Off-Policy Learning Combined with Automatic Feature Expansion for Solving Large MDPs

Alborz Geramifard **Christoph Dann** **Jonathan P. How**

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
77 Massachusetts Ave., Cambridge, MA 02139
{agf,cdann,jhow}@mit.edu

Abstract

Reinforcement learning (RL) techniques with cheap computational complexity and minimal hand-tuning that scale to large problems are highly desired among RL practitioners. Linear function approximation has scaled existing RL techniques to large problems [Lagoudakis and Parr, 2003; Silver *et al.*, 2012], however that technique has two major drawbacks: 1) conventional off-policy techniques such as Q-Learning can be unstable when combined with linear function approximation [Baird, 1995] and 2) finding the "right" set of features for approximation can be challenging.

The first drawback has been recently addressed with the introduction of the Greedy-GQ algorithm, a convergent extension of Q-Learning [Maei *et al.*, 2010]. The second drawback led to representation expansion techniques that add new features along the learning process [Geramifard *et al.*, 2011; Keller *et al.*, 2006; Parr *et al.*, 2007]. Amongst these techniques, incremental Feature Dependency Discovery (iFDD) has shown great potential as it scaled to large problems while enjoying convergence results [Geramifard *et al.*, 2011]. Recently, iFDD$^+$ [Geramifard *et al.*, 2013b] improved the performance of iFDD in the prediction problem while outperforming the previous state-of-the-art batch expansion technique OMP-TD [Painter-Wakefield and Parr, 2012].

This paper connects Greedy-GQ learning with iFDD$^+$ and, for the first time, introduces an online off-policy learning with automatic feature expansion technique. Given sparse features, the new algorithm has per-time-step complexity independent of the total number of features, while for most existing techniques feature discovery is at least quadratic in the number features [Keller *et al.*, 2006; Parr *et al.*, 2007]. Empirical results across 3 domains with sizes up to 77 billion state-action pairs verify the scalability of our new approach.

Keywords: temporal difference learning, representation expansion, off-policy, scaling learning

Acknowledgements

We would like to thank Hamid Maei for his insightful feedback on the use of Greedy-GQ. This research was sponsored by ONR grants N00014-07-1-0749 and N00014-11-1-0688.

1 Introduction

The RL community made a big leap by using linear function approximators to tackle large problems [Lagoudakis and Parr, 2003; Silver *et al.*, 2012]. While using small number of features is much more scalable compared to working with large number of states, it renders off-policy techniques such as Q-Learning unstable [Baird, 1995] and demands careful crafting of the "right" set of features. Off-policy techniques are important part of RL family, as they allow the policy being learnt to be different from the policy used for sampling. To address the latter, Maei *et al.* [2010] recently introduced the Greedy-GQ algorithm as a convergent off-policy technique that employs linear function approximation. To automate the feature crafting process, several representation expansion techniques were developed [Geramifard *et al.*, 2011; Keller *et al.*, 2006; Parr *et al.*, 2007]. Among these methods, the incremental Feature Dependency Discovery (iFDD) [Geramifard *et al.*, 2011] algorithm has shown promise because it successfully scaled to problems with 150 million state-action pairs and has low computational complexity. Recent theoretical analysis on iFDD led to the iFDD$^+$ algorithm that outperformed the previous state-of-the art feature expansion technique in the prediction problem [Geramifard *et al.*, 2013b].

This paper combines Greedy-GQ and iFDD$^+$ algorithms, introducing the first online off-policy technique with automatic feature expansion. Given sparse features, Greedy-GQ-iFDD$^+$ has per-time-step complexity independent of the number of features. Empirical results in domains with up to 77 billion state-action pairs verifies the scalability of our new method.

2 Preliminaries

Markov Decision Processes (MDPs) [Sutton and Barto, 1998] are defined as a tuple $(\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\mathcal{S}_0,\gamma)$ with $\mathcal{S}$ being the set of all states and $\mathcal{A}$ the finite set of all actions. The transition model $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ specifies the probability of moving from state s to state s' following action a. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the corresponding reward value. The initial state distribution is given by $\mathcal{S}_0 : \mathcal{S} \to [0,1]$. $\gamma \in [0,1[$ is a discount factor. A policy $\pi : \mathcal{S} \to \mathcal{A}$ maps each state to an action. A trajectory is a sequence of $s_0, a_0, r_0, s_1, a_1, r_1, s_2 \ldots$, where $s_0 \sim \mathcal{S}_0$, $a_t = \pi(s_t)$ and $s_{t+1} \sim \mathcal{P}(s_t,a_t,\cdot)$ for $t > 0$. The value of a state-action pair under policy π is defined as the expected cumulative discounted reward (*i.e.*, return) the agent will receive when it takes action a at state s and follows policy π thereafter:

$$Q^\pi(s,a) = \mathbb{E}_{\pi,\mathcal{P}}\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, a_0 = a\right]. \tag{1}$$

An optimal policy π^* maximizes the state-action values, satisfying the Bellman equation:

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^{\pi^*}(s,a) \quad \forall s \in \mathcal{S}$$

Consequently, calculating Q^{π^*} will be sufficient to find the optimal policy. However this approach requires the storage of $|\mathcal{A}||\mathcal{S}|$ parameters, not amenable for MDPs with large or infinite number of states. Linear function approximation elevates this problem by representing Q^π in a lower dimensional space: $Q_\theta = \theta^T \phi(s,a) \approx Q^\pi$ where $\theta \in \mathbb{R}^n$ is a parameter vector and $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^n$ is a feature function associating an n-dimensional feature-vector to each state-action pair. For the rest of the paper, we assume $\phi(s,a)$ is formed in two steps. First ϕ maps state s to $\mathbb{R}^m$. Then the resulting vector is copied to the ath slot of a zero vector with $|\mathcal{A}|m = n$ elements, where $|\mathcal{A}|$ is small.

Temporal Difference (TD) learning algorithms improve Q_θ based on the current estimates of Q_θ using sampling techniques. By taking r_0 out of the sum in Equation 1, it can be verified that the value function satisfies

$$Q^\pi(s_t,a_t) = T^\pi Q^\pi(s_t,a_t) \triangleq \mathbb{E}_{\mathcal{P}}[r_t] + \gamma \mathbb{E}_{\mathcal{P},\pi}\left[Q^\pi(s_{t+1},a_{t+1})\right].$$

Hence, the value function is a fixed-point of the *Bellman operator* T^π. The SARSA algorithm [Rummery and Niranjan, 1994] updates the parameters θ_t of the value function estimate at time-step t by $\theta_{t+1} = \theta_t + \alpha_t \delta_t \phi_t$, where α_t is a step-size decreasing over time and δ_t is the TD error of the current transition:

$$\delta_t = r_t + \gamma Q_{\theta_t}(s_{t+1},a_{t+1}) - Q_{\theta_t}(s_t,a_t). \tag{2}$$

The policy π incorporates exploration using the ϵ-greedy policy, where on each step the agent takes a random action with probability ϵ and acts greedy based on the current Q_θ with probability $1 - \epsilon$. Under mild conditions with decaying ϵ, SARSA converges to the optimal policy π^* [Sutton and Barto, 1998]. SARSA is an on-policy technique, because in the TD error calculation (*i.e.*, Equation 2), a_{t+1} is selected according to the current policy π that involves the randomness. Hence, if the random function selects a poor a_{t+1} for which $Q(s_{t+1},a_{t+1})$ has a low value, $Q(s_t,a_t)$ will be panelized accordingly.

Off-policy techniques address this problem by allowing the agent to learn the value function of policy π while collecting samples using a different policy π'. For example Q-Learning [Watkins and Dayan, 1992] can learn about the greedy policy, while collecting samples using an ϵ-greedy policy. Q-Learning updates θ identical to SARSA, yet the TD error is calculated as $\delta_t = r_t + \gamma \operatorname{argmax}_a Q_{\theta_t}(s_{t+1},a) - Q_{\theta_t}(s_t,a_t)$. Unfortunately Q-Learning has been shown to be unstable

with linear function approximation [Baird, 1995]. Recently the Greedy-GQ algorithm [Maei *et al.*, 2010] addressed this drawback by maintaining a second vector ω_t that helps tracking the sub-gradient of the projected Bellman error. The updates for both θ and ω estimates are given by

$$a' = \text{argmax}_a Q_\theta(s_{t+1},a), \tag{3}$$

$$\theta_{t+1} = \theta_t + \alpha_t \left[\delta_t \phi_t - \gamma(\omega_t^\top \phi(s_t,a_t))\phi(s_{t+1},a')\right], \tag{4}$$

$$\omega_{t+1} = \omega_t + \beta_t \left[\delta_t - (\omega_t^\top \phi(s_t,a_t))\right] \phi(s_t,a_t), \tag{5}$$

where δ_t is computed as in Q-Learning and β_t is the learning rate for ω_t. Notice that by setting $\beta_t = 0$ and $\omega_0 = \bar{0}$ we retrieve the Q-Learning algorithm.

Incremental Feature Dependency Discovery (iFDD) [Geramifard *et al.*, 2011] is a recent algorithm for expanding the set of features used for linear function approximation. iFDD assumes the existence of an initial set of binary features ($\boldsymbol{F}$), where $\phi_f : \mathcal{S} \rightarrow \{0,1\}, \forall f \in \boldsymbol{F}$. Through the learning process, iFDD identifies potential features as the conjunction of existing features. For example if $\boldsymbol{F}$ contains 20 features, then a potential feature f can be the conjunction of features 10 and 17, that is $\phi_f(s) = 1 \iff (\phi_{10}(s) = 1 \wedge \phi_{17}(s) = 1)$. Potential features are added as new features once their relevance, η_t, reaches a certain threshold. In the original iFDD algorithm [Geramifard *et al.*, 2011] the relevance of potential feature f at time t is calculated as the cumulative absolute TD error $\eta_t(f) = \sum_{i=0,\phi_f(s_i)=1}^{t} |\delta_i|$. Recently a better relevance criteria was introduced [Geramifard *et al.*, 2013b] based on the rate of convergence analysis of iFDD and its relation to orthogonal matching pursuit algorithms in the prediction case. The new algorithm named, iFDD$^+$, calculates the relevances as

$$\eta_t(f) = \frac{\left|\sum_{i=0,\phi_f(s_i)=1}^{t} \delta_i\right|}{\sqrt{\sum_{i=0,\phi_f(s_i)=1}^{t} 1}}. \tag{6}$$

3 Off-policy Learning with Automatic Feature Expansion

This section combines the Greedy-GQ algorithm with iFDD$^+$ and introduces a novel online off-policy algorithm with feature expansion capability. Algorithm 1 shows the main process. α_t and β_t are the two step size learning parameters (Equations 4-5). ϵ is the exploration used in the ϵ-greedy policy. ξ is the discovery threshold controlling feature expansion. $\boldsymbol{F}$ is the initial set of binary features. ψ and N are two hash maps calculating the sum of TD errors and number of times a feature visited (numerator and denominator of Equation 6). The algorithm follows an ϵ-greedy policy and performs Greedy-GQ updates (Equations 3-5), except for lines 11 and 12. Line 11 executes the *Discover* function defined in Algorithm 2 to expand features. Notice that instead of $\phi(s,a)$, $\phi(s)$ is passed for feature discovery, as all features are shared across all actions. Line 12 pads θ and ω vectors in case of feature expansion. In Algorithm 2, changes to the original iFDD algorithm [Geramifard *et al.*, 2011] to reflect iFDD$^+$ (*i.e.*, Equation 6) are highlighted with the yellow background. All ϕ functions call Algorithm 3 to incorporate new expanded features. Note that, we fixed a drawback of the earlier version of this function [Geramifard *et al.*, 2011] (shown as yellow) which could have led to activation of unnecessary features.

3.1 Per-Time-Step Complexity

Define k_t as the maximum number of non-zero elements for all feature vectors at time t. It can be verified that the per-time-step complexity of Greedy-GQ is $\mathcal{O}(|\mathcal{A}|k_t) = \mathcal{O}(k_t)$. The transition from iFDD to iFDD$^+$ does not raise the per-time-step complexity of the algorithm which is $\mathcal{O}(k_t 2^{k_t})$ [Geramifard *et al.*, 2011]. Hence Greedy-GQ-iFDD$^+$has per-time-step complexity of $\mathcal{O}(k_t 2^{k_t})$. Furthermore, it has been shown that using iFDD, $\forall i > j \rightarrow k_i \leq k_j$ [Geramifard *et al.*, 2011], meaning as new features are discovered the sparsity increases resulting in faster calculations. Due to the exponential term introduced in the complexity of iFDD, the resulting algorithm will be applicable when sparse features are used (*i.e.*, $k_0 \ll m$). Section 4 shows that this condition can be met even in very large domains.

4 Experimental Results

This section investigates the performance of Greedy-GQ (shown as GQ), Q-Learning (*i.e.*, GQ with $\beta_t = 0$), and SARSA with feature expansion (*i.e.*, iFDD$^+$) and without feature expansion (*i.e.*, fixed sparse representation, FSR) across three MDPs: Inverted Pendulum Balancing, BlocksWorld, and Persistent Search and Track. Results are generated using the RLPy framework which is available online [Geramifard *et al.*, 2013a]. For each method 30 runs were used in which after each tenth part of the experiment, a single return of the algorithm was sampled with no exploration. All methods used the same set of random seeds. α_t took the form

$$\alpha_t = \frac{\alpha_0}{k_t} \frac{N_0 + 1}{N_0 + \text{Episode\#}^{1.1}},$$

Algorithm 1: Greedy GQ-iFDD$^+$

Input: $\alpha_t,\beta_t,\epsilon,\xi,\boldsymbol{F}$
Output: Q_θ

Initialize ψ,N to empty maps.
while *time permits* **do**
  initialize s from $\mathcal{S}_0$
  repeat
    $a \leftarrow \epsilon$-greedy w.r.t Q_θ
    $s',r \leftarrow$ execute a
    $a' \leftarrow \text{argmax}_{a'}\, Q_\theta(s',a')$
    $\delta \leftarrow r + Q_\theta(s',a') - Q_\theta(s,a)$
    $\theta \leftarrow \theta + \alpha_t \left[\delta\phi(s,a) - \gamma\left(\omega^\top\phi(s,a)\right)\phi(s',a')\right]$
    $\omega \leftarrow \omega + \beta_t \left[\delta - \phi(s,a)^\top\omega\right]\phi(s,a)$
    $\phi \leftarrow$ Discover$(\phi(s),\delta,\xi,\boldsymbol{F},\psi,N)$
    Pad ω and θ if new features are added.
    $s \leftarrow s'$
  until *s is terminal*;

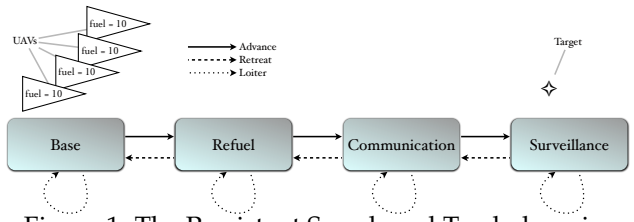


Figure 1: The Persistent Search and Track domain.

Algorithm 2: Discover

Input: $\phi(s),\delta,\xi,\boldsymbol{F},\psi,N$
Output: $\boldsymbol{F},\psi$

foreach $(g,h) \in \{(i,j)|\phi_i(s)\phi_j(s) = 1\}$ **do**
  $f \leftarrow g \wedge h$
  if $f \notin \boldsymbol{F}$ **then**
    $\psi_f \leftarrow \psi_f + \delta$
    $N_f \leftarrow N_f + 1$
    if $|\psi_f|/\sqrt{N_f} > \xi$ **then**
      $\boldsymbol{F} \leftarrow \boldsymbol{F} \cup f$

Algorithm 3: Generate Feature Vector (ϕ)

Input: $\phi^0(s),\boldsymbol{F}$
Output: $\phi(s)$

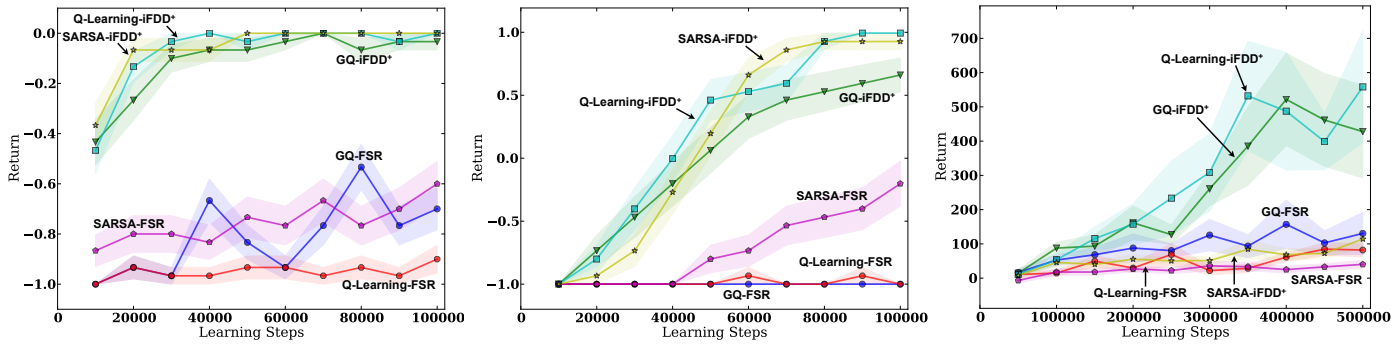
$\phi(s) \leftarrow \vec{0}$
activeInitialFeatures $\leftarrow \{i|\phi_i^0(s) = 1\}$
Candidates $\leftarrow$ SortedPowerSet(*activeInitialFeatures*)
while *activeInitialFeatures* $\neq \emptyset$ **do**
  $f \leftarrow$ *Candidates*.next()
  if $f \subset$ *activeInitialFeatures* **and** $f \in \boldsymbol{F}$ **then**
    activeInitialFeatures $\leftarrow$ *activeInitialFeatures* $\setminus f$
    $\phi_f(s) \leftarrow 1$
return $\phi(s)$

where the best parameters α_0 and N_0 were empirically found from $\{0.1,1\}$ and $\{100,1000,10^6\}$ for each algorithm and domain. Greedy-GQ used $\beta_t = 10^{-6}\alpha_t$ and we set $\epsilon = 0.1$.

Inverted Pendulum Balancing is a widely used benchmark in the control and reinforcement learning community, where a pole mounted on a cart has to be balanced upright by applying force on the cart. We followed the setting of Lagoudakis and Parr [2003] where angle and angular velocity of the pole define the state. The agent can either apply no force or push the cart with $50N$ to the right or left. The pendulum is initialized vertically and the episode stops as soon as the pole reaches the horizontal position with reward -1. All other steps have reward 0. The pendulum is disturbed by uniform noise on the actions between $\pm 10N$. We set $\gamma = 0.95$. Episodes were capped at 3,000 steps. Each state dimension was discretized into 20 bins resulting in 40 initial features per action and k_0 to be 2. This MDP has 1,200 discretized state-action pairs. ξ was empirically found from the set $\{0.1,0.2,0.5\}$.

BlocksWorld is a classical planning problem with the goal of stacking all blocks with a predefined order. We used 6 blocks all initially on the table [Geramifard *et al.*, 2011]. The noise for each movement was 30% resulting in dropping the moving block on the table. The reward is $+1$ for a successful tower build and -0.001 for every other step. The state was defined by a 6 dimensional vector where $s_i = j$ corresponds to block i being on top of block j. For compactness, we interpreted $s_i = i$ for block i being on the table. This MDP has about 1.6×10^6 state-action pairs. Initial features were generated by indicator functions for the position of each block, amounting to 36 features per action and $k_0 = 6$. We set $\gamma = 1$, episodes were capped at 1,000 steps, and ξ was empirically found from the set $\{0.02,0.05,0.1\}$.

Persistent Search and Track (PST) is an MDP with the task of surveilling a target using unmanned aerial vehicles (UAVs) as described in [Geramifard *et al.*, 2011] and shown in Figure 1-(c). Each UAV has three actions $\{retreat,loiter,advance\}$ moving it along the graph. The state of each UAV is described by its $\{fuel,location,motor,camera\}$, where fuel is a discrete value between 0 and 10. The location is the node on the graph. The motor and camera are binary variables indicating the functionality of the corresponding equipment. An additional UAV was added to the previous instantiation of the domain [Geramifard *et al.*, 2011], raising the size of the state-action space to more than 77 billion state-action pairs. The movement of UAVs are deterministic. The motor and the camera of each UAV have 5% chance of failure on each step. The reward of $+20$ is collected when a UAV with a working camera is at the surveillance node and a UAV with a working motor hovers at the communication node. UAVs lose one fuel cell per action and gain full fuel at the refuel node. Both the motor and the camera of a UAV are fixed upon its arrival at base. A scenario is terminated if a UAV crashes due to fuel depletion with a reward of -50. Initial features were generated using indicator functions for each



(a) Inverted Pendulum Balancing (b) Blocksworld (c) Persistent Search and Track

Figure 2: Learning performance in three domains: results are averaged over 30 runs. Shaded areas indicate the standard error of the sampled mean.

dimension of each UAV ignoring zero values for binary dimensions amounting to $17 \times 4 = 68$ features and $k_0 = 16$. We set $\gamma = 0.9$, episodes were capped at 1,000 steps, and ξ was empirically found from the set $\{75,100,150\}$.

Discussion of Results Figure 2 depicts the simulation results. The Y-axis is the average return of each technique, and the X-axis shows the number of interactions. Shaded areas highlight the standard error of the mean. For the first two problems, the number of interactions were capped to 10^5 steps. For the last domain, due to the large size of the problem, this number was increased to 5×10^5. In the pendulum domain (Figure 2-(a)), the feature expansion ability enabled all learning techniques to perform close to optimal after 30,000 steps, while no agent with the fixed representation could reliably balance the pendulum by the end of the learning horizon. The same trend can be observed in the BlocksWorld domain (Figure 2-(b)), although one can observe that SARSA-FSR learns better policies compared to off-policy techniques using FSR. Furthermore, among agents using iFDD$^+$, the gap between GQ and Q-Learning has widened. In the largest problem (Figure 2-(c)), among agents using iFDD$^+$, GQ and Q-Learning performed well, while SARSA performed poorly. We suspect this phenomenon is due to higher noise in on-policy TD errors caused by random actions. Off-policy techniques discard the random actions during TD error calculation, providing better criteria for feature expansion. This effect is visible in this domain, because the consequence of random actions can be much more substantial. Agents using FSR performed similar to SARSA-iFDD$^+$ resulting in poor policies. Notice that compared to iFDD, table-lookup representations have been shown to lead to drastic increase in the sample complexity for all three domains [Geramifard *et al.*, 2011].

5 Conclusion

We combined Greedy-GQ algorithm with iFDD$^+$ and introduced the first online off-policy control algorithm with the ability to expand the representation. As shown in Section 3.1, given sparse features, the new algorithm has a per-time-step complexity independent of the total number of features. Empirical results across three domains with sizes up to 77 billion state-action pairs verified the great potential of using off-policy learning with automated feature expansion.

References

Leemon Baird. Residual Algorithms : Reinforcement Learning with Function Approximation. In *International Conference on Machine Learning*, 1995.

Alborz Geramifard, Finale Doshi, Joshua Redding, Nicholas Roy, and Jonathan How. Online discovery of feature dependencies. In Lise Getoor and Tobias Scheffer, editors, *International Conference on Machine Learning (ICML)*, pages 881–888. ACM, June 2011.

Alborz Geramifard, Robert H Klein, and Jonathan P How. RLPy: The Reinforcement Learning Library for Education and Research. http://acl.mit.edu/RLPy, April 2013.

Alborz Geramifard, Thomas J. Walsh, Nicholas Roy, and Jonathan How. Batch iFDD: A Scalable Matching Pursuit Algorithm for Solving MDPs. In *Proceedings of the 29th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, Bellevue, Washington, USA, 2013. AUAI Press.

Philipp W. Keller, Shie Mannor, and Doina Precup. Automatic Basis Function Construction for Approximate Dynamic Programming and Reinforcement Learning. In *International Conference on Machine Learning*, 2006.

Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research (JMLR)*, 4:1107–1149, 2003.

Hamid Reza Maei, Csaba Szepesvári, Shalabh Bhatnagar, and Richard S. Sutton. Toward off-policy learning control with function approximation. In Johannes Fürnkranz and Thorsten Joachims, editors, *International Conference on Machine Learning (ICML)*, pages 719–726. Omnipress, 2010.

Christopher Painter-Wakefield and Ronald Parr. Greedy Algorithms for Sparse Reinforcement Learning. In *International Conference on Machine Learning*, 2012.

Ronald Parr, Christopher Painter-Wakefield, Lihong Li, and Michael Littman. Analyzing Feature Generation for Value-Function Approximation. In *International Conference on Machine Learning*, 2007.

G. A. Rummery and M. Niranjan. Online Q-learning using connectionist systems (tech. rep. no. cued/f-infeng/tr 166). *Cambridge University Engineering Department*, 1994.

David Silver, Richard S. Sutton, and Martin Müller. Temporal-difference search in computer go. *Machine Learning*, 87(2):183–219, 2012.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.