Lifelong Learning of Structure in the Space of Policies

Majd Hawasly and **Subramanian Ramamoorthy**
School of Informatics, University of Edinburgh
Informatics Forum, 10 Crichton Street
Edinburgh, EH8 9AB, United Kingdom
M.Hawasly@sms.ed.ac.uk S.Ramamoorthy@ed.ac.uk

Abstract

We address the problem faced by an autonomous agent that must achieve quick responses to a family of qualitatively-related tasks, such as a robot interacting with different types of human participants. We work in the setting where the tasks share a state-action space and have the same qualitative objective but differ in the dynamics and reward process. We adopt a transfer approach where the agent attempts to exploit common structure in learnt policies to accelerate learning in a new one. Our technique consists of a few key steps. First, we use a probabilistic model to describe the regions in state space which successful trajectories seem to prefer. Then, we extract policy fragments from previously-learnt policies for these regions as candidates for reuse. These fragments may be treated as options with corresponding domains and termination conditions extracted by unsupervised learning. Then, the set of reusable policies is used when learning novel tasks, and the process repeats. The utility of this method is demonstrated through experiments in the simulated soccer domain, where the variability comes from the different possible behaviours of opponent teams, and the agent needs to perform well against novel opponents.

1 Introduction

Consider a domestic robot that has to interact with humans to perform a specific task over an extended period of time. This robot should be made able to handle the variability in its task, arising, for example, from the inherent variability in human behaviour. This task could then be expressed as a family of related tasks, each slightly different from the others. These can be modelled as a family of Markov decision processes (MDPs) with a shared qualitative objective. These MDPs have the same state-action space, but differ in dynamics and reward processes. We need the robot to react *quickly* and *successfully* in a large fraction of the set of tasks, while it is only able to train extensively on few examples offline.

One should note that the range of variability is unbounded and unknown in advance for this kind of task. This calls for a lifelong learning approach, where the robot keeps learning from every new interaction in order to enhance its performance at the following trials.

To achieve quick response to a new situation, it is desirable to initialise the policy using knowledge acquired from previous instances. This is a question of transfer. In our setting, we consider all the instances to have the same qualitative objective in related circumstances, so it is plausible that the set of policies for the task family has a common structure. Discovering this commonality would give the robot a head start when learning in any new instance. We consider the case where the structure in the set of policies can be described using a collection of component behaviours (or subtasks). A structured policy then is a sequential composition of subtask policies. The framework of options (Sutton, Precup, and Singh 1999) is one example of policy space abstraction, where an *option* is a macro-action, with a domain of applicability and a termination condition, that can be be sequenced in a reinforcement learning setting. Options have been used previously for transfer (e.g. (Konidaris and Barto 2007)), and we will use them to represent the common subtasks.

Describing the structure in policy space of a family of tasks is not easy. It has been approached previously for tasks that have a simple mapping between the task space and the policy space, in that varying the task slightly in some direction would also vary the optimal policy slightly in some other direction. Mainly, these tasks can be parametrised using few parameters, and their policies can be parametrised as well. Then, policies can be automatically generated for a task using the mapping. An example of one successful approach in this regard is parametrized skills (Silva, Konidaris, and Barto 2012) where the structure in the parameter space of optimal policies is learnt. In this paper, we are interested in tasks whose policy spaces do not necessarily have a smooth continuous mapping to the task space, or ones wherein task parameters are unknown or unobservable (e.g. personal preferences of people).

The 'trick' we use to describe the structure in the policy space is by considering the corresponding structure in the state space, manifested as the state regions in which good behaviour is observed for many previously-learnt policies. Then, we assign previously-learnt policy fragments to these regions through a process of policy reuse. This will 'tag' parts of the state space with behaviours, emulating a structuring in the policy space.

The objective of this paper is to *present a framework that continually builds and refines a structured model of the policy space contingencies in a life-long process*. This kind of continual learning is necessary because the variability in the task is unknown beforehand but rather discovered incrementally. So, by incrementally considering good policies for a sequence of tasks drawn from a set, we aim to generate a more refined model in the form of good reusable subpolicies that are expressed as options.

We propose an incremental procedure that hypothesises a decomposition in state space by fitting a probabilistic model to sampled trajectories of learnt policies. Then, fragments of the learnt policies are assigned to the components of the probabilistic model, and these are described as options. These options are used when learning the following instance, to generate more sampled trajectories and hypothesise a refined decomposition.

The rest of the paper is organised as follows. The related literature is discussed in the next section, before presenting the technical setup of the paper. Then, ILPSS (for Incremental Learning of Policy Space Structure) is presented and described in detail. After that, we demonstrate empirical results, followed by a discussion of the merits of the proposed framework.

2 Related Work

Learning to act in a set of related tasks by leveraging the experience gained in some of them is a branch of Transfer Learning (Taylor and Stone 2009), or, in some versions, Multi-task Reinforcement Learning (MTRL). The idea is to use a set of source tasks to accelerate learning in a novel target task, different in rewards or dynamics. Some methods rely on explicit and observable parametrisation of the task space (Mehta et al. 2008; Silva, Konidaris, and Barto 2012), while others assume a distribution of variability, either known (Perkins, Precup, and others 1999), or approximated from previously seen source tasks (Tanaka and Yamamura 2003; Snel and Whiteson 2012). Bayesian methods use these distributions as priors to estimate the model of the novel target task (Sunmola and Wyatt 2006; Wilson et al. 2007; Wilson, Fern, and Tadepalli 2012). A different approach is to find a smaller task space, dubbed the agent-space, in which the related tasks are the same (Konidaris and Barto 2007).

Policy reuse is another Transfer Learning approach that deals with related tasks. Using a mechanism to choose the most similar seen task to the current one, policies can be used *as is* on the target task, or alternatively, they can be used to accelerate learning, e.g. by biasing the exploration scheme in the new task by the most similar learnt policy (Fernández and Veloso 2006).

One way to counter complexity in reinforcement learning is through exploitation of hierarchical structure in policies. Hierarchical reinforcement learning (Barto and Mahadevan 2003) provides a number of tools to organise and learn a hierarchical policy for a specific task, with the framework of options (Sutton, Precup, and Singh 1999) being one of the more flexible. One way to uncover the hierarchy automatically in a task requires discovering subgoals that are essential to the achievement of the task objective. One way to describe this has been through the notion of a *bottleneck* which is a landmark state that successful trajectories tend to go through, while unsuccessful ones do not. Finding bottlenecks has been approached in many ways, including state visitation frequencies (Stolle and Precup 2002; Şimşek and Barto 2004), and through graph-theoretic properties of the transition graph, like Max-flow/Min-cut (Menache, Mannor, and Shimkin 2002) and betweenness (Şimşek and Barto 2009).

Exploiting abstraction for transfer purposes has seen some success, where the abstraction is learnt from many source instances rather than the single task in the case of classical HRL. Using options, Konidaris and Barto (2007) introduce portable options which are abstract actions defined in the agent-space rather than in the full problem-space, so that they can be used in any problem that share that reduced representation. Skills are extracted and chained to construct skill trees in (Konidaris et al. 2010) from expert demonstrations, while generalisation for parametrisable tasks can be achieved through the discovery of smooth low-dimensional spaces where the policies of skills lie (Silva, Konidaris, and Barto 2012).

Also, there have been prior work investigating the use of probabilistic models in hierarchical reinforcement learning. In (Manfredi and Mahadevan 2005), a graphical model for both state and policy abstraction is trained from sample trajectories using Expectation-Maximization (EM) algorithm, but for a single task.

3 Setup

3.1 Markov decision process

We consider tasks that can be modelled as discrete-time Markov decision processes (MDP). An MDP m is the tuple (S, A, T, R), where S is a finite state space, A is a finite action space, $T : S \times A \times S \to [0, 1]$ is the dynamics of the world, and $R : S \times A \times S \to \mathbf{R}$ is the reward process that encodes the goal of the task. A (Markov) policy for an MDP is a stochastic mapping from states to actions, $\pi : S \times A \to [0, 1]$, and the optimal policy π^* is the policy that maximises expected cumulative reward.

We consider episodic, goal-oriented tasks, in which termination occurs either when the agent reaches specific *goal states*, or when the episode elapses.

3.2 Family of MDPs

The agent faces a sequence of instances of its task, $m_i, i = 1, 2, 3, \cdots$, each slightly different. We model this variability by a set of MDPs $\mathcal{M}$ with a common objective. These MDPs share the state-action space $S \times A$, but the dynamics $T : S \times A \times S \to [0, 1]$ and the reward process $R : S \times A \times S \to \mathbf{R}$ may be different for each member of the family $m \in \mathcal{M}$.

We assume that the variability model of the task, from which the instances are drawn, is *unknown* to the agent beforehand.

3.3 Options

The framework of options (Sutton, Precup, and Singh 1999) is one approach to hierarchical reinforcement learning (Barto and Mahadevan 2003) that employs a form of generic temporally-extended actions. An option is the three-tuple $\langle I, \pi, \beta \rangle$: $I \subseteq S$ is the initiation state set where the option is allowed to be invoked, π is a (Markov or semi-Markov) policy which is followed when the option is invoked, and β is a termination probability distribution over the state space which encodes stochastically the success and failure criteria of the option. Options are flexible objects that generalise primitive actions. A primitive action is simply an option that terminates exactly after one step. For this model, learning algorithms designed for MDPs can be extended to solve SMDPs (semi-Markov decision processes) that emerge from the temporal abstraction brought by the use of options.

4 ILPSS: Incremental Learning of Policy Space Structure

Now, we describe our proposed framework. Figure 1 gives a snapshot of its operation. It comprises the following key steps:

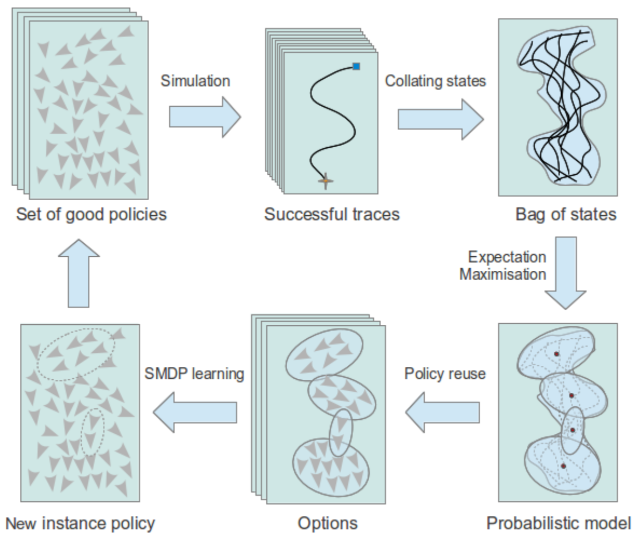

Figure 1: A high level caricature of ILPSS.

1. Starting from a collection of n good policies $\pi_0, \pi_1, \ldots, \pi_n$ of some samples of the task $m_0, m_1, \ldots, m_n \in \mathcal{M}$, we sample a set of complete episode state traces $\tau = [s_0, \ldots, s_{term}] \in S^{|\tau|}$.
2. We label the trajectories with respect to their success $(+)$ or failure $(-)$ in reaching the goal. An episode is successful if it terminates at the goal state, while episodes that elapsed before reaching the goal state are considered failures. This is especially relevant when dealing with highly-stochastic tasks, like interacting with human subjects. Only the successful traces $\bigcup_{i=1}^{n}\{\tau_i^+\} = \mathcal{D}$ are used.
3. We search for a generative probabilistic model of $\mathcal{D}$. The combination of such a model with policy reuse is what defines ILPSS. In more concrete terms, we want to find a model that maximises the log likelihood of the data

$$\begin{aligned} \log P[\mathcal{D};\theta] &= \sum_i \log P[\{\tau_i^+\}|\theta] & (1)\\ &= \sum_i \sum_\tau \log P[\tau_i^+;\theta], & (2) \end{aligned}$$

assuming independence of traces for a specific task instance given the model, as well as independence of different instances' traces, being drawn from the same model which summarises all the correlations. Initially, we ignore time and use these traces as *bags of states*,

$$\log P[\tau;\theta] = \sum_{s\in\tau} \log P[s|\theta] \qquad (3)$$

The model we use here for $P[s|\theta]$ is a mixture model of $|K|$ multivariate Gaussian kernels, each represented by its mean μ_k and covariance matrix Σ_k. However, any suitable alternate model can also be used.

$$\log P[s|\theta] = \log \sum_k p_k \mathcal{N}(s;\mu_k,\Sigma_k) \qquad (4)$$

The choice of the model is task-specific and does not affect the operation of ILPSS.

For our choice of model, the parameters are the weights $\{p_k\}$, means $\{\mu_k\}$ and covariances $\{\Sigma_k\}$. We allow $\{p_k\}$ to be instance-specific, i.e. each instance has it own weighting of the components, $\{p_k\} = \bigcup_{i=1}^{n}\{{p_k}^i\}$, while $\{\mu_k\}$ and $\{\Sigma_k\}$ are instance-independent, i.e. the instances share the exact same components. That is, we push toward finding a common set of mixture components across the different instance policies, regardless of their relative significance in various instances.

The mixture components define kernels in the state space, and they are an important component of our representation of the policy space, in that we probabilistically assign to the regions defined by them ready policy fragments. A fragment for a specific kernel is borrowed from a learnt policy of a previous instance, constrained to the respective kernel's state space region. The scale (or weight) of a component k in a task instance i, as captured by ${p_k}^i$, is ignored as it is irrelevant from a structure-learning point of view.

To fit the model, we choose the desired number of kernels $|K|$ and use an adapted Expectation-Maximisation (EM) algorithm (Dempster, Laird, and Rubin 1977) that takes our constraint into account. In the Expectation phase, the kernels are initialised randomly, and the model weights, or responsibilities, are computed accordingly. In the Maximisation phase, the parameters of the model are recomputed using the responsibilities. We calculate the means and covariance matrices using the complete data set, while the weights $\{{p_k}^i\}$ use only the data from instance i.
4. Then, each of the discovered kernels spawns a set of policy fragments, each borrowed from a different previously-learnt policy. We describe these policies by a set of options $\mathcal{O}_k$. The domain and the termination condition of $\mathcal{O}_k$ options are both the PDF defined by the kernel. That is, an

option from $\mathcal{O}_k$ is allowed to start at a particular state s with a probability equal to $\mathcal{K}(s, \mu_k)$, and will continue, stochastically, with probability equal to $\mathcal{K}(s, \mu_k)$ as well, and terminate with probability $1 - \mathcal{K}(s, \mu_k)$. The option policies are the restriction of the previously-learnt policies $\pi_0, \pi_1, \ldots, \pi_n$ to the corresponding option domain. That is, we are reusing policies of previous tasks in controlled regions of state space where they fared well in experience.

5. Next, the agent is presented with another task instance, m_{n+1}. In a similar fashion, the agent will learn a policy π_{n+1} for that instance with the exception that in this case the agent is allowed to use the new options in learning. The resultant policy will contain, in addition to normal actions, pointers to $\pi_0, \ldots, \pi_n$.

6. We adopt the exact same procedure on the new policy π_{n+1} (extracting traces, bag of states, probabilistic model, then options) and this continues in what can be described as a lifelong learning process.

The full procedure is stated in Algorithm 1.

Algorithm 1 ILPSS: Incremental Learning of Policy Space Structure

```
Require: the number of kernels $|K|$, input policies $\pi_0, \ldots, \pi_n$.
 1: for every new instance do
 2:     Generate state traces $\{\tau\}$ from input policies through simulation/ runtime recording.
 3:     Label and extract successful traces $\{\tau^+\}$.
 4:     Create 'bags of states' data set $\mathcal{D} = \bigcup_{i=1}^{n} \{\tau_i^+\}$.
 5:     Fit a probabilistic model to $\mathcal{D}$ using EM algorithm, generating a set of kernels $K$ in state space.
 6:     Create a set of options $\{\mathcal{O}_k\}$ using the discovered kernels $K$ and the input policies, with $o_k^i = \langle \mathcal{K}(., \mu_k), \mathcal{K}(., \mu_k), \pi_i \rangle \in \mathcal{O}_k$.
 7:     Learn a policy for the new instance with $\{\mathcal{O}_k\}$ by SMDP learning.
 8:     Add the learnt policy to the input policy.
 9:     $n \leftarrow n + 1$.
10: end for
11: return Option set $\{\mathcal{O}_k\}$.
```

5 Empirical Results

We use the domain of simulated robotic soccer to demonstrate the approach. The task in this experiment is a training drill for 2 vs. 2 players, in which the team with ball possession tries to cross a specific line in the field *with the ball*. The task is episodic, starting with the agents in set positions, and terminating either successfully when the goal is achieved, or negatively when the adversaries intercept the ball, kick it out of the training region (35m$\times$35m), or when the episode elapses (100 time steps). The experiment was conducted using Robocup 2D Simulation League Soccer Server (Noda and Matsubara 1996) and the Keepaway extension (Stone, Sutton, and Kuhlmann 2005). The setup is shown in Figure 2.

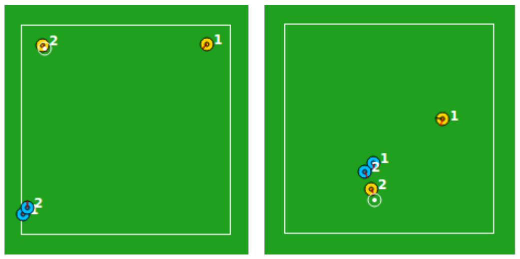

Figure 2: The experiment setup. LEFT: the attacking team starts from the top corners, while the defending team starts from the bottom. RIGHT: the task of the attackers is to reach the bottom line with the ball, with the middle point having the highest reward.

The state space is defined using 9 continuous state features describing the position and orientation of the agent with respect to other agents and the goal, while the action space comprises 3 basic options, for holding the ball, passing to team mate and dribbling toward the goal. The action set is enriched later with discovered options.

Only the agent that has the ball is learning, while others use fixed stochastic behaviours. The variability is in opponent behaviour, in that different opponent teams have different tendencies toward the ball and the goal line. In the experiment four opponent teams are used. The first two only intercept the ball, with different coordination protocols between the players. One opponent in the third type intercepts the ball while the other protects the goal line, while both opponents protect the goal in the fourth type, and only leave it for the ball with small probability.

The opponents are presented to the learning agents sequentially, with structure extraction happening after each individual trial. After every trial, traces are used to generate 5 kernels, which generate 5 options. After experiencing the first three opponents and learning a repertoire of options, the performance against the last opponent using these options is compared to learning the same task from scratch. The results are shown in Figure 3.

As it is clear, using the set of options discovered by ILPSS gives a head start in performance. This shows that ILPSS is able to produce useful abstractions in the policy space that would allow faster convergence in a novel instance.

6 Extensions

To avoid the explosion in the size of the option set that is carried out from one step to the other in ILPSS, a simple 'discounting' procedure is proposed: the options that do not get used in future learning trials will be less likely to survive in the set. One way to achieve this discounting is through reducing the probability with which the option may be selected in future, i.e. shrinking the option's initiation set. Because we define the initiation set probabilistically using a kernel function, we can achieve this effect by multiplying it with a scalar smaller than one in every trial where the option is not selected for the optimal policy. This will have the effect of shrinking the useless option domains with time. Finally, the

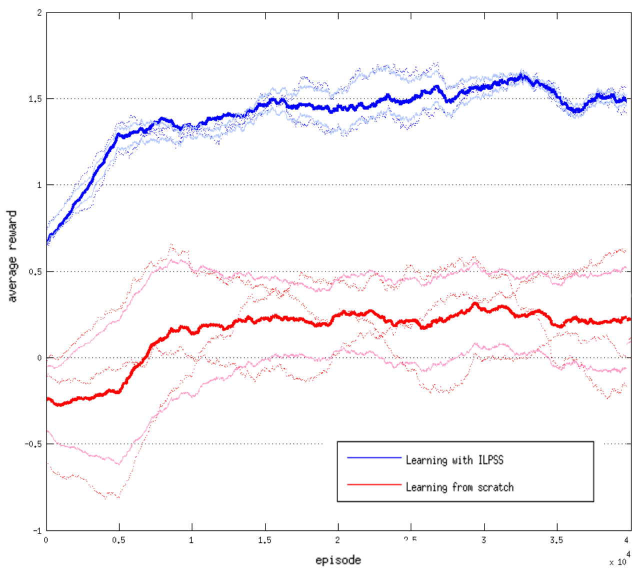

Figure 3: Average reward achieved against a novel opponent using options acquired by ILPSS against three other opponents in three previous task instances (BLUE), compared to learning anew (RED).

option will be pruned out of the set after crossing a threshold on effective option sizes.

Discounting unused options helps reduce the size of the skill set the agent has to carry to new instances. Effectively, this is a kind of 'forgetting' of old behaviours. Still, this does not mean the agent will lose what has been learnt completely, as the iterative nature of the learning process pushes the agent to try learnt skills in new instances and to form new, extended skills based on them using reinforcement learning, leading to useful generalisation of acquired skills.

7 Discussion

One intuition behind the incremental nature of the framework is in 'biasing' the evolved decomposition toward finding common components that would be useful for lifelong learning . Using the previously-learnt options during learning a new task instance would encourage the agent to search for a solution first in the policy subspace spanned by behaviours that proved useful in the past, before extending to the wider policy space. If a satisfactory solution can be found early, behavioural commonalities which are relevant to many of the experienced instances would be developed and maintained. We argue that this kind of structure is what is needed when facing a novel instance of a task.

Many methods of learning the hierarchy in hierarchical reinforcement learning use the notion of *bottleneck* to describe the states that are essential in the achievement of a single task. These are landmark states that successful trajectories tend to go through (e.g. doorways in the rooms environment), and thus found using visitation frequency or through specific attributes in transition graphs. This concept of bottlenecks works best for small, discrete state spaces and single tasks, and its target is to locate possible subgoals, leaving the policies to achieve them to be separately learnt. On the other hand, ILPSS generalises that concept and takes a probabilistic approach in defining the interesting regions in the state space, which might be more appropriate in large and continuous state spaces. Also, it immediately discovers where the *existing* policies might be useful, avoiding the two-step process of explicit discovery of subgoals followed by policy learning.

Another method which defines skills that have probabilistic domains is the SKILLS algorithm (Thrun, Schwartz, and others 1995). However, the aim there is to find a compact set of macro-actions that minimises the description length (DL) of the action set in a single task. The policy of a skill is learnt in a way that balances performance loss with compactness gains, while for ILPSS, policies are chosen to support an extensive family of tasks, while compactness is observed through the option discounting process.

A work of Foster and Dayan (2002) investigates the structure in the space of value functions of optimal policies for related tasks. For that, they employ a mixture model as the generative process of value functions. The components of that mixture represent the discontinuity in the value function caused by inherent properties of the task (e.g., location of walls and barriers in a room environment). Then, the model is used to accelerate learning in new instances through augmenting the state space with the discovered higher-level features. Our framework defines structure in policy space, implicitly, through probability distributions over state space and reusable policy fragments. Our components describe not state regions where the value function is smooth but rather the regions where many learnt policies are stable and performing well. Also, our framework only requires sample traces from good policies rather than complete explicit representation of the value function, which may be infeasible in big worlds, or before learning converges to a stable value function.

The use of Gaussian kernels to describe option domains may not be the optimal choice, as they suggest a kind of symmetry over various dimensions which may not be true. However, they are an intuitive choice from a computational point of view. More work needs to be done on suitable shapes of the boundary of a skill in state space.

8 Conclusion

We introduce a framework for learning and refining a structural description of the space of policies for a set of qualitatively-related task instances. The aim of the model is to enable an agent to react quickly in novel instances of the same task. We employ a principled probabilistic method to decompose the state space, and relate the learnt abstraction with policy fragments through policy reuse. The resulting structure is maintained using a set of temporally-extended options. We note that learning online is essential for extracting useful decompositions in policy space.

This method does not require explicit representation of the space of policies, and it does not rely on the optimality of the input policies, which allows it to scale well. It only uses a set of trajectories of successful trials and the policies that generated them to enable the agent to produce a rough and quick solution to a novel instance, which can then be refined by learning.

Testing this framework on large problems to understand its scalability is a topic of future work, as well as understanding the effects of state abstraction on the produced action abstraction.

9 Acknowledgements

This work has taken place in the Robust Autonomy and Decisions group within the School of Informatics, University of Edinburgh. Research of the RAD Group is supported by the UK Engineering and Physical Sciences Research Council (grant number EP/H012338/1) and the European Commission (TOMSY Grant Agreement 270436, under FP7-ICT-2009.2.1 Call 6).

References

Barto, A., and Mahadevan, S. 2003. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems* 13(4):341–379.

Dempster, A.; Laird, N.; and Rubin, D. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 1–38.

Fernández, F., and Veloso, M. 2006. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 720–727. ACM.

Foster, D., and Dayan, P. 2002. Structure in the space of value functions. *Machine Learning* 49(2):325–346.

Konidaris, G., and Barto, A. 2007. Building portable options: Skill transfer in reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, volume 2, 895–900.

Konidaris, G.; Kuindersma, S.; Barto, A.; and Grupen, R. 2010. Constructing skill trees for reinforcement learning agents from demonstration trajectories. *Advances in neural information processing systems* 23:1162–1170.

Manfredi, V., and Mahadevan, S. 2005. Hierarchical reinforcement learning using graphical models. In *Proceedings of the ICML05 Workshop on Rich Representations for Reinforcement Learning*, 39–44.

Mehta, N.; Natarajan, S.; Tadepalli, P.; and Fern, A. 2008. Transfer in variable-reward hierarchical reinforcement learning. *Machine Learning* 73(3):289–312.

Menache, I.; Mannor, S.; and Shimkin, N. 2002. Q-cut: dynamic discovery of sub-goals in reinforcement learning. *Machine Learning: ECML 2002* 187–195.

Noda, I., and Matsubara, H. 1996. Soccer server and researches on multi-agent systems. In *Proceedings of the IROS-96 Workshop on RoboCup*. Citeseer.

Perkins, T.; Precup, D.; et al. 1999. Using options for knowledge transfer in reinforcement learning. *University of Massachusetts, Amherst, MA, USA, Tech. Rep.*

Silva, B. D.; Konidaris, G.; and Barto, A. 2012. Learning parameterized skills. In Langford, J., and Pineau, J., eds., *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, 1679–1686. New York, NY, USA: Omnipress.

Şimşek, Ö., and Barto, A. 2004. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, 95. ACM.

Şimşek, Ö., and Barto, A. 2009. Skill characterization based on betweenness. In *In Advances in Neural Information Processing Systems 22*.

Snel, M., and Whiteson, S. 2012. Multi-task reinforcement learning: shaping and feature selection. *Recent Advances in Reinforcement Learning* 237–248.

Stolle, M., and Precup, D. 2002. Learning options in reinforcement learning. *Lecture Notes in Computer Science* 2371:212–223.

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

Sunmola, F., and Wyatt, J. 2006. Model transfer for markov decision tasks via parameter matching. In *Proceedings of the 25th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG 2006)*.

Sutton, R.; 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.

Tanaka, F., and Yamamura, M. 2003. Multitask reinforcement learning on the distribution of mdps. In *Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International Symposium on*, volume 3, 1108–1113. IEEE.

Taylor, M., and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research* 10:1633–1685.

Thrun, S.; Schwartz, A.; et al. 1995. Finding structure in reinforcement learning. *Advances in neural information processing systems* 385–392.

Wilson, A.; Fern, A.; Ray, S.; and Tadepalli, P. 2007. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, 1015–1022. ACM.

Wilson, A.; Fern, A.; and Tadepalli, P. 2012. Transfer learning in sequential decision problems: A hierarchical bayesian approach. In *ICML 2011 Unsupervised and Transfer Learning Workshop. JMLR W&CP, this volume*.