Learning When to Switch between Skills in a High Dimensional Domain

Timothy A. Mann and **Daniel J. Mankowitz** and **Shie Mannor**
Electrical Engineering
The Technion
Technion City, Haifa, Israel

Introduction

Complex problems are often easier to model with skills[1] than primitive (single time-step) actions (Stone, Sutton, and Kuhlmann 2005). In addition, skills have been shown to speed up the convergence rates of planning algorithms both experimentally (Sutton, Precup, and Singh 1999; Silver and Ciosek 2012) and theoretically (Mann and Mannor 2014). Skills are generally designed by a domain expert, but designing a 'good' set of skills can be challenging in high-dimensional, complex domains. In some cases, the skills may contain useful prior knowledge but cannot solve the task, resulting in a sub-optimal solution or no solution at all. Given a 'poor' set of skills, we would like to dynamically improve them.

Sutton, Precup, and Singh 1999 suggest Interrupting Options (IO) whereby the agent switches skills whenever there exists another skill with higher value than continuing the current skill. Mankowitz, Mann, and Mannor 2014 prove that the IO process converges on the set of skills with optimal switching rules (under mild assumptions). While the potential of IO is promising, previous experiments only showed the advantage of IO in tasks with few states.

We experiment with Space Invaders (SI, Figure 1) via the Arcade Learning Environment (ALE) using the 1024-bit RAM image as the state (Bellemare et al. 2013). The primitive actions are combinations of move left, move right, do nothing, and shoot. Our hypothesis is that *IOs can improve performance compared to learning with a fixed set of skills despite the fact that SI has a large state-space.* To test our hypothesis, we constructed a naive set of skills for the domain where each skill repeats one of the primitive actions over-and-over until a termination condition occurs. However, designing a rule that determines the optimal time to switch between skills is challenging, so we gave each skill a constant probability of switching at each timestep. We call the set of skills with this naive termination rule the initial skills set, and our experiments compared learning with the initial skill set to learning with a skill set whose switching rules are dynamically adapted by the IO rule.

[1]Here skills are a special case of options (Sutton, Precup, and Singh 1999).

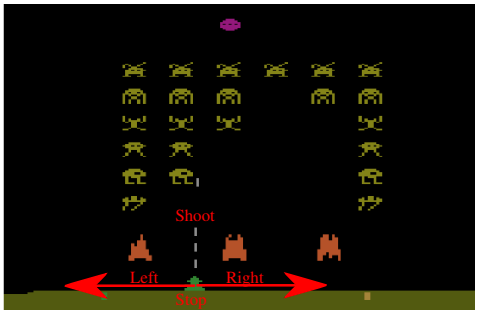


Figure 1: Space Invaders (SI)

Our experiment does not compare the performance of algorithms reported in previous literature (e.g. SARSA (Bellemare et al. 2013)), since those algorithms operated directly on the space of primitive actions. The initial skill set was chosen to test our hypothesis, which is about the effectiveness of IO in a problem with a large state-space. We do not claim that the skills used in our experiment are better than primitive actions. Instead our experiment focuses on demonstrating that the IO rule can improve an initial skill set by learning when to switch between skills. We leave the problem of deriving a good initial set of skills to future work (see Discussion & Conclusion).

The main contribution of this paper is showing that IOs can improve the initial skill set in a high-dimensional, complex domain by learning when to switch between skills. Furthermore, we discuss some of the pitfalls we ran into while trying to get IO to work.

Background

A skill $\sigma = \langle \pi, \beta \rangle$ is a temporally extended control structure where π is the intra-skill control policy that determines the actions to take while executing the skill and $\beta(s)$ is the probability that the skill will terminate in state $s \in S$. Let $Q(s, \sigma)$ be the action-value function and $V(s) = \max_\sigma Q(s, \sigma)$ denote the greedy value function. For a state $s \in S$ and a skill σ, the *IO rule* states that when executing σ, if we encounter a state s where

$$Q(s, \sigma) < V(s) - \Delta \ , \tag{1}$$

then terminate the skill and choose a new skill $\sigma' = \arg\max_j Q(s,j), \forall j \in \Sigma$ where Σ is the set of skills and $\Delta \geq 0$ is a small gap that prevents switching when the difference in value is negligible. $\Delta = 5$ in our experiments, which is the number of points received for shooting a green space invader[2]. The implication is that IO only switches between skills when there is a significant advantage for doing so.

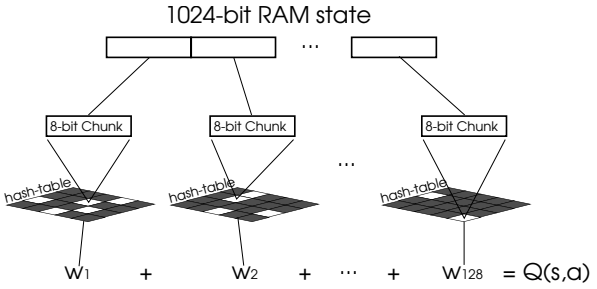


Figure 2: State hashing scheme used to approximate the action-value (Q-) function.

Experiment and Results

We compared RL with and without IOs in the Space Invaders domain via ALE (Bellemare et al. 2013), using the 1024-bit RAM dump at each timestep as the state-space.

Algorithms: Many RL algorithms can easily be adapted to operate over skills rather than primitive actions. All algorithms in our experiment were adapted to operate over skills.

We initially tried the SARSA algorithm with and without IOs. However, IOs performed poorly because they effectively stunted exploration. Once a non-zero reward was discovered for terminating a skill early, the IO rule would always terminate – never exploring what would happen if the skill was allowed to execute longer.

To remedy this problem, we used Q-learning (Watkins 1989) with learning rate $\alpha = 0.2$, discount factor $\gamma = 1$, and UCB-like exploration bonuses (Auer, Cesa-Bianchi, and Fischer 2002). The exploration bonuses cause the agent to explore terminating skills at different durations. To deal with the massive state-space, we introduced a hashing scheme (Figure 2). The same scheme was repeated for each skill to approximate the action-value function. The scheme broke down the 1024-bit RAM dump into 8-bit chunks. Each chunk was then hashed to an integer and a weight was associated with the the hashed integer. If the hashed value did not have any weight in the table, a default value 0 (plus a large bonus) was returned. To obtain an estimate for an action-value function, the weights associated with each chunk were averaged. This approach is similar to Q-learning with CMAC function approximation (Sutton and Barto 1998). We refer to our particular implementation as HashQ. Our experiment considers two conditions: (1) HashQ without IOs (just HashQ) and (2) HashQ with IOs (HashQIO). Note that for HashQIO, we switched on IOs at episode 20 to allow the algorithm a chance to derive a reasonable action-value function.

Initial Skills: For each primitive action $a_i \in A$, we created a skill $\sigma_i = \langle \pi_i, \beta_i \rangle$ where $\pi_i(s) = a_i$ and the termination rule was $\beta_i(s) = 0.01$ for all $s \in S$. So the initial skills executed the same primitive action over-and-over and terminated after about 100 timesteps. We needed the initial skill set to terminate randomly so that the agent could learn about the value of various termination states. Without this, the agent would never be able to apply the IO rule.

Results: Figure 3a compares the median cumulative rewards of a policy that selects from the initial set of skills via a uniform random distribution (Rand), HashQ, and HashQIO. Both HashQ and HashQIO outperform Rand. Furthermore HashQIO outperforms HashQ, suggesting that IOs improve on the termination rules of the initial skill set. Figure 3b compares the median duration of options during training episodes of HashQ and HashQIO, showing how HashQIO opportunistically terminates skills early to switch to better skills.

Discussion & Conclusion

The main problem that we encountered was accurately approximating the action-value function. When the action-value function is estimated poorly, the agent may switch to a skill that actually leads to a lower value than the current skill. Our results may be improved by combining IO with the more sophisticated DQN architecture (Mnih et al. 2013).

The second problem that we encountered was exploring the state-space. Random exploration only visited a small part of the state-space. We experimented with many exploration schemes before settling on one inspired by UCB. One significant advantage of UCB-like exploration is that it also encouraged intra-skill exploration by giving a large bonus to states that had not been visited before. This caused the IO rule to trigger whenever a new state was encountered during a skill's execution. This intra-skill exploration enables HashQIO to quickly find more optimal switching rules.

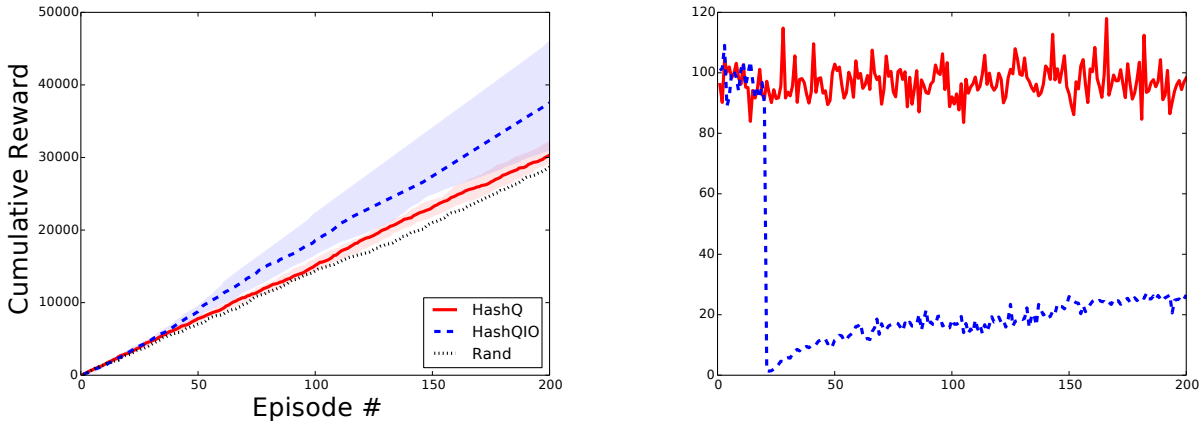
The last problem we encountered was designing the initial skills. We chose extremely naive skills for simplicity, but our results would likely improve significantly with a better initial skill set. For example, initial skills could be derived via the promising HyperNeat algorithm (Hausknecht et al. 2014).

This is the first result showing the advantage of IO in a high-dimensional domain. IO learns the optimal time to switch between skills and therefore improves the solution. This suggests that IOs may be valuable for scaling RL to high-dimensional and complex domains. In the future, we would like to look at how IOs perform in a range of high-dimensional games.

Acknowledgments

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n.306638.

[2]While $\Delta > 0$ invalidates the convergence result for IO, it adds robustness to in practice.



(a) (b)

Figure 3: (a) Median cumulative reward (shaded region: between 1^{st} and 3^{rd} quartiles) over 20 independent trials in Space Invaders for random, HashQ, and HashQIO. **HashQIO learns when to switch skills, which leads to higher cumulative reward.** (b) Median duration of options per episode for HashQ (solid red line) and HashQIO (dashed blue line).

References

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47(2-3):235–256.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47:253–279.

Hausknecht, M.; Lehman, J.; Miikkulainen, R.; and Stone, P. 2014. A neuroevolution approach to general atari game playing. *IEEE Transactions on Computational Intelligence and AI in Games*.

Mankowitz, D. J.; Mann, T. A.; and Mannor, S. 2014. Time regularized interrupting options. *Internation Conference on Machine Learning*.

Mann, T. A., and Mannor, S. 2014. Scaling up approximate value iteration with options: Better policies with fewer iterations. In *Proceedings of the 31^{st} International Conference on Machine Learning*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop 2013*.

Silver, D., and Ciosek, K. 2012. Compositional Planning Using Optimal Option Models. In *Proceedings of the 29th International Conference on Machine Learning*.

Stone, P.; Sutton, R. S.; and Kuhlmann, G. 2005. Reinforcement learning for robocup soccer keepaway. *Adaptive Behavior* 13(3):165–188.

Sutton, R., and Barto, A. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

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

Watkins, C. 1989. *Learning from Delayed Rewards*. Ph.D. Dissertation, University of Cambridge.