Lazaric, "Transfer in reinforcement learning: A framework and a survey," in *Reinforcement Learning: State-of-the-Art*, M. Wiering and M. van Otterlo, Eds. Berlin, Heidelberg: Springer, 2012, pp. 143–173.

[7] A. Lazaric, M. Restelli, and A. Bonarini, "Transfer of samples in batch reinforcement learning," in *Proc. 25th Int. Conf. Mach. Learn.* New York, NY, USA: Association for Computing Machinery, 2008, pp. 544–551.

[8] M. E. Taylor, N. K. Jong, and P. Stone, "Transferring instances for model-based reinforcement learning," in *Joint European Conf. Mach. Learn. Knowledge Discovery Databases*, W. Daelemans, B. Goethals, and K. Morik, Eds. Berlin, Heidelberg: Springer, 2008, pp. 488–505.

[9] A. Tirinzoni, A. Sessa, M. Pirotta, and M. Restelli, "Importance weighted transfer of samples in reinforcement learning," in *Proc. 35th Int. Conf. Mach. Learn.*, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, Jul. 2018, pp. 4936–4945.

[10] F. Fernández and M. Veloso, "Probabilistic policy reuse in a reinforcement learning agent," in *Proc. 5th Int. Joint Conf. Autonomous Agents & Multiagent Syst.* New York, NY, USA: Association for Computing Machinery, 2006, pp. 720–727.

[11] M. G. Azar, A. Lazaric, and E. Brunskill, "Regret bounds for reinforcement learning with policy advice," *Comput. Sci.*, vol. 8188, pp. 97–112, 2013.

[12] B. Rosman, M. Hawasly, and S. Ramamoorthy, "Bayesian policy reuse," *Mach. Learn.*, vol. 104, no. 1, pp. 99–127, Jul. 2016.

[13] G. Konidaris, I. Scheidwasser, and A. G. Barto, "Transfer in reinforcement learning via shared features," *J. Mach. Learn. Res.*, vol. 13, no. null, pp. 1333–1371, May 2012.

[14] J. Song, Y. Gao, H. Wang, and B. An, "Measuring the distance between finite markov decision processes," in *Proc. 15th Int. Conf. Autonomous Agents & Multiagent Syst.* Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 468–476.

[15] Y. Liu, Z. Guo, and E. Brunskill, "Pac continuous state online multitask reinforcement learning with identification," in *Proc. 15th Int. Conf. Autonomous Agents & Multiagent Syst.*, 2016, pp. 438–446.

[16] D. Abel, Y. Jinnai, S. Y. Guo, G. Konidaris, and M. Littman, "Policy and value transfer in lifelong reinforcement learning," in *Proc. 35th Int. Conf. Mach. Learn.*, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, Jul. 2018, pp. 20–29.

[17] B. Digney, "Learning hierarchical control structure for multiple tasks and changing environments," in *Proc. 5th Conf. Simulation Adaptive Behavior*, vol. 98, 1998, p. 295.

[18] A. McGovern and A. G. Barto, "Automatic discovery of subgoals in reinforcement learning using diverse density," in *Proc. 18th Int. Conf. Mach. Learn.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 361–368.

[19] J. Randløv and P. Alstrøm, "Learning to drive a bicycle using reinforcement learning and shaping," in *Proc. 15th Int. Conf. Mach. Learn.*, 1998, pp. 463–471.

[20] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. 16th Int. Conf. Mach. Learn.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 278–287.

[21] R. Bellman, "A markovian decision process," *J. Math. Mech.*, vol. 6, no. 5, pp. 679–684, 1957.

[22] M. Mahmud, M. Hawasly, B. Rosman, and S. Ramamoorthy, "Clustering markov decision processes for continual transfer," *arXiv preprint arXiv:1311.3959*, 2013.

[23] M. J. Mataric, "Reward functions for accelerated learning," in *Mach. Learn. Proc.*, W. W. Cohen and H. Hirsh, Eds. San Francisco (CA): Morgan Kaufmann, 1994, pp. 181–189.

[24] M. Dorigo and M. Colombetti, "Robot shaping: Developing autonomous agents through learning," *Artif. Intell.*, pp. 321–370, Dec. 1994.

[25] E. Wiewiora, "Potential-based shaping and q-value initialization are equivalent," *J. Artif. Intell. Res.*, vol. 19, pp. 205–208, Jun. 2003.

[26] S. Devlin and D. Kudenko, "Dynamic potential-based reward shaping," in *Proc. 11th Int. Conf. Autonomous Agents & Multiagent Syst.* Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 433–440.

[27] A. Harutyunyan, S. Devlin, P. Vrancx, and A. Nowe, "Expressing arbitrary reward functions as potential-based advice," in *Proc. 29th AAAI Conf. Artif. Intell.* AAAI Press, 2015, pp. 2652–2658.

[28] O. Marom and B. Rosman, "Belief reward shaping in reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3762–3769.

[29] P. Goyal, S. Niekum, and R. J. Mooney, "Using natural language for reward shaping in reinforcement learning," in *Proc. 28th Int. Joint Conf. Artif. Intell.* International Joint Conferences on Artificial Intelligence Organization, Jul. 2019, pp. 2385–2391.

[30] N. Waytowich, S. L. Barton, V. Lawhern, and G. Warnell, "A narration-based reward shaping approach using grounded natural language commands," in *Proc. 36th Int. Conf. Mach. Learn.* PMLR, 2019.

[31] B. Xiao, Q. Lu, B. Ramasubramanian, A. Clark, L. Bushnell, and R. Poovendran, "Fresh: Interactive reward shaping in high-dimensional state spaces using human feedback," in *Proc. 19th Int. Conf. Autonomous Agents & MultiAgent Syst.* Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2020, pp. 1512–1520.

[32] W. Guofang, F. Zhou, L. Ping, and L. Bo, "Shaping in reinforcement learning via knowledge transferred from human-demonstrations," in *34th Chinese Control Conf.* IEEE, 2015, pp. 3033–3038.

[33] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, "Pac model-free reinforcement learning," in *Proc. 23rd Int. Conf. Mach. Learn.* New York, NY, USA: Association for Computing Machinery, 2006, pp. 881–888.

[34] E. Wiewiora, G. Cottrell, and C. Elkan, "Principled methods for advising reinforcement learning agents," in *Pro. 20th Int. Conf. Inte. Conf. Mach. Learn.* AAAI Press, 2003, pp. 792–799.

[35] C. J. C. H. Watkins and P. Dayan, "Q-learning. machine learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 279–292, 1992.