PARSR: Priority-Adjusted Replay for Successor Representations

Samuel A. Barnett[*]
Department of Computer Science
Princeton University
Princeton, NJ 08540
samuelab@princeton.edu

Ida Momennejad
Microsoft Research
300 Lafayette Street
New York, NY 10012
idamo@microsoft.com

Abstract

Intelligent agents are capable of transfer and generalization. This flexibility in adapting to new tasks and environments often relies on representation learning and replay. Among these algorithms, successor representation learning and memory replay offer biologically plausible solutions. However, replay prioritization algorithms remain largely limited to reward prediction errors. Here we propose PARSR (pronounced *PARS*-er), Priority-Adjusted Replay for Successor Representations, to address this caveat. Decoupling learning of the environment dynamics and rewards, PARSR can use prediction errors from either representation learning or rewards to prioritize memory replay. We compare PARSR to prioritized sweeping, Dyna, and a number of state of the art algorithms using replay and successor representations in cognitive neuroscience.

Keywords: reinforcement learning, prediction error, replay, planning, decision-making, prioritization, human-like behavior, Dyna

Acknowledgements

We are grateful for the support and feedback provided by Arthur Juliani, Brandon Davis, Katja Hofmann, Sam Devlin, Harm van Seijen, Evan Russek, Marcelo Mattar, and Nathaniel Daw.

[*]Research performed as an intern at Microsoft Research.

1 Introduction

Intelligent agents are capable of transfer and generalization. Imagine driving to a coffee shop to meet a friend. If we encounter a blocked road, we are able to quickly adapt in choosing a new route that will get us there. Or if we decide that a different coffee shop might be preferable, we can just as easily change course.

To accommodate this flexibility to changes in the environment, contemporary reinforcement learning algorithms often rely on representation learning and replay [1, 2, 3, 4]. **Learning flexible yet compact representations of the environment allows us to adapt to changes in its rewards, and replay enables us to adapt to changes in transition structures without the need for significant amounts of further real experience.**

One such algorithm proposes a biologically plausible solution [5], combining successor representation (SR) [6, 7] for representation learning supplemented by memory replay (inspired by Dyna architectures [8]). This algorithm, Dyna-SR for short, captures human behavioral accuracy and reaction times across a number of tabular tasks. A key advantage of the Dyna-SR algorithm is that it is almost as flexible a model-based RL algorithm, while at decision time it is almost as inexpensive as model-free RL. By caching multi-step trajectories of states offline, Dyna-SR remains more flexible than MFRL and SR alone, while avoiding MBRL's high cost of rolling out entire state-action-state-reward trajectories at decision time.

As a caveat, the replay prioritization in current implementations of Dyna-SR focus on more recent memories for efficiency. While this heuristic may capture certain aspects of human memory recall [9], previous work in cognitive neuroscience suggest human-like replay may be better modelled by an error-based replay prioritization [10]. **Thus, here we extend the scope of algorithms combining representation and replay, using both reward-based and representation-based errors for replay prioritization.** While current approaches remain largely limited to tagging memories with reward prediction errors (PE) for priority [11, 12, 13, 14, 15], our proposed algorithm is inspired by Dyna-SR but can prioritize replay using either reward PE and successor PE.

We propose PARSR (pronounced *PARS*-er), Priority-Adjusted Replay for Successor Representations, which improves on Dyna-SR offering more human-like replay prioritization with no effective increase in hyperparameters. As an SR-based algorithm, PARSR learns a representation of the transition structure (i.e., the environment dynamics) and the reward structure separately, allowing either to be quickly relearned for greater generalization. Critically, by decoupling reward and transition representations, PARSR can use the prediction errors from either to prioritize memory replay.

We propose two variants of PARSR based on the choice of prediction error: M-PARSR (prioritizes memories using successor PE) and Q-PARSR (prioritizes memories using value PE). This prioritization for memory selection distinguishes PARSR from Dyna-SR [5], which performs replay-enhanced SR learning with random memory selection with a recency bias.

We test how well PARSR captures human behavior in small tabular experiments (with 6 states) [1] as well as a scaled version of the experiments (with 121 states). We compare PARSR to a number of state of the art algorithms using replay on simple benchmark transfer learning (or *revaluation*) tasks in cognitive neuroscience (Figs. 1a, 1b, 1c) and a scaled up version of these tasks (Figs.1d, 1e, 1f). We find that PARSR matches human-like behavior as well as other algorithms' efficiency in learning speed of the prioritization-based algorithms. To clarify differences among the solutions different replay heuristics provide, we visualize which experiences are prioritized as more important to recall by different prioritization algorithms (Fig. 1e). The code for implementing the algorithm and benchmark experiments is available at `https://github.com/s-a-barnett/PrioritizedSR`.

1.1 Related work

Experience replay has been incorporated as a core part of many reinforcement learning algorithms, both in the tabular setting [8, 16], and in deep reinforcement learning [17]. The use of prioritization based on prediction error as a heuristic for faster learning has also been explored in both settings [11, 12, 13, 14, 15].

The successor representation was first introduced in [6], forming the basis for a number of algorithms in both tabular and function approximation settings [1, 5, 18, 4]. It has also been studied for its neuroscientific plausibility in [7]. For comprehensive overviews of the successor representation and its properties, refer to [19] and [2].

2 Algorithms

We focus on RL algorithms that integrate the advantages of both model-based planning (flexibility) and model-free learning methods (speed). For a full treatment on the RL setting, we refer to the classic text by Sutton and Barto [20], in particular Chapter 8. We use as baselines the Dyna-Q algorithm [8, 20] and prioritized sweeping (PS) algorithm [12, 11, 20], which interleave learning from real experience with learning from simulated experience, or *replay*, sampled from a model learned from previous interactions with the environment. In the deterministic setting considered in this paper, this model is a dictionary whose entries are state-action pairs, and whose values are the next state and received reward.

While Dyna-Q and prioritized sweeping both exploit replay to integrate new experiences into their Q function more efficiently, they differ in their selection mechanisms: Dyna-Q samples past experiences uniformly from its model, whereas PS selects experiences based on a queue ordered according to the magnitude of the temporal difference (TD) error on the Q function that was recorded when the state was most recently encountered. This is then propagated through to states that precede the replayed state, allowing the largest changes to flow backwards through the model. This approach is consistent with human studies suggesting that larger prediction errors are followed by more offline replay, and offline replay of predecessors of states tagged with prediction error is correlated with future revaluation behavior [10].

Algorithm 1 PARSR

```
1: Hyperparameters: Number of replay cycles $n$, exploration parameter $\varepsilon$, SR learning rate $\alpha_M$, reward weights learning rate $\alpha_{\boldsymbol{w}}$, prioritization type *PriType*.
2: Initialize $M(a,s,s')$, $\boldsymbol{w}(s)$, *Model*$(s,a)$ for all $s,a$ and *PQueue* to empty.
3: while True do
4:     $s \leftarrow$ current (nonterminal) state.
5:     $Q \leftarrow \boldsymbol{w}(:)^\top M(a,s,:)$.
6:     $a \leftarrow \varepsilon$-*greedy*$(s,Q)$.
7:     Take action $a$; observe reward, $r$, and state, $s'$.
8:     *Model*$(s,a) \leftarrow r, s'$.
9:     $Q \leftarrow \boldsymbol{w}(:)^\top M(a,s,:)$.
10:    $a' \leftarrow \arg\max_{a''}(Q(s',a''))$.
11:    $\boldsymbol{\delta}_M \leftarrow \big[\mathbf{1}_s + \gamma M(a',s',:) - M(a,s,:)\big]$.    ▷ SR ($M$) prediction error.
12:    $M(a,s,:) \leftarrow M(a,s,:) + \alpha_M \boldsymbol{\delta}_M$.
13:    $\boldsymbol{w}(s) \leftarrow \boldsymbol{w}(s) + \alpha_{\boldsymbol{w}}\big[r - \boldsymbol{w}(s)\big]$.
14:    if *PriType* is M-PARSR then
15:        $p \leftarrow \|\boldsymbol{\delta}_M\|$
16:    else if *PriType* is Q-PARSR then
17:        $p \leftarrow \delta_Q \equiv \boldsymbol{\delta}_M^\top \boldsymbol{w} - \boldsymbol{w}(s) + r$    ▷ $Q$ prediction error.
18:    Insert $s,a$ into *PQueue* with priority $p$.
19:    loop $n$ times
20:        if *PQueue* is not empty then
21:            $\overline{s}, \overline{a} \leftarrow$ *first*(*PQueue*).
22:        else
23:            $\overline{s} \leftarrow$ random previously observed state.
24:            $\overline{a} \leftarrow$ random action previously taken in $\overline{s}$.
25:        $\overline{r}, \overline{s}' \leftarrow$ *Model*$(\overline{s},\overline{a})$.
26:        $Q \leftarrow \boldsymbol{w}(:)^\top M(\overline{a},\overline{s},:)$.
27:        $\overline{a}' \leftarrow \arg\max_{\overline{a}''}(Q(\overline{s}',\overline{a}''))$.
28:        $\boldsymbol{\delta}_M \leftarrow \big[\mathbf{1}_s + \gamma M(\overline{a}',\overline{s}',:) - M(\overline{a},\overline{s},:)\big]$.
29:        $M(\overline{a},\overline{s},:) \leftarrow M(\overline{a},\overline{s},:) + \alpha_M \boldsymbol{\delta}_M$.
```

Though replay alone improves an agent's ability to relearn local aspects of the task at hand, the representation of this task (namely, the Q function) is inflexible inasmuch as it can obscure the different kinds of changes that might take place. This can lead to slower learning, and does not correspond to the representations of such tasks used by biological agents [21, 19, 2]. To provide further flexibility, we define the successor representation (SR) as the discounted sum over expected future state visitations:

$$M(a,s,s') = \mathbb{E}\left[\sum_{t=T}^{\infty} \gamma^{t-T} \mathbf{1}(s_t = s') \mid s_T = s, a_T = a\right]. \quad (1)$$

The Q function can now be expressed as a linear combination of the SR and the rewards of each state, $\boldsymbol{w}(s')$:[1]

$$Q(s,a) = \sum_{s'} M(a,s,s')\boldsymbol{w}(s'). \quad (2)$$

The SR is learned with its own TD update (algorithm 1, line 11), and the weights are learned using a direct update (algorithm 1, line 13).

In SR-based RL algorithms each experience simultaneously updates both the SR and reward weights, enabling flexibility to changes in rewards (reward transfer). However, transition transfer requires updating SR for states with reevaluated transitions, via real or replayed experience. Thus, combining the representational flexibility of the SR with efficient forms of planning through replay achieves both reward and transition transfer.

The Dyna-SR algorithm [5] is such a hybrid, sampling experiences randomly with a recency-weighted bias in order to update SR. However, we hypothesized that sampling past experiences according to a prioritization schema, similar to prioritized sweeping, could further improve performance while still capturing human-like behavior.

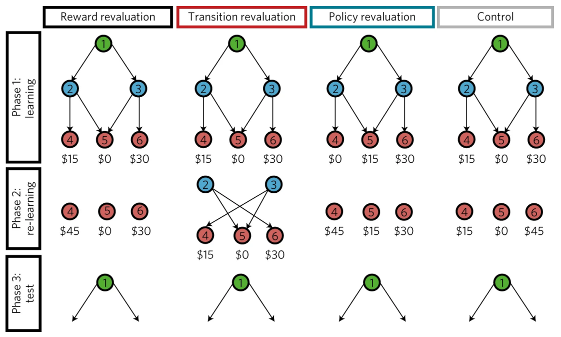
We call this novel algorithm *PARSR*: Priority-Adjusted Replay for Successor Representations (algorithm 1 and Fig. 2, changes from Dyna-SR in blue).

Unlike prioritized sweeping, which only relies on reward prediction errors (PE), PARSR can prioritize replay using PE for either the SR or reward weights. When using the latter priority measure for PARSR we call this variant *Q-PARSR*, and when using successor prediction errors ($\|\boldsymbol{\delta}_M\|$), we refer to the variant as *M-PARSR*. Each algorithm represents different choices about what kinds of experience to prioritize, potentially leading to different behavior and training times.

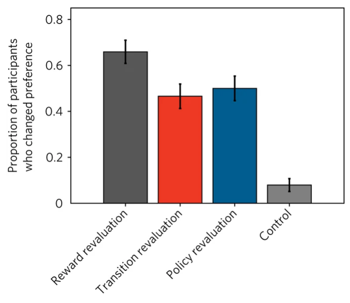
3 Experiments

We evaluate the performance of both variants of the PARSR algorithm in comparison to other SR-based and replay-based RL algorithms on revaluation tasks on different scales. Each experiment is evaluated over 10 seeds with $\varepsilon \in \{0.1, 0.3, 0.5, 1.0\}$. The Six States (resp. Six Rooms) experiment is performed for 100 (resp. 10) runs per seed, with 10 (resp. 1000) replay cycles per timestep for each algorithm. Note that PARSR has the same number of hyperparameters as other algorithms, so any improvement in human-like performance is not due to increased model complexity.

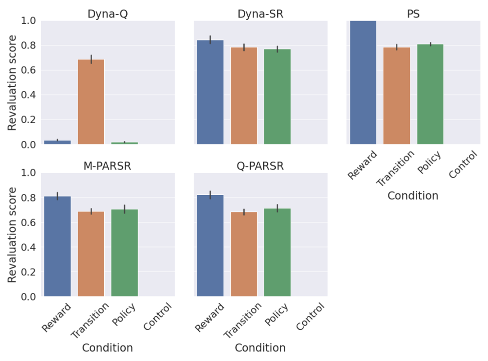
[1]We can extend this analysis to rewards defined on state-action pairs, $\boldsymbol{w}(s',a')$, and define our SR on 4-tuples as $M(a,s,a',s')$.



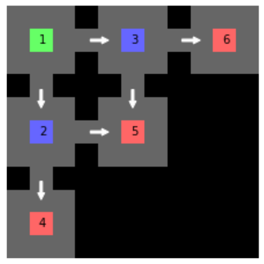
(a) Six States design.



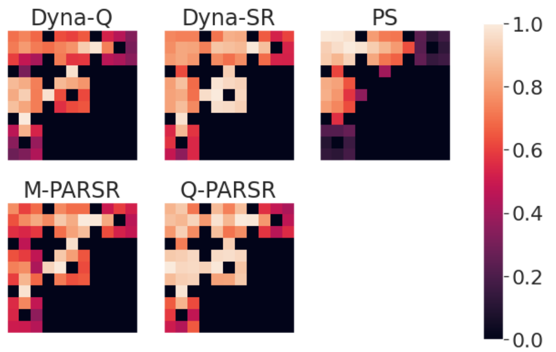
(b) Six States human results.



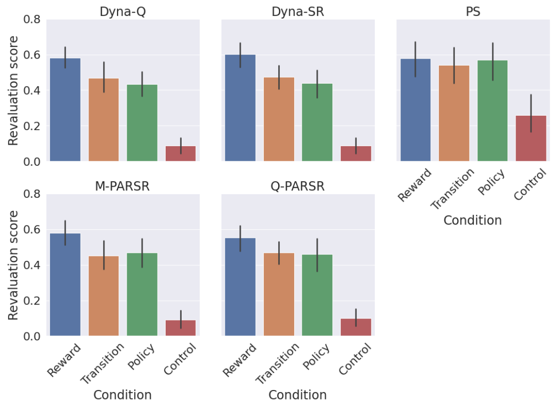
(c) Six States model results (10 replay cycles).



(d) Six Rooms (121 states) design.



(e) Six Rooms priority visualization, phase 2 of transition revaluation.



(f) Six Rooms model results (1000 replay cycles).

Figure 1: **Top row**: structure and results for the Six States experiment, reproduced from Experiment 2 in [1]. In (a), numbered circles denote different states, and arrows denote the unidirectional actions available at each state. For a given phase of a given condition, trials begin only in the earliest-stage states that are displayed in the figure for that condition and phase. (b) shows the proportion of participants in the human experiment ($n = 88$) who changed preference following the re-learning phase for each condition. Participants show greater ability to transfer in the case of reward revaluation than they do for the transition or policy conditions, though they are also capable of performing those tasks in some proportion. (c) reproduces these results for different algorithms. **Bottom row**: results for the Six Rooms experiment. (d) shows the map of the Six Rooms environment, with white arrows denoting "trapdoors" between rooms. (f) shows the revaluation scores for each model: all algorithms with the exception of PS attain results that are analogous to those achieved by human participants in Experiment 2 of [1] (Six States). (e) shows the relative frequency of the states prioritized during replay for the transition revaluation condition during phase 2. *M-PARSR has a much narrower focus on the bottlenecks between the rooms at which the revaluation is taking place, whereas Q-PARSR has a more uniform distribution over the prioritized states despite nonetheless employing a priority queue.* Both achieve similar performance in these tasks in spite of these differences.

3.1 Six States

We first reproduce and extend the results of Experiment 2 of [1], which we refer to as the "Six States" experiment. In this decision-making task (in which, unlike Experiment 1, the participant takes actions at every step), participants complete four games, each corresponding to a different experimental condition (Fig. 1a). The six states of the environment are structured as a unidirectional, three-stage decision tree, where two actions are available at the first two stages and a scalar reward is received at the terminal states in the final stage.

Each task is divided into three phases: one must pass each phase three times consecutively in order to progress to the next phase. In phase 1, participants are trained on a specific reward and transition structure. In phase 2, a change in either the reward or transition structure is changed, and participants learn about the changed structure *without revisiting the starting state*. Hence, participants do not get to experience these new contingencies following an action taken from the first stage. In phase 3, participants perform a single test trial beginning from the starting state, with the revaluation score corresponding to the probability (over multiple experimental runs) that the participant changes their action in state 1 between the end of phase 1 and the single trial in phase 3. Results from a human study in [1] show a greater revaluation score for the reward condition than transition or policy conditions, and no significant difference in revaluation between the latter two. Revaluation in the control condition is significantly lower than all of these.

For both Experiments 1 and 2, we find that both variants of PARSR are able to capture the human revaluation behavior in the task equally as well as Dyna-SR.

3.2 Six Rooms

We wanted to investigate whether the findings from the Six States experiment scaled to tasks with larger state spaces. To test this, we designed Six Rooms, a gridworld analog with 121 states. In this environment, the states in the Six States experiment correspond to the centers of the six rooms, laid out in a similar structure to the smaller environment. Unidirectionality is enforced through one-way corridors between each room, in order to retain the solution structure of the smaller environment. Each of the conditions and phases of the Six States experiment can be defined analogously for the Six Rooms domain.

Despite their similar latent structure, the Six Rooms environment represents a greater challenge for the agents since moving between the rooms requires a sequence of *several* actions, thus requiring more updates to the agents' representations. To successfully pass each phase, therefore, requires not only that the agent navigates to the correct state given its starting state, but moreover that it do so using the shortest path. During phase 2, agents are initialized in the center states of the second-stage rooms and are required to navigate to center states of the correct final rooms using the quickest path. This matches the difficulty of the exclusion criteria across the four conditions.

Fig. 1f shows the results of the experiment for each algorithm. We observe that the revaluation scores for these tasks match that of the human performance on the Six States experiment for all but Priortized Sweeping and most faithfully by PARSR. This suggests that PARSR's performance scales to more complex tasks with a similar latent structure. This further offers the testable prediction that human behavior in the Six Rooms experiment should scale accordingly.

In Fig. 1e, we visualize the relative frequencies at which each state was prioritized by each algorithm during transition revaluation (phase 2). **We observe a difference between the two PARSR variants in their prioritization strategies: while M-PARSR prioritizes replaying bottleneck states at which the revaluation is occurring, Q-PARSR's prioritization focus is more diffuse.** Future work is required to test which conditions and tasks are best served by each prioritization scheme. For instance, adding meta-learning to PARSR could control which type of prioritization is appropriate depending on the task at hand.

4 Discussion and Future Work

We have introduced PARSR: an algorithm that combines successor representation based learning with novel and neurally plausible replay prioritization heuristics. PARSR's two variants prioritize experience replay using either representation-based or reward-based prediction errors. Both PARSR variants show human-like behavior on benchmark tasks with 6 states (Figs. 1a, 1b 1c) as well as as scaled tasks (the Six Rooms environment) with 121 states (Figs.1d, 1e, 1f). The latter offers novel predictions for human behavior in scaled experiments.

In future work we will extend PARSR beyond tabular environments, as a deep RL algorithm with function approximation, similar to that of Prioritized Experience Replay [13]. In addition to extending PARSR to deep learning and more complex environments, future work would investigate further the nature of the two prioritization signals in PARSR variants, e.g. to investigate sequence memory activations at given moments in the task [18]. Moreover, we have visualized how error signals determine the relative frequency of prioritized experiences (Fig. 1e). Future work is required to investigate which problems are better served by which prioritization schemes. One solution is a novel algorithm combining PARSR with meta-learning of a control parameter learned across tasks and environments that, given the problem at hand, determines which PE signal is appropriate for efficient replay prioritization.

References

[1] Ida Momennejad et al. "The successor representation in human reinforcement learning". In: *Nature Human Behaviour* 1.9 (2017), pp. 680–692.

[2] Ida Momennejad. "Learning structures: Predictive representations, replay, and generalization". In: *Current Opinion in Behavioral Sciences* 32 (2020), pp. 155–166.

[3] Lennart Wittkuhn et al. "Replay in minds and machines". In: *Neuroscience & Biobehavioral Review* 129 (2021), pp. 367–388.

[4] Andre Barreto et al. "Transfer in deep reinforcement learning using successor features and generalised policy improvement". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 501–510.

[5] Evan M Russek et al. "Predictive representations can link model-based reinforcement learning to model-free mechanisms". In: *PLoS Computational Biology* 13.9 (2017), e1005768.

[6] Peter Dayan. "Improving generalization for temporal difference learning: The successor representation". In: *Neural Computation* 5.4 (1993), pp. 613–624.

[7] Kimberly L Stachenfeld, Matthew M Botvinick, and Samuel J Gershman. "The hippocampus as a predictive map". In: *Nature Neuroscience* 20.11 (2017), pp. 1643–1653.

[8] Richard S Sutton. "Dyna, an integrated architecture for learning, planning, and reacting". In: *ACM Sigart Bulletin* 2.4 (1991), pp. 160–163.

[9] M.J Kahana, M.W Howard, and S.M Polyn. *2.26 - Associative Retrieval Processes in Episodic Memory*. Elsevier, 2008.

[10] Ida Momennejad et al. "Offline replay supports planning in human reinforcement learning". In: *Elife* 7 (2018), e32548.

[11] Andrew W Moore and Christopher G Atkeson. "Prioritized sweeping: Reinforcement learning with less data and less time". In: *Machine Learning* 13.1 (1993), pp. 103–130.

[12] Jing Peng and Ronald G Williams. "Efficient learning and planning within the Dyna framework". In: *Adaptive Behavior* 1.4 (1993), pp. 437–454.

[13] Tom Schaul et al. "Prioritized experience replay". In: *arXiv preprint arXiv:1511.05952* (2015).

[14] Matteo Hessel et al. "Rainbow: Combining improvements in deep reinforcement learning". In: *Thirty-second AAAI conference on artificial intelligence*. 2018.

[15] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. *Prioritized Level Replay*. 2021. arXiv: `2010.03934 [cs.LG]`.

[16] Long-Ji Lin. "Self-improving reactive agents based on reinforcement learning, planning and teaching". In: *Machine Learning* 8.3 (1992), pp. 293–321.

[17] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

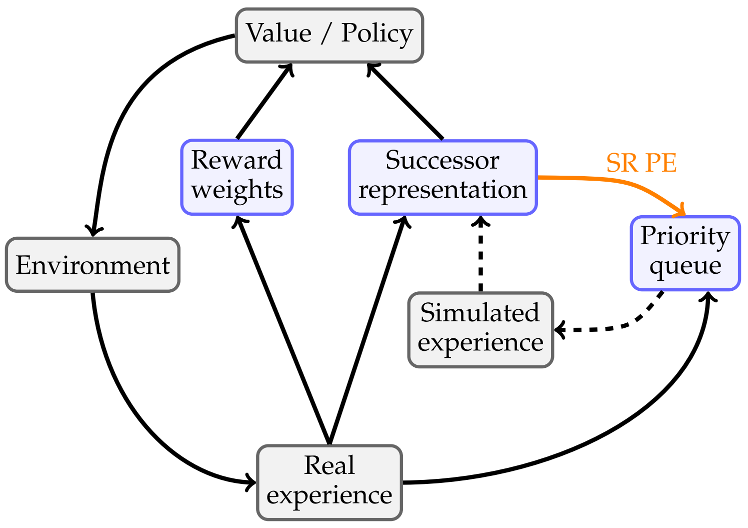
[18] Marcelo G Mattar and Nathaniel D Daw. "Prioritized memory access explains planning and hippocampal replay". In: *Nature Neuroscience* 21.11 (2018), pp. 1609–1617.

[19] Samuel J Gershman. "The successor representation: its computational logic and neural substrates". In: *Journal of Neuroscience* 38.33 (2018), pp. 7193–7200.

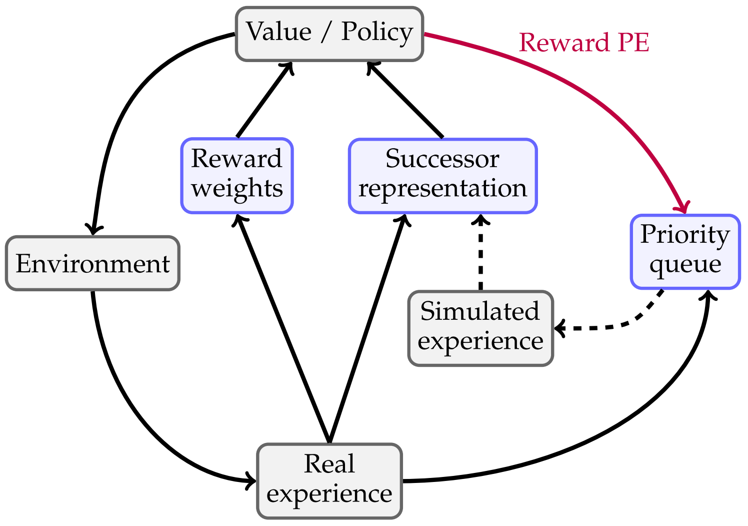
[20] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[21] Edward C Tolman. "Cognitive maps in rats and men". In: *Psychological Review* 55.4 (1948), pp. 189–208.

5 Appendix



(a) M-PARSR



(b) Q-PARSR

Figure 2: Flowcharts for both variants of the PARSR algorithm. Components of the agent are in blue. **Real experience** (s, a, r, s') is fed into the priority queue, and used to update the **reward weights** $\boldsymbol{w}(s')$ and **successor representation** $M(a, s, s')$ via temporal difference learning. The **priority queue** returns **simulated experiences** $(\overline{s}, \overline{a}, \overline{r}, \overline{s}')$, which are used to provide further updates to the successor representation. M-PARSR (a) prioritizes according to the successor representation prediction error (SR PE), whereas Q-PARSR (b) prioritizes according to the reward prediction error (Reward PE). The reward weights and successor representation determine the state-action **value** function $Q(s, a) = \sum_{s'} M(a, s, s')\boldsymbol{w}(s')$, which in turn determines the **policy** for acting in the **environment**.