# Transfer Reinforcement Learning with Shared Dynamics

**Romain Laroche**
Microsoft Maluuba
2000 rue Peel, Montreal QC, Canada
romain.laroche@microsoft.com

## Abstract

This article addresses a particular Transfer Reinforcement Learning (RL) problem: when dynamics do not change from one task to another, and only the reward function does. Our method relies on two ideas, the first one is that transition samples obtained from a task can be reused to learn on any other task: an immediate reward estimator is learnt in a supervised fashion and for each sample, the reward entry is changed by its reward estimate. The second idea consists in adopting the *optimism in the face of uncertainty* principle and to use upper bound reward estimates. Our method is tested on a navigation task, under four Transfer RL experimental settings: with a known reward function, with strong and weak expert knowledge on the reward function, and with a completely unknown reward function. Results reveal that this method constitutes a major improvement for transfer/multi-task problems that share dynamics.

**Keywords:** Reinforcement Learning; Transfer Learning

# 1 Introduction

Reinforcement Learning (RL, [1]) is a framework for optimising an agent behaviour in an environment. It is generally formalised as a Markov Decision Process (MDP): $\langle \mathcal{S}, \mathcal{A}, R, P, \gamma \rangle$ where $\mathcal{S}$ the state space, and $\mathcal{A}$ the action space are known by the agent. $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, the Markovian transition stochastic function, defines the unknown dynamics of the environment. $R : \mathcal{S} \to \mathbb{R}$, the immediate reward stochastic function, defines the goal(s). In some settings such as dialogue systems or board games, $R$ can be inferred directly from the state by the agent, and in some others such as in robotics and in Atari games, $R$ is generally unknown. Finally, $\gamma \in [0, 1)$ the discount factor is a parameter given to the RL optimisation algorithm favouring short-term rewards. As a consequence, the RL problem consists in (directly or indirectly) discovering $P$, sometimes $R$, and planning.

Even when $R$ is unknown, $R$ is often simpler to learn than $P$: its definition is less combinatorial, $R$ is generally sparse, only a mean estimation is required, $R$ tends to be less stochastic than $P$, and finally it is frequently possible for the designer to inject expert knowledge: for instance, an adequate state space representation for $R$, the uniqueness of the state with a positive reward, its determinism or stochastic property, and/or the existence of $R$ bounds: $R_{min}$ and $R_{max}$.

Discovering (directly or indirectly) $P$ and $R$ requires collecting trajectories. In real world problems, trajectory collection is resource consuming (time, money), and Transfer Learning for RL [2, 3], through reuse of knowledge acquired from similar tasks, has proven useful in many RL domains: Atari games, robotics, or dialogue. In this article, we address the problem of Transfer Reinforcement Learning with Shared Dynamics (TRLSD), *i.e.* the transfer problem when $P$ is constant over tasks $\tau \in \mathcal{T}$, which thus only differ from each other by their reward functions $R_\tau$. This family of problems may be encountered for instance in robotics, where the robot agent has to understand the complex shared environment dynamics in order to perform high level tasks that rely on this understanding.

In this article, we advocate that experience gathered on a task can be indirectly and directly reused on another task and that transfer can be made at the transition sample level. Additionally, the *optimism in the face of uncertainty* principle allows to guide the exploration efficiently on a new task, thanks to the dynamics knowledge transferred from the other tasks. The combination of those two principles allows us to define a general algorithm for TRLSD, on a continuous state space, enabling the injection of task related expert knowledge.

Section 2 offers an overview of the known studies related to TRLSD, and introduces the principles of transition sample sharing between tasks. Then, Section 3 recalls the *optimism in the face of uncertainty* principle, explains how to apply it to our setting, and explores different ways of computing this optimism. Finally, Section 4 presents various experiments illustrating and demonstrating the functioning of our algorithms in Transfer RL and Multi-Task RL experiments. The experimental results demonstrate the significant improvement brought by our approach, in comparison with the state-of-the-art algorithms in TRLSD.

# 2 Background and Principle

To the authors knowledge, only two recent works were dedicated to TRLSD. First, [4] present the framework as a kind of hierarchical reinforcement learning, where composite and compoundable subtasks are discovered by generalisation over tasks. In order to do so, tasks share the successor features of their policies, which are invariant from one task to another. Their decomposition of the reward function from the dynamics is unfortunately restricted to policies characterising the successor features. Additionally, the theoretical analysis depends on $R_\tau$ similarities, which is not an assumption that is made in this article. Second, [5] address the same problem in a Multi-Task RL setting by sharing the value-function representation: they build a transition sample set for all tasks and apply generalised versions of Fitted-$Q$ iteration and Fitted Policy Iteration learning on those transitions as a whole. The generalisation amongst tasks occurs in the regularisation used in the supervised learning step of Fitted-$Q$ iteration (and policy iteration/evaluation).

Instead of sharing successor features or value-function representations, we argue that transition samples can be shared across tasks. A transition sample (or sample in short) is classically defined as a 4-tuple $\xi = \langle s, a, r, s' \rangle$, where $s$ is the state at the beginning of the transition, $a$ is the action performed, $r$ is the reward immediately received, and $s'$ is the state reached at the end of the transition. For Transfer and Multi-Task RL, it is enhanced with task $\tau$ to keep in memory which task generated the sample: $\xi_\tau = \langle \tau, s, a, r, s' \rangle$. Formulated in another way, $s$ is drawn according to a distribution depending on the behavioural policy $\pi^\tau$, $a$ according to the behavioural policy $\pi^\tau(s)$, $r$ according to the reward function $R_\tau(s)$ of task $\tau$ and $s'$ according to the shared dynamics $P(s, a)$.

As a consequence, with a transition sample set for all tasks $\Xi = \bigcup_{\tau \in \mathcal{T}} \Xi_\tau$, one can independently learn $\hat{P}$, an estimate of $P$, in a supervised learning way. In the same manner, with the sample set constituted exclusively of task $\tau$ transitions $\Xi_\tau$, one can independently learn $\hat{R}_\tau$, an estimate of the reward function expected value $\mathbb{E}[R_\tau]$, in a supervised learning way. This is what model-based RL does. In other words, if transition sample $\xi_\tau$ was generated on task $\tau$, and if task $\tau'$ shares the dynamics $P$ with $\tau$, then $\xi_\tau$ can be used for learning the dynamics model of task $\tau'$. The adaptation to non-stationary reward functions has been an argument in favour of model-based RL for twenty years. In particular, [6] applies it successfully on a task transfer with shared dynamics and similar reward functions on the inverted pendulum problem.

Nevertheless, this approach has never been theorised nor applied to Transfer or Multi-Task RL. We also advocate that learning the dynamics model $P$ is not necessary and that efficient policies can be learnt in a direct fashion: given a target task $\tau$, any transition sample $\xi_{\tau'} = \langle \tau', s, a, r, s' \rangle$ from any other task $\tau' \neq \tau$ can be projected on task $\tau$, just by modifying the immediate reward $r$ with $\hat{R}_\tau(s)$, the estimate of the reward function expected value $\mathbb{E}[R_\tau]$: $\psi_{\hat{R}_\tau}(\xi_{\tau'}) = \langle \tau, s, a, \hat{R}_\tau(s), s' \rangle$. The approach consists thus in

translating the transition sample set $\Xi$ into $\hat{R}_\tau$ estimate: $\Psi_{\hat{R}_\tau}(\Xi) = \{\psi_{\hat{R}_\tau}(\xi_{\tau'})\}_{\xi_{\tau'} \in \Xi}$, and then in using any off-policy RL algorithm to learn policy $\pi_\tau$ on $\Psi_{\hat{R}_\tau}(\Xi)$. The off-policy characteristic is critical in order to remove the bias originated from the behavioural policies $\pi^\tau$ controlling the transition sample set $\Xi$ generation. In our experiments, we will use Fitted-$Q$ Iteration.

## 3 Optimism in the Face of Uncertainty

In practice, using an estimate $\hat{R}_\tau$ of $R_\tau$ is inefficient in early stages, when only a few samples have been collected on task $\tau$ and reward has never been observed in most states, because the algorithm cannot decide if it should exploit or explore further. We generalise our approach to a reward proxy $\tilde{R}_\tau$ in Algorithm 1.

**Algorithm 1** Transition reuse algorithm

**Data:** $\Xi$: transition sample set on various tasks
**Data:** $\Xi_\tau \subseteq \Xi$: transition sample set on task $\tau$
Learn on $\Xi_\tau$ an immediate reward proxy: $\tilde{R}_\tau$
Cast sample set $\Xi$ on task $\tau$: $\Psi_{\tilde{R}_\tau}(\Xi)$
Learn on $\Psi_{\tilde{R}_\tau}(\Xi)$ a policy for task $\tau$: $\pi_\tau$

In order to guide the exploration, we adopt the well-known *optimism in the face of uncertainty* heuristic, which can be found in Prioritized Sweeping, R-MAX, UCRL, and VIME. In the optimistic view, $\tilde{R}_\tau$ is the most favourable plausible reward function. Only UCRL and VIME use an explicit representation of the exploration mechanism that is embedded into the transition and the reward functions. The way UCRL [7] separates the dynamics uncertainty from the immediate rewards uncertainty makes it more convenient to implement and the following of the article is built on it, but any other optimism-based algorithm could have been considered in its place. [8] is proposing an interesting alternative option for guiding the exploration in our setting.

UCRL [7] algorithm keeps track of statistics for rewards and transitions: the number of times $N(s,a)$ action $a$ in state $s$ has been performed, the average immediate reward $\hat{r}(s)$ in state $s$, and the observed probability $\hat{p}(s,a,s')$ of reaching state $s'$ after performing action $a$ in state $s$. Those statistics are only an estimate of their true values. However, confidence intervals may be used to define a set $\mathcal{M}$ of plausible MDPs in which the true MDP belongs with high probability. As said in last paragraph, UCRL adopts the *optimism in the face of uncertainty* principle over $\mathcal{M}$ and follows one of the policies that maximise the expected return in the most favourable MDP(s) in $\mathcal{M}$. The main idea behind the optimism exploration is the fact that mistakes will be eventually observed and knowledge of not doing it again will be acquired and realised through a narrowing of the confidence interval.

One UCRL practical problem is the need for searching the optimal policy inside $\mathcal{M}$ [9], which is complex and computer time consuming. In our case, we can however consider that $\hat{P}$ is precise enough in comparison with $\hat{R}_\tau$ and that dynamics uncertainty should not guide the exploration. Therefore, the optimal policy on $\mathcal{M}$ is necessarily the optimal policy of the MDP with the highest reward function inside the confidence bounds, *i.e.* $\tilde{R}_\tau(s)$ defined by the following equation:

$$\tilde{R}_\tau(s) = \hat{R}_\tau(s) + CI_\tau(s), \tag{1}$$

where $CI_\tau(s)$ is the confidence interval of reward estimate $\hat{R}_\tau$ in state $s$. Afterwards, the optimal policy can be directly learnt on data $\Psi_{\tilde{R}_\tau}(\Xi)$ with Fitted-$Q$ Iteration.

Another UCRL limitation is that it does not accommodate continuous state representations: $\Phi(\mathcal{S}) = \mathbb{R}^d$. UCCRL [10] has been considered, but it suffers from two heavy drawbacks: it does not define any method for computing the optimistic plausible MDP and the respective optimal policy; and it relies on the definition of a discretisation of the state representation space, which is exponential on its dimension, therefore intractable in most case, and in our experimental problem more particularly.

We decided to follow the same idea and compute confidence intervals around the regression $\hat{R}_\tau$. The natural way of computing such confidence intervals would be to use confidence bands. Holm–Bonferroni method [11] consists in defining a band of constant diameter around the learnt function such that the probability of the true function to be outside of this band is controlled to be sufficiently low. Unfortunately, this method does not take into account the variability of confidence in different parts of the space, and this variability is exactly the information we are looking for. Similarly, Scheffé's method [12] studies the contrasts between the variables, and although its uncertainty bound is expressed in function of the state, it is only dependent on its distance to the sampling mean, not on the points density near the point of interest. Both methods are indeed confidence measures for the regression, not for the individual points.

Instead, we propose to use the density of neighbours in $\Xi_\tau$ around the current state to estimate its confidence interval. In order to have a neighbouring definition, one needs a similarity measure $\mathcal{S}(s_1, s_2)$ that equals 1 when $s_1 = s_2$ and tends towards 0 when $s_1$ and $s_2$ get infinitely far from each other. In this article, we use the Gaussian similarity measure relying on the Euclidean distance in the state space $\mathcal{S}$ or its linear representation $\Phi(\mathcal{S})$. Once a similarity measure $\mathcal{S}(s_1, s_2)$ has been chosen, the next step consists in computing the neighbouring weight in a sample set $\Xi_\tau$ around a state $s$. Similarly to UCB, UCRL and UCCRL upper confidence, the confidence interval can be obtained thanks to the neighbouring weight with the following equation:

$$CI_\tau(s) = \kappa \sqrt{\frac{\log(|\Xi_\tau|)}{W_\tau(s)}}, \qquad \text{with} \quad W_\tau(s) = \sum_{\langle \tau, s_j, a_j, s'_j, r_j \rangle \in \Xi_\tau} \mathcal{S}(s, s_j), \tag{2}$$

and where parameter $\kappa$ denotes the *optimism* of the agent.

This confidence interval definition shows several strengths: contrarily to Holm-Bonferroni and Scheffé's methods, it is locally defined, and it works with any regression method computing $\hat{R}_\tau$. But it also has two weaknesses: it relies on two parameters $\mathcal{S}$ and $\kappa$, and it does not take into account the level of agreement between the neighbours.

The estimates $\hat{R}_\tau$ of the rewards can be computed with any regression algorithm, from linear regression to neural nets. In our experiments, we use linear regression with a Tikhonov regularisation and $\lambda = 1$, which, in addition to standard regularisation benefits, enables to find regression parameters before reaching a number of examples equal or higher to the dimension $d$ of $\Phi(\mathcal{S})$. As in UCRL, the current policy is updated when the confidence interval in some encountered state has been divided by 2 since the last update.

Since, in our setting, the optimism principle is only used for the reward confidence interval, we can dissociate the continuous linear parametrisation $\Phi(\mathcal{S})$ used for learning the optimal policy and a simpler discrete representation for estimating $\hat{R}_\tau$. If $\hat{R}_\tau$ is estimated by averaging on this discrete representation, its confidence interval $CI_\tau$ might be computed in the same way as UCB or UCRL.

The possibility to use a different state representation for estimating $P$ and $\tilde{R}_\tau$ is a useful property since it enables to include expert knowledge on the tasks: structure, bounds, or priors on $R_\tau$, which may drastically speed up the learning convergence in practice. In particular, the possibility to use priors is interesting when the task distribution is known or learnt from previously encountered tasks.

## 4 Experiments and results

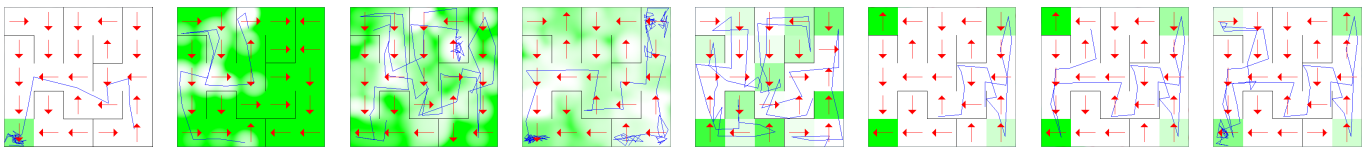
We consider a TRLSD navigation toy problem, where the agent navigates in a 2D maze world as depicted by Figure 1. The state representation $\mathcal{S}$ is the agent's real-valued coordinates $s_t = \{x_t, y_t\} \in (0, 5)^2$, and the set of 25 features $\Phi(s_t)$ is defined with 5*5 Gaussian radial basis functions placed at $s_{ij} = \{i - 0.5, j - 0.5\}$ for $i, j \in [\![1, 5]\!]$.

At each time step, the agent selects an action among four possibilities: $\mathcal{A} = \{\text{NORTH}, \text{WEST}, \text{SOUTH}, \text{EAST}\}$. $P$ is defined as follows for the NORTH action: $x_{t+1} \sim x_t + \mathcal{N}(0, 0.25)$ and $y_{t+1} \sim y_t - 1 + \mathcal{N}(0, 0.5)$, where $\mathcal{N}(\mu, \nu)$ is the Gaussian distribution with centre $\mu$ and standard deviation $\nu$. This works similarly with the other three directions. Then, wall and out-of-grid events intervene in order to respect the dynamics of the maze. When a wall is encountered, a *rebound* is drawn according to $\mathcal{U}(0.1, 0.5)$ is applied, where $\mathcal{U}(\cdot, \cdot)$ denotes the uniform distribution. The stochastic reward function $R_{\tau_{ij}}$ is corrupted with a strong noise and is defined for each task $\tau_{ij}$ with $i, j \in [\![1, 5]\!]$ as follows: $R_{\tau_{ij}}(s_t) \sim 1 + \mathcal{N}(0, 1)$ if $i = \lceil x_t \rceil$ and $j = \lceil y_t \rceil$, and $R_{\tau_{ij}}(s_t) \sim \mathcal{N}(0, 1)$ otherwise.

Transfer Learning experiments unfold as follows. First, 25000 transitions are generated with a random policy and stored in $\Xi$. After each transition, the trajectory has a 2% chance to be terminated, then, the next state is reset to a uniformly drawn state. Those transitions are considered enough to construct a perfect representation of $P$. Whatever the reward function has been used during this data collection, the reward information is discarded, such that any target task $\tau$ would be regarded as new and undiscovered during the transfer phase.

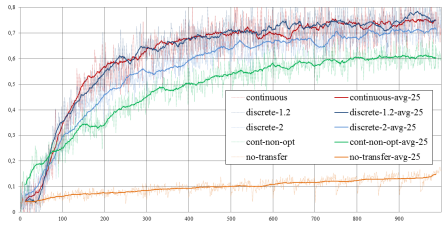
In the first experiment, Task $\tau$ is assumed to be known (*i.e.* the reward function $R_\tau$ is known). It is the case when the agent is instructed to perform a specific task. In this setting, the reward estimator $\hat{R}_\tau$ equals $R_\tau$ and the uncertainty is null. Therefore, $\tilde{R}_\tau = R_\tau$. An optimal policy can directly be computed from $\Psi_{R_\tau}(\Xi)$ with Fitted-$Q$ Iteration. Figure 1a shows that the agent immediately follows a rational policy and heads towards the reward slot. In the other transfer RL experiments, where $R_\tau$ is partially unknown, the transfer learning phase consists in the gathering of 1,000 transitions on the target task.

In the second experiment, 25 tasks are considered. Two transition reuse settings are compared: the *continuous* and the *discrete* settings. In the *continuous* setting, nothing is assumed to be known about the task $\tau$: state representation $\Phi(\mathcal{S})$ is used to learn $R_\tau$. Figures 1b, 1c, and 1d show the exploration-exploitation trade-off of typical trajectories in the *continuous* setting. Initially, the agent is attracted by unknown areas, but as the task is discovered, it exploits more and more its knowledge, and exploration is eventually limited to less visited places when the agents passes by them. In the *discrete* setting, it is assumed to be known that task $\tau$ is defined as follows: $R_\tau : [\![1, 5]\!]^2 \to \mathbb{R}$ and the reward function is cast into this simpler discrete space. Figure 1e shows the first trajectory in the *discrete* setting, exploring methodically the 25 slots.

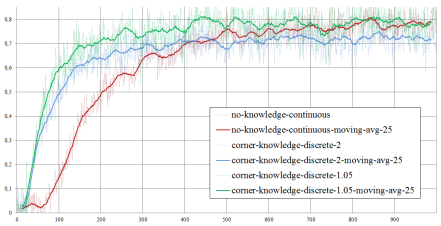


(a) Reward function is known: 20 first transitions. (b) Reward function is unknown: 15 first transitions. (c) Reward function is unknown: after 150 transitions. (d) Reward function is unknown: after 650 transitions. (e) Rewards are known to be defined by cell: 50 first transitions. (f) Rewards are known to be at the corners: 15 first transitions. (g) Rewards are known to be at the corners: 30 first transitions. (h) Rewards are known to be at the corners: 60 first transitions.

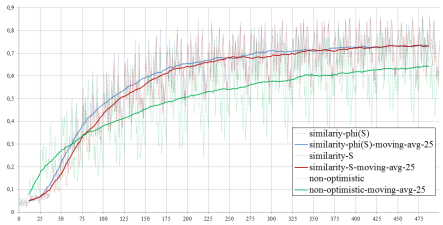
Figure 1: Trajectories directed by policies learnt from 25,000 transition samples with various reward function representations. Some of the policies are motivated by the knowledge of the estimated reward function and some are motivated by curiosity and optimism. The walls are displayed in black and are constant over tasks, the reward function representation are displayed in green, the current trajectory is the broken line in blue and the policy at the centre of cells is coarsely exhibited with the red arrows. In all these screenshots, the real reward function is the one rewarding the bottom left cell.




(a) 25 tasks: average immediate rewards received in function of transition index. Comparison of no-transfer learning and non-optimistic and optimistic in *continuous* and *discrete* settings.




(b) 4 tasks (in corners): average immediate rewards received in function of transition index. Comparison of optimistic in *continuous* setting and in *discrete* setting with injection of strong expert knowledge.




(c) Multi-Task RL: average immediate rewards received in function of trajectory index. Comparison of non-optimistic with two similarity functions: one defined on $\mathcal{S}$, and the other on $\Phi(\mathcal{S})$ in the *continuous* setting.

Figure 2: Performance plots for Transfer RL and Multi-Task RL experiments under *continuous* and *discrete* settings.

The results of the second experiment are displayed on Figure 2a: it compares the collected rewards averaged on 500 independent runs (*i.e.* 20 times each task) with the best parametrisation of *continuous* and *discrete* settings, with the non-optimistic version (*i.e.* with $\kappa = 0$) which corresponds to the method in [6], and with a learning that would not use any transfer. Without transfer, the time scale (only 1000 transitions) is too short to converge to any efficient policy because the dynamics are too complex. The non-optimistic version is better at the very beginning because it exploits right away what it finds, but also regularly fails to converge because of this premature exploitation trend. But the most interesting result that can be observed is the fact that the *continuous* setting dominates the *discrete* one if the updates are made every time a confidence interval is divided by 2. Actually, the *continuous* setting triggers twice more policy updates (80 vs. 40 on average) than the *discrete* setting. By setting the update parameter to 1.2 instead of 2, the number of updates becomes similar as well as the performance curves. The knowledge of the reward function shape does not help: there is the same number of unknown values (25), and our continuous version of confidence bounds demonstrate good properties.

In the third experiment, only corner tasks $\tau_{11}, \tau_{15}, \tau_{51}$, and $\tau_{55}$ are considered. Once again, we compare the *continuous* setting and the *discrete* setting where it is known that rewards are exclusively distributed at the four corners. More formally, task $\tau$ is known to be defined as follows: $R_\tau : \{1, 5\}^2 \to \mathbb{R}$. It corresponds to the case when strong expert knowledge may be injected into the system in order to speed up learning. Figures 1f, 1g, and 1h show that exploration is well targeted at the corners and that unnecessary exploration is avoided. Figure 2b compares both settings and one can notice that initial knowledge boosts significantly early convergence. Once again, one needs to pay attention that less than 10 updates on average are realised after the 1000 fist transitions with the update parameter set to 2, and the performance of online learning gets impaired. The update parameter set to 1.05 reaches 50 updates and a better convergence curve.

A similar experiment has been led in a Multi-Rask RL setting. We invite the readers to refer to [13] for more details. Figure 2c displays the 200-run average performance over time under the *continuous* setting. It compares the non-optimistic version and two different similarity measures. It reveals that both similarity definitions are equally efficient and that they significantly outperform the non-optimistic setting.

Unfortunately, direct comparison with [4] and [5] is difficult for the following reasons: in addition to the shared dynamics, [4] assumes that the reward function parameters are similar among tasks, which is not an assumption that is made in our experiments. Still, one can notice that their experiment dynamics were simpler: no wall and reduced action noise, and that their reward function is deterministic. Despite dealing with a much easier problem, the reward function changed every 5000 transitions in their most complex setting. Our problem of 25 tasks is near solved after 300 episodes of 50 transitions for a total of 15,000 transitions, *i.e.* 600 transition per task. Additionally, they needed 30,000 transitions over 6 tasks to find a good transfer model. It is two times more than our approach on 25 tasks. [5] does not make any assumption of this kind, but their algorithm cannot be directly applied to an online learning problem. As for [4], their experiment is much simpler: deterministic dynamics and reward functions, and despite this, the required sample budget was of 500 transitions per task, when we only need 600 in our very noisy setting.

## References


[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, March 1998.

[2] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

[3] Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*. Springer, 2012.

[4] André Barreto, Rémi Munos, Tom Schaul, and David Silver. Successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1606.05312*, 2016.

[5] Diana Borsa, Thore Graepel, and John Shawe-Taylor. Learning shared representations in multi-task reinforcement learning. *arXiv preprint arXiv:1603.02041*, 2016.

[6] Christopher G. Atkeson and Juan Carlos Santamaria. A comparison of direct and model-based reinforcement learning. In *Proceedings of the 14th International Conference on Robotics and Automation*, pages 3557–3564. IEEE Press, 1997.

[7] Peter Auer and Ronald Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems (NIPS)*, volume 19, page 49, 2007.

[8] Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. In *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016*, pages 2377–2386, 2016.

[9] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

[10] Ronald Ortner and Daniil Ryabko. Online regret bounds for undiscounted continuous reinforcement learning. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1763–1771, 2012.

[11] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[12] Henry Scheffe. *The analysis of variance*, volume 72. John Wiley & Sons, 1999.

[13] Romain Laroche and Merwan Barlier. Proceedings of the 31st conference on artificial intelligence (aaai). 2017.