Towards TempoRL: Learning When to Act

André Biedenkapp [1] **Raghu Rajan** [1] **Frank Hutter** [1 2] **Marius Lindauer** [3]

Abstract

Reinforcement Learning is a powerful approach to learning behaviour through interactions with an environment. However, behaviours are learned in a purely reactive fashion, where an appropriate action is selected based on an observation. In this form, it is challenging to learn *when* it is necessary to make new decisions. This makes learning inefficient especially in environments with with very fine-grained time steps. Instead we propose a more proactive setting in which not only an action is chosen in a state but also for how long to commit to that action. We demonstrate the effectiveness of our proposed approach on a set of small grid worlds, showing that our approach is capable of learning successful policies much faster than vanilla $\mathcal{Q}$-learning.

1. Introduction

In reinforcement learning (RL), the goal is to learn policies that optimize a reward signal through interactions with an environment. In recent years RL has been shown to be a powerful approach for learning successful policies in various domains, such as game playing (Mnih et al., 2015), continuous control (Lillicrap et al., 2016) and multi agent systems (Baker et al., 2020).

In the classical view on RL, policies are learned in a mostly reactive fashion, i.e., observe a state and react to that state with an action. Guided by the reward signal, policies that are learned in such a way can decide which action is expected to yield a desired outcome. However, these policies generally do not learn *when a new decision has to be made*. This potentially complicates the learning process as many state-action-reward-sequences need to be observed in which the same action has to be chosen always. This reactive way of learning is particularly problematic in environments with fine-grained discrete or continuous time.

[1]Department of Computer Science, University of Freiburg, Germany [2]BCAI, Renningen, Germany [3]Information Processing Institute (tnt), Leibniz University Hannover, Germany. Correspondence to: André Biedenkapp <biedenka@cs.uni-freiburg.de>.

Big@ICML Workshop, 2020.

Temporal abstractions are a common way to simplify learning of policies with potentially many required actions. Typically, the temporal abstraction is learned on the highest level of a hierarchy and the required behaviour on a lower level (Schaul et al., 2015). E.g., on the highest level a goal policy learns which states are necessarily visited and on the lower level the behaviour to reach goals is learned. Spacing goals very far apart still requires to learn very complex behaviour policies whereas a narrow goal spacing requires to learn complex goal policies. Another form of temporal abstraction is to use actions that work at different timescales (Precup et al., 1998). E.g., an agent is tasked with moving an object from one place to another. On the highest level the agent would follow a policy with abstract actions, such as *pick-up object*, *move object*, *put-down object*, whereas on the lower level actions could directly control some actuators to perform the abstract actions. Temporal abstractions enable interaction with complex environments by abstracting away complexities.

Such hierarchical approaches are still reactive, but instead of reacting to an observation on only one level, reactions are learned on multiple levels. Further, though these approaches might allow us to learn *which* states are necessarily traversed in the environment, they do not enable us to learn *when* a new decision has to be made on the behaviour level.

In this work, we propose an alternative: a more proactive view on learning policies that allows us to learn how long an action should be played. We do this by reexamining the relationship between agent and environment and the dependency on time. This allows us to introduce *skip connections* into the environment. We can show that these skip connections do not change the optimal policy or state-action-values but allow us to propagate information much faster. We examine our method TEMPORL on a variety of finite MDPs. Specifically, our contributions are as follows:

1. We propose a proactive alternative to classical RL.
2. We introduce skip-connections for MDPs by playing an action for several consecutive states, which leads to faster propagation of information about future rewards.
3. We propose a mechanism based on a flat hierarchy for learning when to make decisions through the use of skip-connections.

2. Related Work

A common framework for temporal abstraction in RL is the options framework (see e.g. Precup et al., 1998; Sutton et al., 1999; Stolle & Precup, 2002; Bacon et al., 2017; Harb et al., 2018; Harutyunyan et al., 2018; Mankowitz et al., 2018; Khetarpal & Precup, 2019). Options are triples $\langle \mathcal{I}, \pi, \beta \rangle$ where $\mathcal{I}$ is the set of admissible states that defines in which states the option can be played; π is the policy the option follows when it is played; and β is a random variable that determines when an option is terminated. In contrast to our proposed method, options require a lot of prior knowledge about the environment to determine the set of admissible states as well as the option policies themselves.

An important element in DQN's success in tackling various Atari games (Mnih et al., 2015) is due to the use of *frame skipping* (Bellemare et al., 2013). Thereby the agent does not have to act at every possible state but skips over a few states, always playing the same action, before making a new decision. Without the use of frame skipping, the change between successive observations is virtually indistinguishable and would have required more observations to learn the same policy. Tuning the *skip-size* can additionally improve performance (Braylan et al., 2015; Khan et al., 2019). However, a static skip-size might not be ideal. E.g., in Pong there is little to no need to react to the ball moving away.

Different techniques have been proposed to handle continuous time environments (Baird, 1994; Doya, 2000; Tiganj et al., 2017). A well known technique is advantage learning (Baird, 1994) which allows identifying which actions are more promising than others. Recently, Huang et al. (2019) proposed to use *Markov Jump Processes* (MJPs) that are closely connected to the idea of skip-MDPs which we present here. MJPs are designed to study optimal control in MDPs where observations have an associated cost. The goal there is to balance the costs of observations and the environment to act in an optimal manner with respect to total cost. Their analysis demonstrated that frequent observations are necessary in regions where an optimal action might change rapidly, while in areas of infrequent change, fewer observations are sufficient. In contrast to ours, this formalism prohibits observations of intermediate transitions.

Lakshminarayanan et al. (2017) proposed a network with multiple output heads per action to handle different repetition lengths, drastically increasing the action space. In contrast to that, Sharma et al. (2017) proposed a framework, *FiGAR*, that jointly learns an action policy and a second policy that decides how often to repeat an action. FiGAR proved to be useful with different algorithms for different tasks. However, its repetition policy is not conditioned on the chosen action. The polices are learned together through a joint loss. Thus, the repetition policy only learns which repetition length works well on average for all actions.

3. TempoRL

We begin this section by introducing skip connections into MDPs to propagate information about expected future rewards faster. We then introduce a novel learning mechanism that makes use of a flat hierarchy to learn a policy that is capable of not only learning which action to take, but also for how *long* this action should be taken.

3.1. Temporal Abstraction through Skip MDPs

We contextualize an existing MDP $\mathcal{M}$ to allow for skip connections as $\mathcal{M}_{\mathcal{J}} := \{\mathcal{M}_j\}_{j \in \mathcal{J}}$ with $\mathcal{M}_j := \langle \mathcal{S}, \mathcal{A}, \mathcal{P}_j, \mathcal{R}_j \rangle$. The skip-connections $j \in \mathcal{J}$ act as context to the MDP and induce different MDPs with shared state and action spaces ($\mathcal{S}$, $\mathcal{A}$), but different transitions $\mathcal{P}_j$ and reward functions $\mathcal{R}_j$. All parts of the skip MDPs are inherited from the original MDP, but the transition function is adapted to allow for skip transitions. The introduction of the skip transitions also requires the reward function to handle the newly introduced transitions. Therefore, the skip MDPs are all equal except in those states where skips introduce new transitions from a state s to states that were not reachable from s before.

Similarly to options, a skip is a triple $\langle s, a, j \rangle$, where s is the starting state for a skip transition (and not a set of states as in the options framework); a is the action that is executed when skipping through the MDP; and j is the skip-length. A skip connects two states s and s' iff state s' is reachable from state s by repeating action a, $(j+1)$-times. This gives us the following skip transition function:

$$\mathcal{P}_j(s, a, s') = \begin{cases} \prod_{k=0}^{j} \mathcal{P}^a_{s_k s_{k+1}} & \text{if reachable} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

with s_k and s_{k+1} the states traversed by playing action a for the kth time, and with $s_0 = s$ and $s_{j+1} = s'$. This change in the transition function is reflected accordingly in the reward function:

$$\mathcal{R}_j(s, a, s') = \begin{cases} \sum_{k=0}^{j} \gamma^k \mathcal{R}^a_{s_k s_{k+1}} & \text{if reachable} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Note that for a skip of length 0 we recover the original transition function $\mathcal{P}_0(s, a, s') = \mathcal{P}^a_{ss'}$ as well as the original reward function $\mathcal{R}_0(s, a, s') = \mathcal{R}^a_{ss'}$. Through this formulation of skip MDPs, information about future rewards can be propagated much more quickly and enables us to determine *when* it becomes beneficial to switch actions. The goal with skip-MDPs is to find an optimal skip policy, i.e., a policy with the fewest decision points to reach the optimal reward.

3.2. Learning When to Make Decisions

In order to learn using skip connections we need a new mechanism that selects which skip connection to use. In

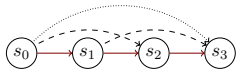
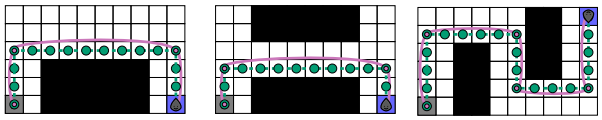


Figure 1. Example transitions with skip of length two ($\cdots$). At the same time we can also observe shorter skips of length one (- - -).



(a) Cliff (b) Bridge (c) ZigZag

Figure 2. 6×10 Grid Worlds. Agents have to reach a fixed goal state from a fixed start state. Large/small dots represent decision steps of vanilla and TEMPORL $\mathcal{Q}$-learning policies.

order to facilitate this, we propose using a flat hierarchy in which a *behaviour* policy determines the action a to be played given the current state s, and a *skip* policy determines how long to play this action for.

To learn the behaviour, we can make use of classical $\mathcal{Q}$-learning, where the $\mathcal{Q}$-function $\mathcal{Q}^\pi(s_t, a) := \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s = s_t, a\right]$ gives a mapping of expected future rewards when playing action a in state s_t at time t and continuing to follow the behaviour policy π thereafter. To learn to skip, we first have to define a *skip-action space* that determines all possible lengths of skip-connections, e.g., $a_j \in \langle 0, 1, \ldots, J \rangle$. To learn the value of a skip we can simply make use of n-step $\mathcal{Q}$-learning with the condition that, at each step, the action stays the same: $\mathcal{Q}^{\pi_j}(s_t, j|a) := \mathbb{E}\left[\sum_{k=0}^{j-1} \gamma^k r_{t+k} + \gamma^j \mathcal{Q}^\pi(s_{t+j}, a) | s = s_t, a, j\right]$. We call this a flat hierarchy since the behaviour and the skip policy have to always make decisions at the same time.

One observation we can make about this learning scheme is that, when playing skip action a_j, we are able to observe all smaller skip transitions for all intermediate steps. Figure 1 gives a visual representation. This allows us to learn about all smaller skips when playing a larger skip action, making training potentially much more efficient.

4. Experiments

In this section, we describe experiments for a tabular $\mathcal{Q}$-learning implementation[1] that we evaluated on various grid-worlds with sparse rewards (see Figure 2). We first evaluate our approach on the *cliff* environment (see Figure 2a) before evaluating the influence of the exploration schedule on both vanilla $\mathcal{Q}$-learning as well as TEMPORL.

[1] https://github.com/automl/TabularTempoRL

Gridworlds All considered environments (see Figure 2) are discrete, deterministic, have sparse rewards and have size 6×10. Falling off a cliff results in a negative reward (-1) and reaching a goal state results in a positive reward ($+1$). Both cliff and goal states terminate an episode. All other states result in no reward. An agent can only execute the actions up, $down$, $left$, $right$ with diagonal moves not possible. If the agent does not reach a goal/cliff in 100 steps, an episode terminates without a reward.

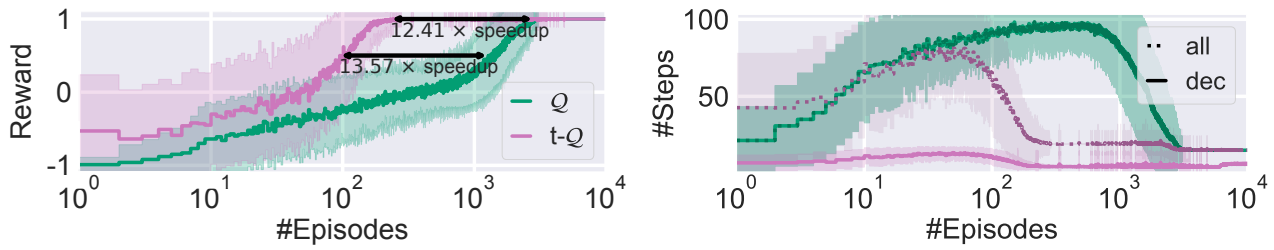
For the cliff environment, a shortest path through the environment requires 16 steps. However, to reach the goal, decisions about unique actions are only required at 3 time points. The first is in the starting state and determines that action up should be repeated 3-times, the next is repeating action $right$ 10-times thereafter and the final one is repeating action $down$ 3-times. Thereby, an optimal proactive policy that is capable of joint decision of action and skip length requires $\sim$ 80% fewer decisions than an optimal reactive policy that has to make decisions in each state.

For this experiment, we limit our TEMPORL agent to a maximum skip length of 7; thus, a learned optimal policy requires 4 decision points instead of 3.[2] We compare the learning speed, in terms of training policies, of our approach to a vanilla $\mathcal{Q}$-learning agent. Both methods are trained for 10 000 episodes using the same ϵ-greedy strategy, where ϵ is linearly decayed from 1.0 to 0.0 over all episodes.

Figure 3a depicts the evaluation performance of both methods. We can observe that TEMPORL is $13.6\times$ faster than its vanilla counterpart to reach a reward of 0.5, and $12.4\times$ faster to reach a reward of 1.0 (i.e., always reach the goal). Figure 3b shows the number of required steps in the environment, as well as the number of decision steps. We can observe that TEMPORL is capable of finding a successful policy much faster than vanilla $\mathcal{Q}$-learning while requiring far fewer decision steps. Further, TEMPORL recovers the optimal policy quicker than vanilla $\mathcal{Q}$-learning. Lastly we can observe that after having trained for $\sim$ 6 000 episodes TEMPORL starts to increase the number of decision points. This can be attributed to all skip values of an action having converged to the same value. Our implementation selects the skip action at random instead of always selecting the largest skip as a tie-breaker, which would keep the number of decisions as small as possible.

Table 1a summarizes the result on all environments in terms of normalized area under the reward curve and in terms of number of decisions when using a linearly decaying ϵ-greedy schedule. In terms of reward AUC, a value closer to 1.0 indicates that the agent not only was capable of learning to reach the goal but also that it was done quickly. For the number of decisions, a lower value is better as it indicates

[2] For evaluations using larger skips we refer to the appendix.



(a) (b)

Figure 3. Evaluation performance of tabular Q-learning agents on a 6×10 cliff environment (similar to Figure 2a) over 100 random seeds with a maximum of 100 steps per episode. The agents were trained with a linearly-decaying ϵ-greedy policy. t-Q is our proposed agent. (a) Achieved reward. (b) Length of executed policy ($\cdots$) and number of decisions (—) made by the policies. An optimal policy requires 15 steps to reach the goal but only 3 decision points. The lines/shaded area represent the mean/standard deviation over 100 seeds.

	Cliff		Bridge		ZigZag	
	Q	t-Q	Q	t-Q	Q	t-Q
Reward	0.92	**0.99**	0.75	**0.97**	0.57	**0.92**
Decisions	27.9	**5.2**	49.5	**5.0**	83.6	**7.9**

(a) linearly decaying ϵ-schedule

Reward	0.96	**0.99**	0.94	**0.98**	0.90	**0.96**
Decisions	21.7	**4.9**	21.4	**5.3**	35.6	**6.9**

(b) logarithmically decaying ϵ-schedule

Reward	**0.99**	**0.99**	0.98	**0.99**	0.95	**0.99**
Decisions	17.1	**5.1**	14.7	**5.2**	27.6	**7.1**

(c) constant $\epsilon = 0.1$

Table 1. Normalized AUC for reward and average number of decision steps. Both agents are trained with the same ϵ schedule.

that fewer decisions were required to reach the goal, making a policy easier to learn. We can see that the TEMPORL agent readily outperforms the vanilla agent, always learning much faster as well as requiring far fewer decisions.

Sensitivity to Exploration As the used exploration mechanism can have a dramatic impact on agent performance we evaluated the agents for three commonly used ϵ-greedy exploration schedules. In the cases of linearly and logarithmically decaying schedules, we decay ϵ over all 10 000 training episodes, starting from 1.0 and decaying it to 0 / 10^{-5}, respectively. In the constant case we used a constant ϵ of 0.1.

As would be expected, we observe that too much exploration (linear) and too little exploration (log) are both detrimental to the agent's performance, see Table 1. However, we can also see that TEMPORL already performs quite well using suboptimal exploration strategies. TEMPORL outperforms its vanilla counterpart in all cases, showing the effectiveness of our proposed method.

In the difficult *ZigZag* world, both agents experience the largest improvement when switching from a linearly decaying schedule to a constant. Due to the difficulty of the environment, a vanilla agent takes much longer than a TEMPORL agent in propagating reward information when reaching the goal. When learning with a constant ϵ this leads to $5.4\times$ speedup over the vanilla variant to reach a mean reward of 0.5 and a $5.8\times$ speedup to reach a mean reward of 1.0.

5. Conclusion

We introduced skip-connections into the existing MDP formulation to propagate information about future rewards much faster. Based on the concept of skip-MDPs, we presented a learning mechanism that makes use of existing and well understood learning methods. We demonstrated that our new method, TEMPORL is capable of learning not only how to act in a state, but also *when* a new action has to be applied, without the need for prior knowledge about the environment. We empirically evaluated our method in a tabular setting.

As pointed out by Huang et al. (2019), observations might be costly. In such cases, we could make use of TEMPORL to learn how to behave and when new actions need to be taken; then, when using the learned policies, we could use the learned skip behaviour to only observe after having executed the longest skips possible.

All in all, we believe that TEMPORL opens up new avenues for RL methods to be more sample efficient and to learn complex behaviours. As future work, we plan to study TEMPORL with neural network function approximators as well as how to employ different exploration policies when learning the skip policies and behaviour policies.

Acknowledgements

We thank the reviewers for their insightful feedback. The authors acknowledge funding by the Robert Bosch GmbH.

References

Bacon, P., Harb, J., and Precup, D. The option-critic architecture. In *Proc. of AAAI'17*, 2017.

Baird, L. C. Reinforcement learning in continuous time: Advantage updating. In *Proc. of ICNN'94*, volume 4, pp. 2448–2453. IEEE, 1994.

Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., and Mordatch, I. Emergent tool use from multi-agent autocurricula. In *Proc. of ICLR'20*, 2020.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.*, 47:253–279, 2013.

Braylan, A., Hollenbeck, M., Meyerson, E., and Miikkulainen, R. Frame skip is a powerful parameter for learning to play atari. In *Proc. of Workshops at AAAI'15*, 2015.

Doya, K. Reinforcement learning in continuous time and space. *Neural Computation*, 12(1):219–245, 2000.

Harb, J., Bacon, P., Klissarov, M., and Precup, D. When waiting is not an option: Learning options with a deliberation cost. In *Proc. of AAAI'18*, pp. 3165–3172, 2018.

Harutyunyan, A., Vrancx, P., Bacon, P., Precup, D., and Nowé, A. Learning with options that terminate off-policy. In *Proc. of AAAI'18*, pp. 3173–3182, 2018.

Huang, Y., Kavitha, V., and Zhu, Q. Continuous-time markov decision processes with controlled observations. In *Proc. of Allerton'19*, pp. 32–39. IEEE, 2019.

Khan, A., Feng, J., Liu, S., and Asghar, M. Z. Optimal skipping rates: training agents with fine-grained control using deep reinforcement learning. *Journal of Robotics*, 2019, 2019.

Khetarpal, K. and Precup, D. Learning options with interest functions. In *Proc. of AAAI'19*, pp. 9955–9956, 2019.

Lakshminarayanan, A. S., Sharma, S., and Ravindran, B. Dynamic action repetition for deep reinforcement learning. In *Proc. of AAAI'17*, pp. 2133–2139, 2017.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *Proc. of ICLR'16*, 2016.

Mankowitz, D. J., Mann, T. A., Bacon, P., Precup, D., and Mannor, S. Learning robust options. In *Proc. of AAAI'18*, pp. 6409–6416, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Precup, D., Sutton, R. S., and Singh, S. P. Theoretical results on reinforcement learning with temporally abstract options. In *Proc. of ECML'98*, pp. 382–393, 1998.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *Proc. of ICML'15*, pp. 1312–1320, 2015.

Sharma, S., Lakshminarayanan, A. S., and Ravindran, B. Learning to repeat: Fine grained action repetition for deep reinforcement learning. In *Proc. of ICLR'17*, 2017.

Stolle, M. and Precup, D. Learning options in reinforcement learning. In *Proc. of SARA'02*, pp. 212–223, 2002.

Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999.

Tiganj, Z., Shankar, K. H., and Howard, M. W. Scale invariant value computation for reinforcement learning in continuous time. In *Proc. of AAAI Spring Symposia'17*, 2017.

	$\mathcal{Q}$	t-$\mathcal{Q}$														
J	0	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\mathcal{R}$	0.57	0.63	0.76	0.87	0.93	0.93	0.92	0.91	0.90	0.91	0.88	0.87	0.86	0.87	0.85	0.84
D	83.6	36.5	20.6	13.2	10.1	8.3	7.7	7.8	7.5	7.4	7.6	7.4	7.6	7.6	7.8	7.4

(a) linear decaying ϵ-schedule

$\mathcal{R}$	0.90	0.91	0.93	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.95	0.96	0.95	0.95
D	35.6	21.7	14.9	11.6	9.5	8.6	6.4	6.3	6.5	5.9	6.1	6.2	7.0	6.8	7.0	6.0

(b) logarithmic decaying ϵ-schedule

$\mathcal{R}$	0.95	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98
D	27.6	15.8	12.0	9.1	8.2	7.8	6.8	6.9	6.7	7.1	6.6	7.2	6.2	6.5	7.0	6.9

(c) constant $\epsilon = 0.1$

Table 2. Normalized AUC for reward and average number of decision steps for varying maximal skip-lengths J. All agents are trained with the same ϵ schedule. $\mathcal{R}$ denotes normalized area under the reward curve and D the average number of decision steps. Values are results of running 10 random seeds. Columns 1 and 7 are equivalent to columns 5 & 6 in Table 1.

A. Appendix

A.1. Influence of the Maximum Skip-Length

The maximum skip length J is a crucial hyperparameter of TEMPORL. A too large value might lead to many irrelevant choices which the agent has to learn to ignore; whereas a too small value might not reduce the complexity of the environment sufficiently enough, leading to barely an improvement over the vanilla counterpart. To evaluate the influence of the hyperparameter on our method we trained various TEMPORL agents with varying maximal skip-lengths, starting from 2 up to 16. Larger skips than 10 will never be beneficial for the agent as the agent is guaranteed to run into a wall for some steps. Depending on where in the environment the agent is located, smaller skip-values might allow it to quickly traverse through the environment.

Table 2 shows the influence of J on the ZigZag environment (see Figure 2c). In this environment, the largest skip value that is possible without running into a wall is 6. Thus, small skip values up to 5 quickly improve the performance over the vanilla counterpart, not only in terms of anytime performance but also in terms of required decisions. In the case of a suboptimal exploration policy, in the form of linearly decaying ϵ-greedy schedule (see Table 2a), larger skip-values quickly lead to a decrease in anytime performance, as the agent has to learn to never choose many non-improving skip actions.

For a more suiting exploration policy, too large skip-actions do not as quickly degrade the anytime performance of our TEMPORL agents. In the case of a logarithmically decaying ϵ schedule (Table 2b), we can see that skip sizes larger or equal than 12 start to negatively influence the anytime performance, whereas with a constant ϵ schedule only a skip-size of 16, nearly 3 times as large as the largest sensible choice, has a negative effect.

It is worth noting that all evaluated skip-sizes J result in better anytime-performance and a lower number of required decision points compared to vanilla $\mathcal{Q}$-learning, for all considered exploration strategies. In future work, we will study how to allow TEMPORL to select large skip-actions without needing to learn to distinguish between many irrelevant choices. One possible way of doing this could be by putting the skip-size on a log scale. For example using $\log_2$ could result in only 10 actions where a TEMPORL agent could skip up to 1024 steps ahead but would still be able to exert fine control with the smaller actions.

A.2. Generalization

So far we studied TEMPORL only in a tabular setting, and thus we can only conjecture about its generalization capabilities if we combine TEMPORL with deep reinforcement learning. We guess that for fine-grained time environments TEMPORL will be able to improve learning speed across environments that operate on a similar time-scale. For example in games like Super Mario, multiple levels require an agent to reach the right side of the screen and skipping behaviour on one level is likely transferable to another level. A TEMPORL agent might be able to learn to anticipate when it will collide with an enemy,

leading to a negative reward, and thus only repeat the action *go_right* until the *jump* action allows it to avoid the negative reward. As situations like this are seldom unique to a single level, we expect that TEMPORL can also improve generalization to environments that exhibit similar elements.