frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 21 June 2018
doi: 10.3389/fnbot.2018.00032

Experience Replay Using Transition Sequences

Thommen George Karimpanal and Roland Bouffanais*

Engineering Product Development, Singapore University of Technology and Design, Singapore, Singapore

Experience replay is one of the most commonly used approaches to improve the sample efficiency of reinforcement learning algorithms. In this work, we propose an approach to select and replay sequences of transitions in order to accelerate the learning of a reinforcement learning agent in an off-policy setting. In addition to selecting appropriate sequences, we also artificially construct transition sequences using information gathered from previous agent-environment interactions. These sequences, when replayed, allow value function information to trickle down to larger sections of the state/state-action space, thereby making the most of the agent's experience. We demonstrate our approach on modified versions of standard reinforcement learning tasks such as the mountain car and puddle world problems and empirically show that it enables faster, and more accurate learning of value functions as compared to other forms of experience replay. Further, we briefly discuss some of the possible extensions to this work, as well as applications and situations where this approach could be particularly useful.

Keywords: experience replay, Q-learning, off-policy, multi-task reinforcement learning, probabilistic policy reuse

OPEN ACCESS

Edited by:
Timothy P. Lillicrap,
Google, United States

Reviewed by:
Eiji Uchibe,
Advanced Telecommunications Research Institute International (ATR), Japan
Önder Tutsoy,
Adana Science and Technology University, Turkey

****Correspondence:***
Thommen George Karimpanal
thommen_george@mymail.sutd.edu.sg

Received: *16 March 2018*
Accepted: *31 May 2018*
Published: *21 June 2018*

Citation:
Karimpanal TG and Bouffanais R (2018) Experience Replay Using Transition Sequences. Front. Neurorobot. 12:32. doi: 10.3389/fnbot.2018.00032

1. INTRODUCTION

Real-world artificial agents ideally need to be able to learn as much as possible from their interactions with the environment. This is especially true for mobile robots operating within the reinforcement learning (RL) framework, where the cost of acquiring information from the environment through exploration generally exceeds the computational cost of learning (Adam et al., 2012; Schaul et al., 2016; Wang et al., 2016).

Experience replay (Lin, 1992) is a technique that reuses information gathered from past experiences to improve the efficiency of learning. In order to replay stored experiences using this approach, an off-policy (Sutton and Barto, 1998; Geist and Scherrer, 2014) setting is a prerequisite. In off-policy learning, the policy that dictates the agent's control actions is referred to as the behavior policy. Other policies corresponding to the value/action-value functions of different tasks that the agent aims to learn are referred to as target policies. Off-policy algorithms utilize the agent's behavior policy to interact with the environment, while simultaneously updating the value functions associated with the target policies. These algorithms can hence be used to parallelize learning, and, thus gather as much knowledge as possible using real experiences (Sutton et al., 2011; White et al., 2012; Modayil et al., 2014). However, when the behavior and target policies differ considerably from each other, the actions executed by the behavior policy may only seldom correspond to those recommended by the target policy. This could lead to poor estimates of the corresponding value function. Such cases could arise in multi-task scenarios where multiple tasks are learned in an off-policy manner. Also, in general, in environments where desirable experiences are rare occurrences, experience replay could be employed to improve the estimates by storing and replaying transitions (state, actions, and rewards) from time to time.

Although most experience replay approaches store and reuse individual transitions, replaying sequences of transitions could offer certain advantages. For instance, if a value function update following a particular transition results in a relatively large change in the value of the corresponding state or state-action pair, this change will have a considerable influence on the bootstrapping targets of states or state-action pairs that led to this transition. Hence, the effects of this change should ideally be propagated to these states or state-action pairs. If instead of individual transitions, sequences of transitions are replayed, this propagation can be achieved in a straightforward manner. Our approach aims to improve the efficiency of learning by replaying transition sequences in this manner. The sequences are selected on the basis of the magnitudes of the temporal difference (TD) errors (Sutton and Barto, 1998), associated with them. We hypothesize that selecting sequences that contain transitions associated with higher magnitudes of TD errors allow considerable learning progress to take place. This is enabled by the propagation of the effects of these errors to the values associated with other states or state-action pairs in the transition sequence.

Replaying a larger variety of such sequences would result in a more efficient propagation of the mentioned effects to other regions in the state/state-action space. Hence, in order to aid the propagation in this manner, other sequences that could have occurred are artificially constructed by comparing the state trajectories of previously observed sequences. These virtual transition sequences are appended to the replay memory, and they help bring about learning progress in other regions of the state/state-action space when replayed.

The generated transition sequences are virtual in the sense that they may have never occurred in reality, but are constructed from sequences that have actually occurred in the past. The additional replay updates corresponding to the mentioned transition sequences supplement the regular off-policy value function updates that follow the real-world execution of actions, thereby making the most out of the agent's interactions with the environment.

2. RELATED WORK

The problem of learning from limited experience is not new in the field of RL (Thrun, 1992; Thomas and Brunskill, 2016). Generally, learning speed and sample efficiency are critical factors that determine the feasibility of deploying learning algorithms in the real world. Particularly for robotics applications, these factors are even more important, as exploration of the environment is typically time and energy expensive (Bakker et al., 2006; Kober et al., 2013). It is thus important for a learning agent to be able to gather as much relevant knowledge as possible from whatever exploratory actions occur.

Off-policy algorithms are well suited to this need as it enables multiple value functions to be learned together in parallel. When the behavior and target policies vary considerably from each other, importance sampling (Sutton and Barto, 1998; Rubinstein and Kroese, 2016) is commonly used in order to obtain more accurate estimates of the value functions. Importance sampling reduces the variance of the estimate by taking into account the distributions associated with the behavior and target policies, and making modifications to the off-policy update equations accordingly. However, the estimates are still unlikely to be close to their optimal values if the agent receives very little experience relevant to a particular task.

This issue is partially addressed with experience replay, in which information contained in the replay memory is used from time to time in order to update the value functions. As a result, the agent is able to learn from uncorrelated historical data, and the sample efficiency of learning is greatly improved. This approach has received a lot of attention in recent years due to its utility in deep RL applications (Adam et al., 2012; Mnih et al., 2013, 2015, 2016; de Bruin et al., 2015).

Recent works (Narasimhan et al., 2015; Schaul et al., 2016) have revealed that certain transitions are more useful than others. Schaul et al. (2016) prioritized transitions on the basis of their associated TD errors. They also briefly mentioned the possibility of replaying transitions in a sequential manner. The experience replay framework developed by Adam et al. (2012) involved some variants that replayed sequences of experiences, but these sequences were drawn randomly from the replay memory. More recently, Isele et al. (Isele and Cosgun, 2018) reported a selective experience replay approach aimed at performing well in the context of lifelong learning (Thrun, 1996). The authors of this work proposed a long term replay memory in addition to the conventionally used one. Certain bases for designing this long-term replay memory, such as favoring transitions associated with high rewards and high absolute TD errors are similar to the ones described in the present work. However, the approach does not explore the replay of sequences, and its fundamental purpose is to shield against catastrophic forgetting (Goodfellow et al., 2013) when multiple tasks are learned in sequence. The replay approach described in the present work focuses on enabling more sample-efficient learning in situations where positive rewards occur rarely. Apart from this, Andrychowicz et al. (2017) proposed a hindsight experience replay approach, directed at addressing this problem, where each episode is replayed with a goal that is different from the original goal of the agent. The authors reported significant improvements in the learning performance in problems with sparse and binary rewards. These improvements were essentially brought about by allowing the learned value/Q values (which would otherwise remain mostly unchanged due to the sparsity of rewards) to undergo significant change under the influence of an arbitrary goal. The underlying idea behind our approach also involves modification of the $Q-$values in reward-sparse regions of the state-action space. The modifications, however, are not based on arbitrary goals, and are selectively performed on state-action pairs associated with successful transition sequences associated with high absolute TD errors. Nevertheless, the hindsight replay approach is orthogonal to our proposed approach, and hence, could be used in conjunction with it.

Much like in Schaul et al. (2016), TD errors have been frequently used as a basis for prioritization in other RL problems (Thrun, 1992; White et al., 2014; Schaul et al., 2016). In particular,

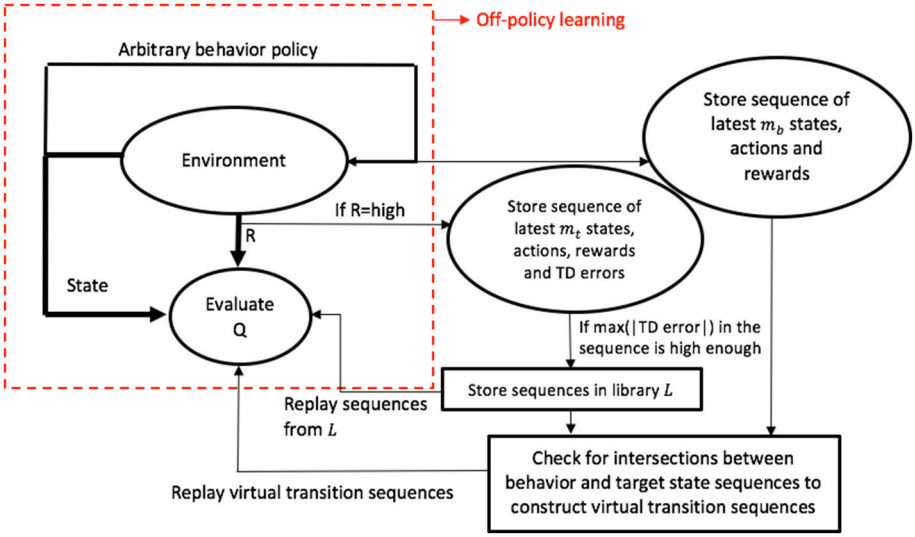

FIGURE 1 | Structure of the proposed algorithm in contrast to the traditional off-policy structure. *Q* and *R* denote the action-value function and reward, respectively.

the model-based approach of prioritized sweeping (Moore and Atkeson, 1993; van Seijen and Sutton, 2013) prioritizes backups that are expected to result in a significant change in the value function.

The algorithm we propose here uses a model-free architecture, and it is based on the idea of selectively reusing previous experience. However, we describe the reuse of sequences of transitions based on the TD errors observed when these transitions take place. Replaying sequences of experiences also seems to be biologically plausible (Buhry et al., 2011; Ólafsdóttir et al., 2015). In addition, it is known that animals tend to remember experiences that lead to high rewards (Singer and Frank, 2009). This is an idea reflected in our work, as only those transition sequences that lead to high rewards are considered for being stored in the replay memory. In filtering transition sequences in this manner, we simultaneously address the issue of determining which experiences are to be stored.

In addition to selecting transition sequences, we also generate virtual sequences of transitions which the agent could have possibly experienced, but in reality, did not. This virtual experience is then replayed to improve the agent's learning. Some early approaches in RL, such as the dyna architecture (Sutton, 1990) also made use of simulated experience to improve the value function estimates. However, unlike the approach proposed here, the simulated experience was generated based on models of the reward function and transition probabilities which were continuously updated based on the agent's interactions with the environment. In this sense, the virtual experience generated in our approach is more grounded in reality, as it is based directly on the data collected through the agent-environment interaction. In more recent work, Fonteneau et al. describe an approach to generate artificial trajectories and use them to find policies with acceptable performance guarantees (Fonteneau et al., 2013). However, this approach is designed for batch RL, and the generated artificial trajectories are not constructed using a TD error basis. Our approach also recognizes the real-world limitations of replay memory (de Bruin et al., 2015), and stores only a certain amount of information at a time, specified by memory parameters. The selected and generated sequences are stored in the replay memory in the form of libraries which are continuously updated so that the agent is equipped with transition sequences that are most relevant to the task at hand.

3. METHODOLOGY

The idea of selecting appropriate transition sequences for replay is relatively straightforward. In order to improve the agent's learning, first, we simply keep track of the state, actions, rewards, and absolute values of the TD errors associated with each transition. Generally, in difficult learning environments, high rewards occur rarely. So, when such an event is observed, we consider storing the corresponding sequence of transitions into a replay library L. In this manner, we use the reward information as a means to filter transition sequences. The approach is similar to that used by Narasimhan et al. (2015), where transitions associated with positive rewards are prioritized for replay.

Among the transition sequences considered for inclusion in the library L, those containing transitions with high absolute TD error values are considered to be the ones with high potential for learning progress. Hence, they are accordingly prioritized for replay. The key idea is that when the TD error associated with a particular transition is large in magnitude, it generally implies a proportionately greater change in the value of the corresponding state/state-action pair. Such large changes have the potential to influence the values of the states/state-action pairs leading to it, which implies a high potential for learning. Hence, prioritizing such sequences of transitions for replay is likely to bring about

greater learning progress. Transition sequences associated with large magnitudes of TD error are retained in the library, while those with lower magnitudes are removed and replaced with superior alternatives. In reality, such transition sequences may be very long and hence, impractical to store. Due to such practical considerations, we store only a portion of the sequence, based on a predetermined memory parameter. The library is continuously updated as and when the agent-environment interaction takes place, such that it will eventually contain sequences associated with the highest absolute TD errors.

As described earlier, replaying suitable sequences allows the effects of large changes in value functions to be propagated throughout the sequence. In order to propagate this information even further to other regions of the state/state-action space, we use the sequences in L to construct additional transition sequences which could have possibly occurred. These virtual sequences are stored in another library L_v, and later used for experience replay.

In order to intuitively describe our approach of artificially constructing sequences, we consider the hypothetical example shown in **Figure 2A**, where an agent executes behavior policies that help it learn to navigate toward location B from the start location. However, using off-policy learning, we aim to learn value functions corresponding to the policy that helps the agent navigate toward location T.

The trajectories shown in **Figure 2A** correspond to hypothetical actions dictated by the behavior policy midway through the learning process, during two separate episodes. The trajectories begin at the start location and terminate at location B. However, the trajectory corresponding to behavior policy 2 also happens to pass through location T, at which point the agent receives a high reward. This triggers the transition sequence storage mechanism described earlier, and we assume that some portion of the sequence (shown by the highlighted portion of the trajectory in **Figure 2A**) is stored in library L. Behavior policy 1 takes the agent directly from the start location toward the location B, where it terminates. As the agent moves along its trajectory, it intersects with the state trajectory corresponding to the sequence stored in L. Using this intersection, it is possible to artificially construct additional trajectories (and their associated transition sequences) that are successful with respect to the task of navigating to location T. The highlighted portions of the trajectories corresponding to the two behavior policies in **Figure 2B** show such a state trajectory, constructed using information related to the intersection of portions of the two previously observed trajectories. The state, action, and reward sequences associated with this highlighted trajectory form a virtual transition sequence.

Such artificially constructed transition sequences present the possibility of considerable learning progress. This is because, when replayed, they help propagate the large learning potential (characterized by large magnitudes of TD errors) associated with sequences in L to other regions of the state/state-action space. These replay updates supplement the off-policy value function updates that are carried out in parallel, thus accelerating the learning of the task in question. This outlines the basic idea behind our approach.

Fundamentally, our approach can be decomposed into three steps:

1. Tracking and storage of relevant transition sequences
2. Construction of virtual transition sequences using the stored transition sequences
3. Replaying the transition sequences

These steps are explained in detail in sections 3.1, 3.2, and 3.3.

3.1. Tracking and Storage of Relevant Transition Sequences

As described, virtual transition sequences are constructed by joining together two transition sequences. One of them, say Θ_t, composed of m_t transitions, is historically successful—it has experienced high rewards with respect to the task, and is part of the library L. The other sequence, Θ_b, is simply a sequence of the latest m_b transitions executed by the agent.

If the agent starts at state s_0 and moves through intermediate states s_i and eventually to s_{j+1} (most recent state) by executing a series of actions $a_0...a_i...a_j$, it receives rewards $R_0...R_i...R_j$ from the environment. These transitions comprise the transition sequence Θ_b.

$$\Theta_b = \begin{cases} [S(0:j) \quad \pi(0:j) \quad R(0:j)] & \text{if} \quad j \leq m_b \\ [S((j-m_b):j) \quad \pi((j-m_b):j) \quad R((j-m_b):j)] & \text{otherwise} \end{cases} \tag{1}$$

where:

$$S(x:y) = (s_x...s_i...s_y),$$
$$\pi(x:y) = (a_x...a_i...a_y),$$
$$R(x:y) = (R_x...R_i...R_y).$$

We respectively refer to $S(x:y)$, $\pi(x:y)$, and $R(x:y)$ as the state, action, and reward transition sequences corresponding to a series of agent-environment interactions, indexed from x to y ($x, y \in \mathbb{N}$).

$$\Theta_t = \begin{cases} [S(0:k) \quad \pi(0:k) \quad R(0:k) \quad \Delta(0:k)] & \text{if} \quad k \leq m_t \\ [S((k-m_t):k) \quad \pi((k-m_t):k) \quad R((k-m_t):k) \quad \Delta((k-m_t):k)] & \text{otherwise} \end{cases} \tag{2}$$

For the case of the transition sequence Θ_t, we keep track of the sequence of TD errors $\delta_0...\delta_i...\delta_k$ observed as well. If a high reward is observed in transition k, then:
where $\Delta(x:y) = (|\delta_x|...|\delta_i|...|\delta_y|)$.

The memory parameters m_b and m_t are chosen based on the memory constraints of the agent. They determine how much of the recent agent-environment interaction history is to be stored in memory.

It is possible that the agent encounters a number of transitions associated with high rewards while executing the behavior policy.

Corresponding to these transitions, a number of successful transition sequences Θ_t would also exist. These sequences are maintained in the library L in a manner similar to the *Policy Library through Policy Reuse (PLPR)* algorithm (Fernández and Veloso, 2005). To decide whether to include a new transition sequence $\Theta_{t_{\text{new}}}$ into the library L, we determine the maximum value of the absolute TD error sequence Δ corresponding to $\Theta_{t_{\text{new}}}$ and check whether it is τ-close—the parameter τ determines the exclusivity of the library—to the maximum of the corresponding values associated with the transition sequences in L. If this is the case, then $\Theta_{t_{\text{new}}}$ is included in L. Since the transition sequences are filtered based on the maximum of the absolute values of TD errors among all the transitions in a sequence, this approach should be able to mitigate problems stemming from low magnitudes of TD errors associated with local optima (Baird, 1999; Tutsoy and Brown, 2016b). Using the absolute TD error as a basis for selection, we maintain a fixed number (l) of transition sequences in the library L. This ensures that the library is continuously updated with the latest transition sequences associated with the highest absolute TD errors. The complete algorithm is illustrated in Algorithm 1.

3.2. Virtual Transition Sequences

Once the transition sequence Θ_b is available and a library L of successful transition sequences Θ_t is obtained, we use this information to construct a library L_v of virtual transition sequences Θ_v. The virtual transition sequences are constructed by first finding points of intersection s_c in the state transition sequences of Θ_b and the Θ_t's in L.

Let us consider the transition sequence Θ_b:

$$\Theta_b = [S(x:y) \quad \pi(x:y) \quad R(x:y)],$$

Algorithm 1 Maintaining a replay library of transition sequences

Inputs:
τ : Parameter that determines the exclusivity of the library
l: Parameter that determines the number of transition sequences allowed in the library
Δ_k : Sequence of absolute TD errors corresponding to a transition sequence Θ_k
$L = \{\Theta_{t0}...\Theta_{ti}...\Theta_{tm}\}$: A library of transition sequences ($m \leq l$)
$\Theta_{t_{\text{new}}}$: New transition sequence to be evaluated
$W_{\text{new}} = \max(\Delta_{t_{\text{new}}})$
for $j = 1 : m$ **do**
  $W_j = \max(\Delta_{tj})$
end for
if $W_{\text{new}} * \tau > \max(W)$ **then**
  $L = L \cup \{\Theta_{t_{\text{new}}}\}$
  n_t =Number of transition sequences in L
  if $n_t > l$ **then**
    $L = \{\Theta_{t n_t - l}...\Theta_{ti}...\Theta_{t n_t}\}$
  end if
end if

and a transition sequence Θ_t:

$$\Theta_t = [S(x':y') \quad \pi(x':y') \quad R(x':y') \quad \Delta(x':y')],$$

Let Θ_{t_s} be a sub-matrix of Θ_t such that:

$$\Theta_{t_s} = [S(x':y') \quad \pi(x':y') \quad R(x':y')], \tag{3}$$

Now, if σ_x^y and $\sigma_{x'}^{y'}$ are sets containing all the elements of sequences $S(x:y)$ and $S(x':y')$, respectively, and if $\exists s_c \in \{\sigma_x^y \cap \sigma_{x'}^{y'}\}$, then:

$$S(x:y) = (s_x, ...s_c, s_{c+1}, ...s_y),$$

and

$$S(x':y') = (s_{x'}...s_c, s_{c+1}...s_{y'}).$$

Once points of intersection have been obtained as described above, each of the two sequences Θ_b and Θ_{t_s} are decomposed into two subsequences at the point of intersection such that:

$$\Theta_b = \begin{bmatrix} \Theta_b^1 \\ \Theta_b^2 \end{bmatrix} \tag{4}$$

where $\Theta_b^1 = [S(x:c) \quad \pi(x:c) \quad R(x:c)]$
and $\Theta_b^2 = [S((c+1):y) \quad \pi((c+1):y) \quad R((c+1):y)]$
Similarly,

$$\Theta_{t_s} = \begin{bmatrix} \Theta_{t_s}^1 \\ \Theta_{t_s}^2 \end{bmatrix} \tag{5}$$

where
$\Theta_{t_s}^1 = [S(x':c) \quad \pi(x':c) \quad R(x':c)]$
and
$\Theta_{t_s}^2 = [S((c+1):y') \quad \pi((c+1):y') \quad R((c+1):y')]$
The virtual transition sequence is then simply:

$$\Theta_v = \begin{bmatrix} \Theta_b^1 \\ \Theta_{t_s}^2 \end{bmatrix} \tag{6}$$

We perform the above procedure for each transition sequence in L to obtain the corresponding virtual transition sequences Θ_v. These virtual transition sequences are stored in a library L_v:

$$L_v = \{\Theta_{v1}...\Theta_{vi}...\Theta_{vn_v}\},$$

where n_v denotes the number of virtual transition sequences in L_v, subjected to the constraint $n_v \leq l$.

The overall process for constructing and storing virtual transition sequences is summarized in Algorithm 2. Once the library L_v has been constructed, we replay the sequences contained in it to improve the estimates of the value function. The details of this are discussed in section 3.3.

Algorithm 2 Constructing virtual transition sequences

Inputs:
Sequence of latest m_b transitions Θ_b
Library L containing n_t stored transition sequences
Library L_v for storing virtual transition sequences
for $t = 1 : n_t$ **do**
  Extract Θ_{t_s} from Θ_t (Equation 2)
  Find set of states S_I corresponding to the intersection of the state trajectories of Θ_b and Θ_{t_s}
  if S_I is not empty, **then**
    for each state s_i in S_I, **do**
      Treat s_i as the intersection point and decompose Θ_b and Θ_{t_s} as per Equations 4 and 5
    end for
    Choose s_c from S_I such that the number of transitions in Θ_b^1 is maximized
  end if
  Use the selected s_c to construct the virtual transition sequence Θ_v as per Equation 6
  Use library L_v to store the constructed sequence ($L_v = L_v \cup \{\Theta_v\}$)
end for

3.3. Replaying the Transition Sequences

In order to make use of the transition sequences described, each of the state-action-reward triads $\{s \quad a \quad r\}$ in the transition sequence L_v is replayed as if the agent had actually experienced them.

Similarly, sequences in L are also be replayed from time to time. Replaying sequences from L and L_v in this manner causes the effects of large absolute TD errors originating from further up in the sequence to propagate through the respective transitions, ultimately leading to more accurate estimates of the value function. The transitions are replayed as per the standard Q-learning update equation shown below:

$$Q(s_j, a_j) \leftarrow Q(s_j, a_j) + \alpha[R(s_j, a_j) + \gamma \max_{a'} Q(s_{j+1}, a') - Q(s_j, a_j)]. \tag{7}$$

Where s_j and a_j refer to the state and action at transition j, and Q and R represent the action-value function and reward corresponding to the task. The variable a' is a bound variable that represents any action in the action set $\mathcal{A}$. The learning rate and discount parameters are represented by α and γ respectively.

The sequence Θ_{t_s} in Equation (6) is a subset of Θ_t, which is in turn part of the library L and thus associated with a high absolute TD error. When replaying Θ_v, the effects of the high absolute TD errors propagate from the values of state/state-action pairs in $\Theta_{t_s}^2$ to those in Θ_b^1. Hence, in case of multiple points of intersection, we consider points that are furthest down Θ_b. In other words, the intersection point is chosen to maximize the length of Θ_b^1. In this manner, a larger number of state-action values experience improvements brought about by replaying the transition sequences.

Algorithm 3 Replay of virtual transition sequences from library L_v

Inputs:
α : learning rate
γ : discount factor
$L_v = \{\Theta_{v0} ... \Theta_{vi} ... \Theta_{vn_v}\}$: A library of virtual transition sequences with n_v sequences
for $i = 1 : n_v$ **do**
  n_{sar} =number of $\{s \quad a \quad r\}$ triads in Θ_{vi}
  $j = 1$
  while $j \leq n_{sar}$ **do**
    $Q(s_j, a_j) \leftarrow Q(s_j, a_j) + \alpha[R(s_j, a_j) + \gamma \max_{a'} Q(s_{j+1}, a') - Q(s_j, a_j)]$
    $j \leftarrow j + 1$
  end while
end for

4. RESULTS AND DISCUSSION

We demonstrate our approach on modified versions of two standard reinforcement learning tasks. The first is a multi-task navigation/puddle-world problem (**Figure 3**), and the second is a multi-task mountain car problem (**Figure 6**). In both these problems, behavior policies are generated to solve a given task (which we refer to as the primary task) relatively greedily, while the value function for another task of interest (which we refer to as the secondary task) is simultaneously learned in an off-policy manner. The secondary task is intentionally made more difficult by making appropriate modifications to the environment. Such adverse multi-task settings best demonstrate the effectiveness of our approach and emphasize its advantages over other experience replay approaches. We characterize the difficulty of the secondary task with a difficulty ratio ρ, which is the fraction of the executed behavior policies that experience a high reward with respect to the secondary task. A low value of ρ indicates that achieving the secondary task under the given behavior policy is difficult. In both tasks, the $Q-$ values are initialized with random values, and once the agent encounters the goal state of the primary task, the episode terminates.

4.1. Navigation/Puddle-World Task

In the navigation environment, the simulated agent is assigned tasks of navigating to certain locations in its environment. We consider two locations, B and T, which represent the primary and secondary task locations respectively. The environment is set up such that the location corresponding to high rewards with respect to the secondary task lies far away from that of the primary task (see **Figure 3**). In addition to this, the accessibility to the secondary task location is deliberately limited by surrounding it with obstacles on all but one side. These modifications contribute toward a low value of ρ, especially when the agent operates with a greedy behavior policy with respect to the primary task.

The agent is assumed to be able to sense its location in the environment accurately, and can detect when it "bumps" into an obstacle. It can move around in the environment at a maximum speed of 1 unit per time step by executing actions to

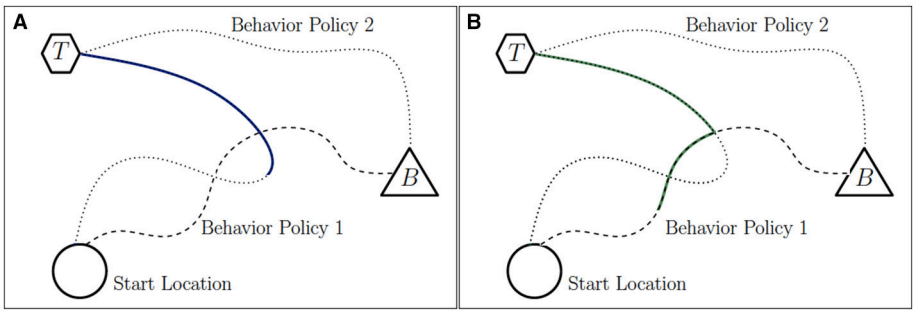

FIGURE 2 | (A) Trajectories corresponding to two hypothetical behavior policies are shown. A portion of the trajectory associated with a high reward (and stored in *L*) is highlighted. **(B)** The virtual trajectory constructed from the two behavior policies is highlighted. The states, actions, and rewards associated with this trajectory constitute a virtual transition sequence.

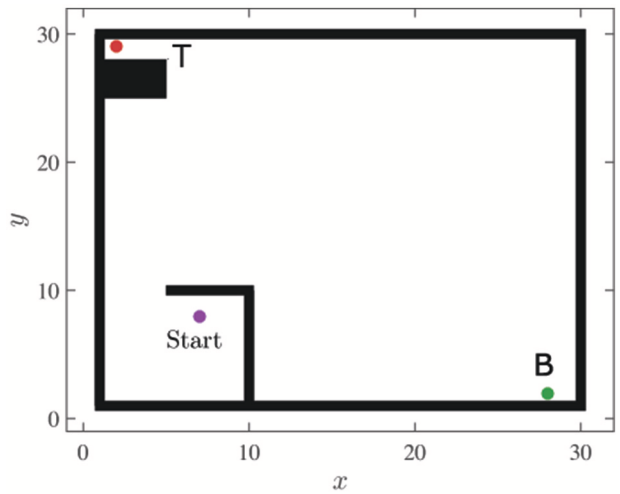

FIGURE 3 | Navigation environment used to demonstrate the approach of replaying transition sequences.

take it forwards, backwards, sideways, and diagonally forwards or backwards to either side. In addition to these actions, the agent can choose to hold its current position. However, the transitions resulting from these actions are probabilistic in nature. The intended movements occur only 80% of the time, and for the remaining 20%, the x- and y-coordinates may deviate from their intended values by 1 unit. Also, the agent's location does not change if the chosen action forces it to run into an obstacle.

The agent employs Q-learning with a relatively greedy policy ($\epsilon = 0.1$) that attempts to maximize the expected sum of primary rewards. The reward structure for both tasks is such that the agent receives a high reward (100) for visiting the respective goal locations, and a high penalty (−100) for bumping into an obstacle in the environment. In addition to this, the agent is assigned a living penalty (−10) for each action that fails to result in the goal state. In all simulations, the discount factor γ is set to be 0.9, the learning rate α is set to a value of 0.3 and the parameter τ mentioned in Algorithm 1 is set to be 1. Although various approaches exist to optimize the values of the $Q-$learning hyperparameters (Garcia and Ndiaye, 1998; Even-Dar and Mansour, 2003; Tutsoy and Brown, 2016a), the values were chosen arbitrarily, such that satisfactory performances were obtained for both the navigation as well as the mountain-car environments.

In the environment described, the agent executes actions to learn the primary task. Simultaneously, the approach described in section 3 is employed to learn the value functions associated with the secondary task. At each episode of the learning process, the agent's performance with respect to the secondary task is evaluated. In order to compute the average return for an episode, we allow the agent to execute $n_{ga}(= 100)$ greedy actions from a randomly chosen starting point, and record the accumulated reward. The process is repeated for $n_{trials}(= 100)$ trials, and the average return for the episode is reported as the average accumulated reward per trial. The average return corresponding to each episode in **Figure 4** is computed in this way. The mean of these average returns over all the episodes is reported as G_e in **Table 1**. That is, the average return corresponding to the k^{th}episode g_k is given by:

$$g_k = \frac{\sum_{i=1}^{n_{trials}} \sum_{j=1}^{n_{ga}} R_{ij}}{n_{trials}}$$

and

$$G_e = \frac{\sum_{k=1}^{N_E} g_k}{N_E}$$

Where R_{ij} is the reward obtained by the agent in a step corresponding to the greedy action j, in trial i, and N_E is the maximum number of episodes.

Figure 4 shows the average return for the secondary task plotted for 50 runs of 1,000 learning episodes using different learning approaches. The low average value of ρ (= 0.0065 as indicated in **Figure 4**) indicates the relatively high difficulty of the secondary task under the behavior policy being executed. As observed in **Figure 4**, an agent that replays transition sequences

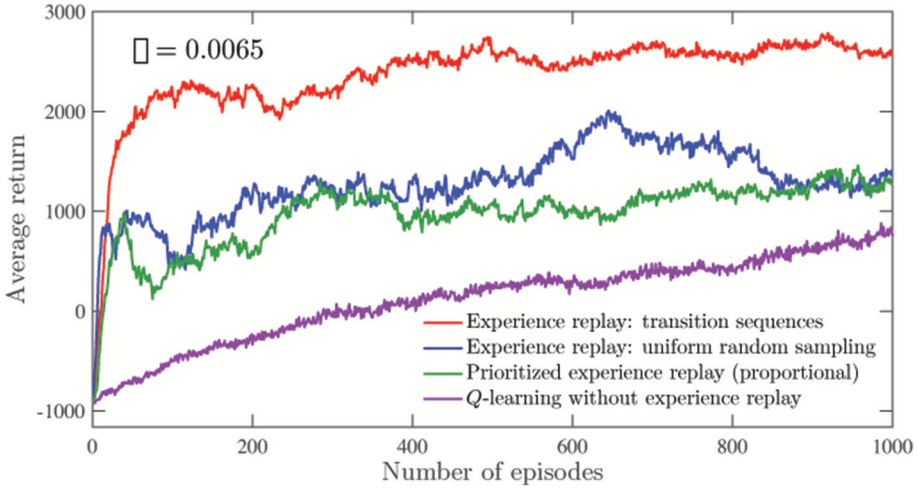

FIGURE 4 | Comparison of the average secondary returns over 50 runs using different experience replay approaches as well as Q-learning without experience replay in the navigation environment. The standard errors are all <300. For the different experience replay approaches, the number of replay updates are controlled to be the same.

TABLE 1 | Average secondary returns accumulated per episode (G_e) using different values of the memory parameters in the navigation environment.

A	
m_b	G_e
10	1559.7
100	2509.7
1,000	2610.4
B	
m_t	G_e
10	1072.5
100	1159.2
1,000	2610.4
C	
n_v	G_e
10	2236.6
50	2610.4
100	2679.5

With regular Q-learning (without experience replay), G_e = 122.9

manages to accumulate high average returns at a much faster rate as compared to regular Q-learning. The approach also performs better than other experience replay approaches for the same number of replay updates. These replay approaches are applied independently of each other for the secondary task. In **Figure 4**, the prioritization exponent for prioritized experience replay is set to 1.

Table 1 shows the average return for the secondary task accumulated per episode (G_e) during 50 runs of the navigation task for different values of memory parameters m_b, m_t and n_v used in our approach. Each of the parameters are varied separately while keeping the other parameters fixed to their default values. The default values used for m_b, m_t, and n_v are 1,000, 1,000, and 50, respectively.

Application to the Primary Task

In the simulations described thus far, the performance of our approach was evaluated on a secondary task, while the agent executed actions relatively greedily with respect to a primary task. Such a setup was chosen in order to ensure a greater sparsity of high rewards for the secondary task. However, the proposed approach of replaying sequences of transitions can also be applied to the primary task in question. In particular, when a less greedy exploration strategy is employed (that is, when ϵ is high), such conditions of reward-sparsity can be recreated for the primary task. **Figure 5** shows the performance of different experience replay approaches when applied to the primary task, for different values of ϵ. As expected, for more exploratory behavior policies, which correspond to lower probabilities of obtaining high rewards, the approach of replaying transition sequences is significantly beneficial, especially at the early stages of learning. However, as the episodes progress, the effects of drastically large absolute TD errors would have already penetrated into other regions of the state-action space, and the agent ceases to benefit as much from replaying transition sequences. Hence, other forms of replay such as experience replay with uniform random sampling, or prioritized experience replay were found to be more useful after the initial learning episodes.

4.2. Mountain Car Task

In the mountain car task, the agent, an under-powered vehicle represented by the circle in **Figure 6** is assigned a primary task of getting out of the trough and visiting point *B*. The act of visiting point *T* is treated as the secondary task. The agent is assigned a high reward (100) for for fulfilling the respective objectives, and a living penalty (−1) is assigned for all other

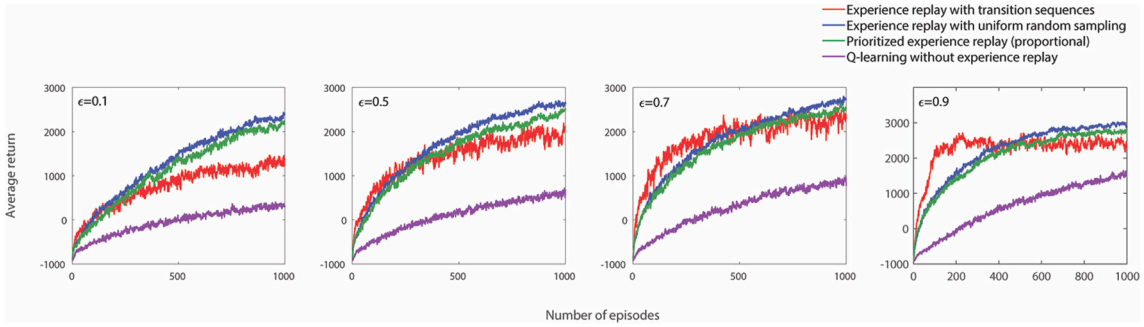

FIGURE 5 | The performance of different experience replay approaches on the primary task in the navigation environment for different values of the exploration parameter ϵ, averaged over 30 runs. For these results, the memory parameters used are as follows: $m_b = 1,000$, $m_t = 1,000$, and $n_v = 50$.

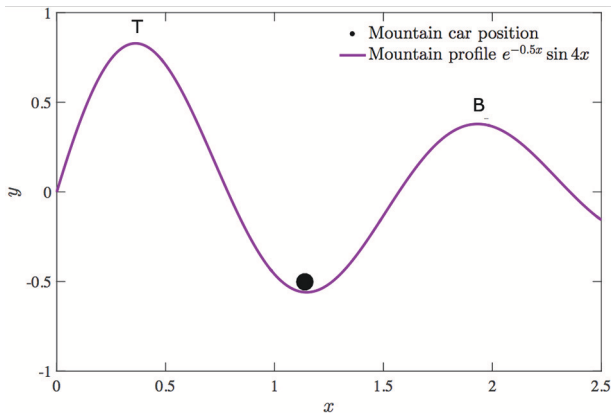

FIGURE 6 | Mountain car environment used to demonstrate off-policy learning using virtual transition sequences.

situations. At each time step, the agent can choose from three possible actions: (1) accelerating in the positive x direction, (2) accelerating in the negative x direction, and (3) applying no control. The environment is discretized such that 120 unique positions and 100 unique velocity values are possible.

The mountain profile is described by the equation $y = e^{-0.5x}\sin(4x)$ such that point T is higher than B. Also, the average slope leading to T is steeper than that leading to B. In addition to this, the agent is set to be relatively greedy with respect to the primary task, with an exploration parameter $\epsilon = 0.1$. These factors make the secondary task more difficult, resulting in a low value of ρ ($= 0.0354$) under the policy executed.

Figure 7 shows the average secondary task returns for 50 runs of 5,000 learning episodes. It is seen that especially during the initial phase of learning, the agent accumulates rewards at a higher rate as compared to other learning approaches. As in the navigation task, the number of replay updates are restricted to be the same while comparing the different experience replay approaches in **Figure 7**. Analogous to **Table 1**, **Table 2** shows the average secondary returns accumulated per episode (G_e) over 50 runs in the mountain-car environment, for different values of the memory parameters. The default values for m_b, m_t, and n_v are the same as those mentioned in the navigation environment, that is, 1,000, 1,000, and 50, respectively.

From **Figures 4**, **7**, the agent is seen to be able to accumulate significantly higher average secondary returns per episode when experiences are replayed. Among the experience replay approaches, the approach of replaying transition sequences is superior for the same number of replay updates. This is especially true in the navigation environment, where visits to regions associated with high secondary task rewards are much rarer, as indicated by the low value of ρ. In the mountain car problem, the visits are more frequent, and the differences between the different experience replay approaches are less significant. The value of the prioritization exponent used here is the same as that used in the navigation task. The approach of replaying sequences of transitions also offers noticeable performance improvements when applied to the primary task (as seen in **Figure 5**), especially during the early stages of learning, and when highly exploratory behavior policies are used. In both the navigation and mountain-car environments, the performances of the approaches that replay individual transitions—experience replay with uniform random sampling and prioritized experience replay—are found to be nearly equivalent. We have not observed a significant advantage of using the prioritized approach, as reported in previous studies (Schaul et al., 2016; Hessel et al., 2017) using deep RL. This perhaps indicates that improvements brought about by the prioritized approach are much more pronounced in deep RL applications.

The approach of replaying transition sequences seems to be particularly sensitive to the memory parameter m_t, with higher average returns being achieved for larger values of m_t. A possible explanation for this could simply be that larger values of m_t correspond to longer Θ_t sequences, which allow a larger number of replay updates to occur in more regions of the state/state-action space. The influence of the length of the Θ_b sequence, specified by the parameter m_b is also similar in nature, but its impact on the performance is less emphatic. This could be because longer Θ_b sequences allow a greater chance for their state trajectories to intersect with those of Θ_t, thus improving the chances of virtual transition sequences being discovered, and of the agent's value functions being updated using virtual experiences. However, the parameter n_v, associated with the size

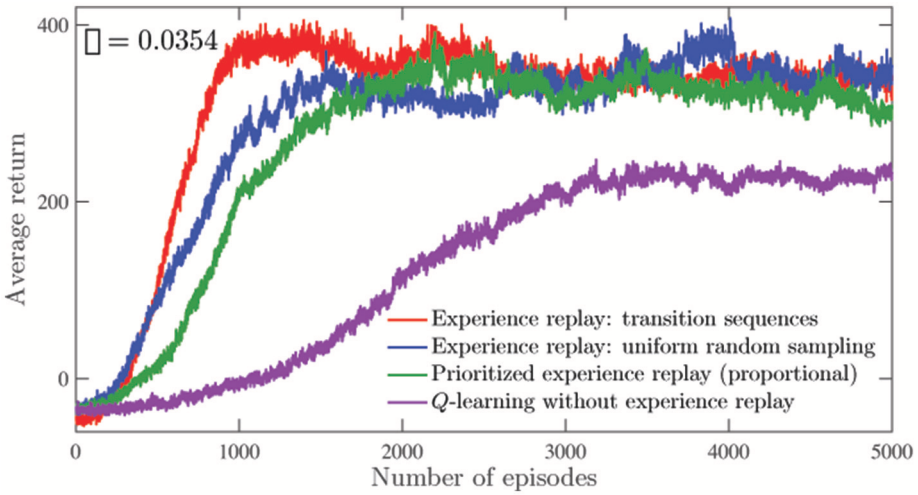

FIGURE 7 | Comparison of the average secondary returns over 50 runs using different experience replay approaches as well as *Q*-learning without experience replay in the mountain-car environment. The standard errors are all <85. For the different experience replay approaches, the number of replay updates are controlled to be the same.

TABLE 2 | Average secondary returns accumulated per episode (G_e) using different values of the memory parameters in the mountain car environment.

A	
m_b	G_e
10	221.0
100	225.1
1,000	229.9
B	
m_t	G_e
10	129.9
100	190.5
1,000	229.9
C	
n_v	G_e
10	225.6
50	229.9
100	228.4

With regular Q-learning (without experience replay), $G_e = 132.9$

of the library L_v does not seem to have a noticeable influence on the performance of this approach. This is probably due to the fact that the library L (and consequently L_v) is continuously updated with new, suitable transition sequences (successful sequences associated with higher magnitudes of TD errors) as and when they are observed. Hence, the storage of a number of transition sequences in the libraries becomes largely redundant.

Although the method of constructing virtual transition sequences is more naturally applicable to the tabular case, it could also possibly be extended to approaches with linear and non-linear function approximation. However, soft intersections between state trajectories would have to be considered instead of absolute intersections. That is, while comparing the state trajectories $S(x:y)$ and $S(x':y')$, the existence of s_c could be considered if it is close to elements in both $S(x:y)$ and $S(x':y')$ within some specified tolerance limit. Such modifications could allow the approach described here to be applied to deep RL. Transitions that belong to the sequences Θ_v and Θ_t could then be selectively replayed, thereby bringing about improvements in the sample efficiency. However, the experience replay approaches (implemented with the mentioned modifications) applied to the environments described in section 4 did not seem to bring about significant performance improvements when a neural network function approximator was used. The performance of the corresponding deep Q-network (DQN) was approximately the same even without any experience replay. This perhaps, reveals that the performance of the proposed approach needs to be evaluated on more complex problems such as the Atari domain (Mnih et al., 2015). Reliably implementing virtual transition sequences to the function approximation case could be a future area of research. One of the limitations of constructing virtual transition sequences is that in higher dimensional spaces, intersections in the state trajectories become less frequent, in general. However, other sequences in the library L can still be replayed. If appropriate sequences have not yet been discovered or constructed, and are thus not available for replay, other experience replay approaches that replay individual transitions can be used to accelerate learning in the meanwhile.

Perhaps another limitation of the approach described here is that constructing the library L requires some notion of a goal state associated with high rewards. By tracking the statistical properties such as the mean and variance of the rewards experienced by an agent in its environment in an online manner, the notion of what qualifies as a high reward could be automated using suitable thresholds (Karimpanal and Wilhelm, 2017). In

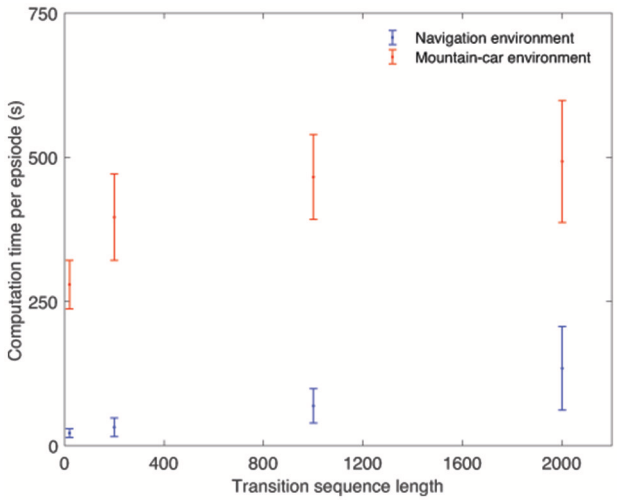

FIGURE 8 | The variation of computational time per episode with sequence length for the two environments, computed over 30 runs.

addition to this, other criteria such as the returns or average absolute TD errors of a sequence could also be used to maintain the library.

It is worth adding that the memory parameters m_b, m_t, and n_v have been set arbitrarily in the examples described here. Selecting appropriate values for these parameters as the agent interacts with its environment could be a topic for further research. **Figure 8** shows the mean and standard deviations of the computation time per episode for different sequence lengths, over 30 runs. The figure suggests that the computation time increases as longer transition sequences are used, and the trend can be approximated to be linear. These results could also be used to inform the choice of values for m_b and m_t for a given application. The values shown in **Figure 8** were obtained from running simulations on a computer with an Intel i7 processor running at 2.7 GHz, using 8 GB of RAM, running a Windows 7 operating system.

The approach of replaying transition sequences has direct applications in multi-task RL, where agents are required to learn multiple tasks in parallel. Certain tasks could be associated with the occurrence of relatively rare events when the agent operates under specific behavior policies. The replay of virtual transition sequences could further improve the learning in such tasks. Such as robotics, where exploration of the state/state-action space is typically expensive in terms of time and energy. By reusing the agent-environment interactions in the manner described here, reasonable estimates of the value functions corresponding to multiple tasks can be maintained, thereby improving the efficiency of exploration.

5. CONCLUSION

In this work, we described an approach to replay sequences of transitions to accelerate the learning of tasks in an off-policy setting. Suitable transition sequences are selected and stored in a replay library based on the magnitudes of the TD errors associated with them. Using these sequences, we showed that it is possible to construct virtual experiences in the form of virtual transition sequences, which could be replayed to improve an agent's learning, especially in environments where desirable events occur rarely. We demonstrated the benefits of this approach by applying it to versions of standard reinforcement learning tasks such as the puddle-world and mountain-car tasks, where the behavior policy was deliberately made drastically different from the target policy. In both tasks, a significant improvement in learning speed was observed compared to regular Q-learning as well as other forms of experience replay. Further, the influence of the different memory parameters used was described and evaluated empirically, and possible extensions to this work were briefly discussed. Characterized by controllable memory parameters and the potential to significantly improve the efficiency of exploration at the expense of some increase in computation, the approach of using replaying transition sequences could be especially useful in fields such as robotics, where these factors are of prime importance. The extension of this approach to the cases of linear and non-linear function approximation could find significant utility, and is currently being explored.

AUTHOR CONTRIBUTIONS

TK conceived the idea, coded the simulations, performed the experiments, authored the manuscript. RB reviewed and edited the manuscript, provided advice regarding the presentation of some of the ideas, prepared some of the figures and authored some parts of the manuscript.

ACKNOWLEDGMENTS

This work is supported by the President's graduate fellowship (MOE, Singapore) and TL@SUTD under the Systems Technology for Autonomous Reconnaissance & Surveillance (STARS-Autonomy & Control) program. The authors thank Richard S. Sutton from the University of Alberta for his feedback and many helpful discussions during the development of this work.

REFERENCES

Adam, S., Busoniu, L., and Babuska, R. (2012). Experience replay for real-time reinforcement learning control. *IEEE Trans. Syst. Man Cybern. Part C* 42, 201–212. doi: 10.1109/TSMCC.2011.2106494

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., et al. (2017). "Hindsight experience replay," in *Advances in Neural Information Processing Systems* (Long Beach, CA), 5048–5058.

Baird, L. C. (1999). *Reinforcement Learning Through Gradient Descent.* Robotics Institute, 227.

Bakker, B., Zhumatiy, V., Gruener, G., and Schmidhuber, J. (2006). "Quasi-online reinforcement learning for robots," in *ICRA 2006. Proceedings 2006 IEEE International Conference onRobotics and Automation, 2006* (Orlando, FL), 2997–3002.

Buhry, L., Azizi, A. H., and Cheng, S. (2011). Reactivation, replay, and preplay: how it might all fit together. *Neural Plast.* 2011:203462. doi: 10.1155/2011/203462

de Bruin, T., Kober, J., Tuyls, K., and Babuška, R. (2015). "The importance of experience replay database composition in deep reinforcement learning," in *Deep Reinforcement Learning Workshop, NIPS* (Montreal).

Even-Dar, E., and Mansour, Y. (2003). Learning rates for q-learning. *J. Mach. Learn. Res.* 5, 1–25. doi: 10.1007/3-540-44581-1_39

Fernández, F., and Veloso, M. (2005). *Building a Library of Policies Through Policy Reuse.* Technical Report CMU-CS-05-174, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.

Fonteneau, R., Murphy, S. A., Wehenkel, L., and Ernst, D. (2013). Batch mode reinforcement learning based on the synthesis of artificial trajectories. *Ann. Oper. Res.* 208, 383–416. doi: 10.1007/s10479-012-1248-5

Garcia, F., and Ndiaye, S. M. (1998). "A learning rate analysis of reinforcement learning algorithms in finite-horizon," in *Proceedings of the 15th International Conference on Machine Learning (ML-98* (Madison, WI: Citeseer).

Geist, M., and Scherrer, B. (2014). Off-policy learning with eligibility traces: a survey. *J. Mach. Learn. Res.* 15, 289–333. Available online at: www.jmlr.org

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., et al. (2017). Rainbow: combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*.

Isele, D., and Cosgun, A. (2018). Selective experience replay for lifelong learning. *arXiv preprint arXiv:1802.10269*.

Karimpanal, T. G., and Wilhelm, E. (2017). Identification and off-policy learning of multiple objectives using adaptive clustering. *Neurocomputing* 263, 39–47. doi: 10.1016/j.neucom.2017.04.074

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: a survey. *Int. J. Robot. Res.* 32, 1238–1274. doi: 10.1007/978-3-642-27645-3_18

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.* 8, 293–321.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., et al. (2016). "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning* (New York, NY).

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature* 518, 529–533. doi: 10.1038/nature14236

Modayil, J., White, A., and Sutton, R. S. (2014). Multi-timescale nexting in a reinforcement learning robot. *Adapt. Behav.* 22, 146–160. doi: 10.1177/1059712313511648

Moore, A. W., and Atkeson, C. G. (1993). Prioritized sweeping: reinforcement learning with less data and less time. *Mach. Learn.* 13, 103–130.

Narasimhan, K., Kulkarni, T. D., and Barzilay, R. (2015). "Language understanding for text-based games using deep reinforcement learning," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, eds L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, and Y. Marton (Lisbon:The Association for Computational Linguistics), 1–11.

Ólafsdóttir, H. F., Barry, C., Saleem, A. B., Hassabis, D., and Spiers, H. J. (2015). Hippocampal place cells construct reward related sequences through unexplored space. *Elife* 4:e06063. doi: 10.7554/eLife.06063

Rubinstein, R. Y., and Kroese, D. P. (2016). *Simulation and the Monte Carlo Method*. Hoboken, NJ: John Wiley & Sons.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). "Prioritized experience replay," in *International Conference on Learning Representations* (Puerto Rico).

Singer, A. C., and Frank, L. M. (2009). Rewarded outcomes enhance reactivation of experience in the hippocampus. *Neuron* 64, 910–921. doi: 10.1016/j.neuron.2009.11.016

Sutton, R. S. (1990). "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Proceedings of the Seventh International Conference on Machine Learning* (Austin, TX), 216–224.

Sutton, R. S., and Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., et al. (2011). "Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2* (Taipei: International Foundation for Autonomous Agents and Multiagent Systems), 761–768.

Thomas, P. S., and Brunskill, E. (2016). "Data-efficient off-policy policy evaluation for reinforcement learning," in *International Conference on Machine Learning* (New York, NY).

Thrun, S. (1996). "Is learning the n-th thing any easier than learning the first?," in *Advances in Neural Information Processing Systems* (Denver, CO), 640–646.

Thrun, S. B. (1992). *Efficient Exploration in Reinforcement Learning.* Technical report, Carnegie Mellon University, Pittsburgh, PA.

Tutsoy, O., and Brown, M. (2016a). An analysis of value function learning with piecewise linear control. *J. Exp. Theor. Artif. Intell.* 28, 529–545. doi: 10.1080/0952813X.2015.1020517

Tutsoy, O., and Brown, M. (2016b). Chaotic dynamics and convergence analysis of temporal difference algorithms with bang-bang control. *Optim. Control Appl. Methods* 37, 108–126. doi: 10.1002/oca.2156

van Seijen, H., and Sutton, R. S. (2013). "Planning by prioritized sweeping with small backups," in *Proceedings of the 30th International Conference on Machine Learning, Cycle 3, Vol. 28 of JMLR Proceedings* (Atlanta, GA), 361–369.

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., et al. (2016). Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*.

White, A., Modayil, J., and Sutton, R. S. (2012). "Scaling life-long off-policy learning," in *IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)* (San Diego, CA), 1–6.

White, A., Modayil, J., and Sutton, R. S. (2014). "Surprise and curiosity for big data robotics," in *AAAI-14 Workshop on Sequential Decision-Making with Big Data* (Quebec City, QC).

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.