massachusetts institute of technology — artificial intelligence laboratory

Exploration in Gradient-Based Reinforcement Learning

Nicolas Meuleau, Leonid Peshkin and Kee-Eung Kim

AI Memo 2001-003 April 3, 2001

Abstract

Gradient-based policy search is an alternative to value-function-based methods for reinforcement learning in non-Markovian domains. One apparent drawback of policy search is its requirement that all actions be "on-policy"; that is, that there be no explicit exploration. In this paper, we provide a method for using importance sampling to allow any well-behaved directed exploration policy during learning. We show both theoretically and experimentally that using this method can achieve dramatic performance improvements.

During this work, Nicolas Meuleau was at the MIT Artificial Intelligence laboratory, supported in part by a research grant from NTT; Leonid Peshkin by grants from NSF and NTT; and Kee-Eung Kim in part by AFOSR/RLF 30602-95-1-0020.

1 Introduction

Reinforcement learning (Sutton & Barto, 1998) provides a framework for solving and learning to solve large combinatorial decision problems such as Markov decision processes (MDPs) (Puterman, 1994) and partially observable MDPs (POMDPs) (Kaelbling et al., 1998). There are two main approaches to solving reinforcement learning problems. On one side, *value search* algorithms such as Q-learning (QL) and SARSA(λ) (Sutton & Barto, 1998) find the optimal policy by first searching for the optimal *value function*, and then deducing the optimal policy from the optimal value function. Look-up table implementations of these algorithms can be showed to converge to a *global* optimum of the expected reward (Watkins & Dayan, 1992; Singh et al., 2000). However, they work only in completely observable (Markovian) environments. Although some attempts to use value search in partially observable settings have been made (Littman, 1994; Wiering & Schmidhuber, 1997), none of these techniques is guaranteed to find an optimal solution. This is simply because Bellman's equation does not transfer to non-Markovian environments. More precisely, we cannot rewrite Bellman's fundamental equation by replacing states with observations in a partially observable environment. The use of (Bayesian) belief-states instead of the original states enables a value-function approach, but it increases the complexity of the problem dramatically (Kaelbling et al., 1998).

On the other side, *policy search* algorithms such as REINFORCE (Williams, 1992) work directly in the policy space, trying to maximize the expected reward without the help of value functions. Most policy search algorithms are based on approximating gradient descent in some way (*e.g.*, (Williams, 1992; Baxter & Bartlett, 2000; Marbach & Tsitsiklis, 2001)). Therefore, they typically find only local optima of the expected reward. Moreover, it is often believed that value search is faster: in some sense, Bellman's optimality principle is a powerful heuristic to guide the search, and policy search algorithms are less informed than value search algorithms. However, gradient-based policy search accommodates partial observability and non-Markovianism very well (Baird & Moore, 1999; Baxter & Bartlett, 2000). It can be used to find (locally) optimal controllers under many kinds of constraints, with many different forms of memory (Meuleau et al., 1999; Peshkin et al., 1999; Kim et al., 2000), including in partially observable multi-agent settings (Peshkin et al., 2000; Bartlett & Baxter, 2000).

The basic gradient-based policy search algorithm is REINFORCE (Williams, 1992). We consider a more general definition of REINFORCE independently of the neural network used to encode the policy. It performs stochastic gradient descent of the expected reward. REINFORCE is basically an *on-policy* algorithm (Sutton & Barto, 1998): the gradient at a given point in the policy space is estimated by following precisely this policy during learning trials. It corresponds to a naive way of estimating expectation by sampling. A more sophisticated way is offered by the Monte Carlo technique known as *importance sampling* (Rubinstein, 1981). Applied to REINFORCE, it allows *off-policy* implementations of the algorithm, that is, we may execute a policy

different from the current policy during the learning trials.

Off-policy implementations are interesting in many respects. First, we may not have a choice. For instance, we may not be able to execute *any* policy during learning trials, or we may have to learn from a given training set of trajectories. Second, off-policy algorithms may be used to optimize several policies or controllers at the same time. In Sutton et al.'s intra-option learning, an off-policy (value search) algorithm is used to train several policies (or options) simultaneously (Sutton et al., 1998). In the case of gradient-based policy search, *off-policyness* allows simultaneous optimization of several controllers, possibly with different architecture, which could lead to evolving controller architecture. Another advantage of off-policy algorithms is that they can re-use past experience at any step of learning: the trajectories sampled in the beginning of learning may be used later on, even if the policy changed in between. (Kearns et al., 1999) proposed an algorithm based on this idea for POMDPs. Finally, off-policy implementations may allow for dramatic reduction in the (sample) complexity of learning. In the following, we focus particularly on the latter point.

In this paper, we propose off-policy implementations of REINFORCE based on importance sampling. We show that they can be used to reduce the complexity of learning, which is measured as a function of two parameters: the number of learning trials required to reach a given performance, and the length of learning trials. We present simulation results where our off-policy algorithms reduce both, which results in a dramatic reduction of the total number of time-steps of interaction required to reach a given performance level. This work bears many similarities with certain aspects of Sutton and Barto's book (1998), and with (Precup et al., 2000). However, it concerns gradient-based policy search, while previous work focus on value search approaches. To save space, we suppose that the reader is familiar with basic notions of Markov decision processes (MDPs) and reinforcement learning. A brief introduction to REINFORCE is given in section 2. Throughout the paper, we focus on fully observable MDPs, which are solved by REINFORCE implementing a reactive policy. However, the technique of importance sampling can be used in any variation of REINFORCE, including in partially observable environments, and in other gradient-based algorithm as well. The end of the paper presents simulation results obtained in a partially observable environment.

2 Gradient-Based Policy Search

To simplify the presentation, we suppose that the environment is a finite MDP and that REINFORCE is used to optimize the parameters (action probabilities) of a stochastic memoryless policy, which is sufficient in such an environment (see (Meuleau et al., 1999) for more complex setting). We also suppose that the problem is a goal-achievement task, *i.e.*, there is an absorbing goal state that must be reached as fast as possible. We will insure that all policies are proper, *i.e.*, that we finally reach the goal with probability 1 under any policy, by preventing the action probabilities from

going to 0. Typically, the policy will be stored as set of weights, $w_{sa} \in I\!R$, decoded using Boltzmann's law:

$$\mu(s,a) \stackrel{\text{def}}{=} \Pr(a_t = a \mid s_t = s) = \frac{e^{w_{sa}/\theta}}{\sum_{a'} e^{w_{sa'}/\theta}} > 0 \ ,$$

where s_t and a_t are the state and action at time t, and θ is a constant temperature parameter.

(Episodic) REINFORCE was introduced by Williams (Williams, 1992). It performs stochastic gradient ascent of the expected cumulative reward $V_\mu = \mathrm{E}_\mu\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where γ is the discount factor, r_t is the reward at time t and μ is the current policy. The less sophisticated implementation of the algorithm is based on the equality

$$V_\mu = \sum_{T=1}^{\infty} \sum_{\bar{h}_T \in \bar{H}_T} \Pr\left(\bar{h}_T \mid \mu\right) \sum_{t=0}^{T-1} R^t\left(\bar{h}_T\right) \ ,$$

where $\bar{h}_T = (s_0, a_0, r_0, \ldots s_{T-1}, a_{T-1}, r_{T-1}, s_T)$ is an *experience sequence* of length T (s_0, s_1, $\ldots$ s_{T-1} are all different from the goal state, s_T is the goal state), $\bar{H}_T$ is the set of all such sequences, and $R^t(\bar{h}_T)$ is the reward at time t in $\bar{h}_T$. Then

$$\frac{\partial}{\partial w_{sa}} V_\mu = \sum_{T=1}^{\infty} \sum_{\bar{h}_T \in \bar{H}_T} \Pr\left(\bar{h}_T \mid \mu\right) \bar{C}_{sa}\left(\bar{h}_T\right) \quad = \mathrm{E}_\mu\left[\bar{C}_{sa}\left(\bar{h}_T\right)\right] \ ,$$

where $\bar{C}_{sa}(\bar{h}_T)$ is the contribution of $\bar{h}_T$ to the partial derivative of V_μ with respect to w_{sa}:

$$\bar{C}_{sa}(\bar{h}_T) \quad \stackrel{\text{def}}{=} \quad \sum_{t=0}^{T-1} \gamma^t r_t \sum_{t'=0}^{T-1} \frac{\partial}{\partial w_{sa}} \ln(\mu(s_{t'}, a_{t'})) \ .$$

If experience sequences are sampled following $\Pr(\cdot \mid \mu)$, that is, if the current policy is followed during learning trials, then the contribution $\bar{C}_{sa}$ of an experience trial $\bar{h}_T$ is an unbiased estimate of the gradient that may be used to make a step in the policy-space. The contribution is generally easy to calculate. For instance, if the policy is stored using look-up tables and Boltzmann's law, then we have:

$$\frac{\partial}{\partial w_{sa}} \ln(\mu(s',a')) \quad = \begin{cases} (1 - \mu(s',a'))/\theta & \text{if } s = s' \text{ and } a = a' \ , \\ -\mu(s',a')/\theta & \text{if } s = s' \text{ and } a \neq a' \ , \\ 0 & \text{otherwise .} \end{cases}$$

A simple algorithm using this policy architecture is presented in fig. 1.

As Williams stressed (Williams, 1992), this brute implementation of REINFORCE does not perform optimal temporal credit assignment, since it ignores the fact that the reward at time t does not depend on the actions performed after time t. A more

1. **Initialize** the controller weights w_{sa};
2. **Beginning of a trial**:
   - for all (s,a): $N_s \leftarrow 0$, $N_{sa} \leftarrow 0$;
   - $R \leftarrow 0$;
3. **At each time-step t of the trial**:
   - draw an action a_t at random following $\mu(s_t, \cdot)$;
   - $N_{s_t} \leftarrow N_{s_t} + 1$, $N_{s_t a_t} \leftarrow N_{s_t a_t} + 1$;
   - execute an action a_t, receive s_t, r_t from environment;
   - $R \leftarrow R + \gamma^t r_t$;
4. **End of the trial**:
   - for all (s,a): $w_{sa} \leftarrow w_{sa} + \alpha R(N_{sa} - \mu(s,a)N_s)/\theta$;
5. **Loop**: return to 2.

Figure 1: Algorithm 1: a simple implementation of REINFORCE using look-up tables and the Boltzmann's law. α is the step-size parameter, or learning rate.

efficient implementation that takes into account the *causality* of rewards can be obtained by rewriting the expected reward in the form

$$V_\mu = \sum_{t=1}^{\infty} \sum_{h_t \in H_t} \Pr(h_t \mid \mu) R^t(h_t) \,,$$

where $h_t = (s_0, a_0, r_0, \ldots s_{t-1}, a_{t-1}, r_{t-1}, s_t)$ is an *experience prefix* of length t (s_t may be different from the goal state), and H_t is the set of all such experience prefixes ($\bar{H}_t \subset H_t$). The gradient of V_μ may then be expressed in the form

$$\frac{\partial}{\partial w_{sa}} V_\mu = \sum_{t=1}^{\infty} \sum_{h_t \in H_t} \Pr(h_t \mid \mu) C_{sa}(h_t) = \sum_{t=1}^{\infty} \mathrm{E}_\mu \left[X^t_{Csa} \right] ,$$

where $C_{sa}(s_0, a_0, r_0, \ldots s_{t-1}, a_{t-1}, r_{t-1}, s_t)$ is the contribution of the prefix h_t

$$C_{sa}(h_t) \quad \stackrel{\text{def}}{=} \quad \gamma^{t-1} r_{t-1} \sum_{t'=0}^{t-1} \frac{\partial}{\partial w_{sa}} \ln(\mu(s_{t'}, a_{t'})),$$

and X^t_{Csa} is the random variable that takes the value $C_{sa}(h_t)$ if an experience trial ends after or at time t, and 0 otherwise. During an experience trial following the current policy μ, we calculate an unbiased estimate of $E_\mu \left[X^t_{Csa} \right]$ *for each* $t \in I\!N$. The sum of these estimates—which is an unbiased estimate of the gradient—is used to update the policy at the end of the trial. The different estimates are not mutually independent, but it does not hurt since we are only summing them. Figure 2 presents a look-up table implementation of this algorithm. Note that algorithms 1 and 2 are exactly equivalent if we use the goal-reward model, that is, if the reward is always 0 except at the goal. Algorithm 2 has been later generalized by Baird and Moore so that it can mix value and policy search (Baird & Moore, 1999).

1. **Initialize** the controller weights w_{sa};
2. **Beginning of a trial**:
   - for all (s,a): $N_s \leftarrow 0$, $N_{sa} \leftarrow 0$, $\Delta w_{sa} \leftarrow 0$;
3. **At each time-step t of the trial**:
   - draw an action a_t at random following $\mu(s_t,\cdot)$;
   - $N_{s_t} \leftarrow N_{s_t} + 1$, $N_{s_t a_t} \leftarrow N_{s_t a_t} + 1$;
   - execute an action a_t, receive s_t, r_t from environment;
   - for all (s,a): $\Delta w_{sa} \leftarrow \Delta w_{sa} + \gamma^t r_t (N_{sa} - \mu(s,a) N_s)/\theta$;
4. **End of the trial**:
   - for all (s,a): $w_{sa} \leftarrow w_{sa} + \alpha \Delta w_{sa}$;
5. **Loop**: return to 2.

Figure 2: Algorithm 2: a look-up table implementation of REINFORCE that takes into account the causality of rewards.

3 Off-Policy Implementations of REINFORCE

REINFORCE is basically an on-policy algorithm (Sutton & Barto, 1998): the gradient at a given point in the policy space is estimated by acting following precisely this policy during learning trials. Conversely, off-policy algorithms are able to improve a given policy while executing a different one during interactions with the environment. In this section, we present off-policy implementations of REINFORCE.

3.1 Simple Importance Sampling

The key to off-policy implementations of REINFORCE is the Monte Carlo technique known as *importance sampling* (IS) (Rubinstein, 1981). REINFORCE uses *naive sampling* because it estimates the expectations of $\bar{C}_{sa}$ and X^t_{Csa} conditionally on the current policy μ, by sampling experience sequences and prefixes following the current policy μ. A more sophisticated solution consists of defining another policy μ' such that $\Pr(h_t \mid \mu) > 0 \implies \Pr(h_t \mid \mu') > 0$ for all h_t and all t, executing μ' instead of μ during learning trials, and multiplying the contribution $\bar{C}_{sa}$ or C_{sa} of each experience sequence or prefix $h_t = (s_o, a_0, r_0, \ldots s_{t-1}, a_{t-1}, r_{t-1}, s_t)$ by the *importance coefficient*

$$K(h_t) \stackrel{\text{def}}{=} \frac{\Pr(h_t \mid \mu)}{\Pr(h_t \mid \mu')} = \Pi_{t'=0}^{t-1} \frac{\mu(s_{t'}, a_{t'})}{\Pr(a_{t'} \mid \text{Trunc}(h_t, t'), \mu')} ,$$

where $\text{Trunc}(h_t, t')$ is h_t truncated at time t'. For instance, if μ' is a stationary policy ($\mu'(s,\cdot)$ is a probability distribution on actions, for all s), then

$$K(h_t) = \Pi_{t'=0}^{t-1} \frac{\mu(s_{t'}, a_{t'})}{\mu'(s_{t'}, a_{t'})} .$$

As $\mathrm{E}_\mu[X(h_t)] = \mathrm{E}_{\mu'}[K(h_t)X(h_t)]$ for all random variables $X(h_t)$, we still get an unbiased estimate of the gradient. Note that the importance coefficients may be

1. **Initialize** the controller weights w_{sa};
2. **Beginning of a trial**:
   - for all (s, a): $N_s \leftarrow 0$, $N_{sa} \leftarrow 0$;
   - $R \leftarrow 0$;
   - $K \leftarrow 1$;
   - $h \leftarrow (s_0)$;
3. **At each time-step t of the trial**:
   - with probability ϵ: $a_t \leftarrow \mu_e(h)$,
     with prob. $1 - \epsilon$: draw a_t at random following $\mu(s_t, \cdot)$;
   - if $a_t = \mu_e(h)$: $K \leftarrow K\mu(s_t, a_t)/(\epsilon + (1 - \epsilon)\mu(s_t, a_t))$,
     else: $K \leftarrow K/(1 - \epsilon)$;
   - $N_{s_t} \leftarrow N_{s_t} + 1$, $N_{s_t a_t} \leftarrow N_{s_t a_t} + 1$;
   - execute an action a_t, receive s_t, r_t from environment;
   - $R \leftarrow R + \gamma^t r_t$;
   - append the triple (a_t, r_t, s_{t+1}) to h;
4. **End of the trial**:
   - for all (s, a): $w_{sa} \leftarrow w_{sa} + \alpha K R(N_{sa} - \mu(s, a)N_s)/\theta$;
5. **Loop**: return to 2.

Figure 3: Algorithm 3: an off-policy implementation of REINFORCE that does not take into account the causality of rewards. During the learning trials, the algorithm executes a mixture of the deterministic non-stationary policy μ_e with probability ϵ, and of the current policy μ with probability $1 - \epsilon$.

calculated incrementally and without knowing the dynamics of the environment. Moreover, μ' may be any policy, it does not have to be stationary (as is μ). Notably, μ' can use any type of extra information such as counters of state visits. The only requirement is that any trajectory possible under μ is still possible under μ'. Hence we cannot use any *deterministic* policy. But we can, for instance, mix such a policy with the current policy, that is, at each time step we follow a deterministic policy with probability $\epsilon \in [0, 1)$, and the current policy μ with probability $(1 - \epsilon) > 0$.

Figures 3 and 4 present the algorithms obtained if μ' is the mixture of the deterministic non-stationary policy $\mu_e : \bigcup_{t=1}^{\infty} H_t \to A$ with probability ϵ, and of the current policy with probability $1 - \epsilon$. The variable h is used to store the history of the trial, which is formally necessary because we defined μ_e as a function of the whole past history. However, the non-stationary policy μ_e may not use such a complete memory. For instance, if μ_e bases the choice of next action only on counters of state visit, then we do not need to remember the whole previous history, we just have to maintain the necessary counters.

3.2 Weighted Importance Sampling

Numerical simulations using algorithms 3 and 4 showed a big instability that increases with ϵ (see section 5). This instability is the expression of a known drawback of IS when the sampling distribution (the strategy used during learning) differs a lot

1. **Initialize** the controller weights w_{sa};
2. **Beginning of a trial**:
   - $N_s \leftarrow 0$, $N_{sa} \leftarrow 0$, for all (s, a);
   - $\Delta w_{sa} \leftarrow 0$, for all (s, a);
   - $K \leftarrow 1$;
   - $h \leftarrow (s_0)$;
3. **At each time-step t of the trial**:
   - with probability ϵ: $a_t \leftarrow \mu_e(h)$,
     with prob. $1 - \epsilon$: draw a_t at random following $\mu(s_t, \cdot)$;
   - if $a_t = \mu_e(h)$: $K \leftarrow K\mu(s_t, a_t)/(\epsilon + (1 - \epsilon)\mu(s_t, a_t))$,
     else: $K \leftarrow K/(1 - \epsilon)$;
   - $N_{s_t} \leftarrow N_{s_t} + 1$, $N_{s_t a_t} \leftarrow N_{s_t a_t} + 1$;
   - execute an action a_t, receive s_t, r_t from environment;
   - for all (s, a):
     $\Delta w_{sa} \leftarrow \Delta w_{sa} + K\gamma^t r_t(N_{sa} - \mu(s, a)N_s)/\theta$;
   - append the triple (a_t, r_t, s_{t+1}) to h;
4. **End of the trial**:
   - for all (s, a): $w_{sa} \leftarrow w_{sa} + \alpha\Delta w_{sa}$;
5. **Loop**: return to 2.

Figure 4: Algorithm 4: an off-policy implementation of REINFORCE that takes into account the causality of rewards.

from the target distribution (the current policy) (Zlochin & Baram, 2000). In this case, very unlikely events are associated with huge importance coefficients. Hence, whenever they happen, they induce devastating weight updates that can stick the algorithm to a very bad policy. The technique known as *weighted importance sampling* has been designed to alleviate this drawback (Precup et al., 2000). In REINFORCE, it consists of estimating $\mathrm{E}_\mu\left[\bar{C}_{sa}\right]$ by drawing n samples $\bar{C}_1, \bar{C}_2 \ldots \bar{C}_n$ of $\bar{C}_{sa}(h_t)$ following a sampling policy μ', calculating the n associated importance coefficients $K_1, K_2 \ldots K_n$, and using the quantity $\sum_{i=1}^n K_i \bar{C}_i / \sum_{i=1}^n K_i$ as an estimate of the gradient. This estimate is biased, but the bias tends to 0 as n tends to infinity. It is often a faster and more stable estimate than simple IS's estimate.

Figure 5 presents the algorithm obtained if we use look-up tables and Boltzmann law. This algorithm is in the line of algorithm 1 and 3, that is, it does not perform optimal credit assignment. Unfortunately, there is no incremental implementation of weighted IS in the line of algorithm 2. To integrate weighted IS in algorithm 2, we need to sum in a variable Δ^t_{sa} the values of $X^t_{C_{sa}}$ during the n learning trials preliminary to a weight-update, *for each time t*. In the same way, the sum of the importance coefficient $K(h_t)$ must be calculated in a different variable $\mathcal{K}^t$ for each time t. Therefore, the algorithm loses its incrementality. As we will see in section 5, algorithm 5 achieved the best performances of all the algorithms presented in this paper. This shows that it can be worthwhile to sacrifice the quality of temporal credit assignment to the possibility of using efficient sampling policies.

1. Initialize the controller weights w_{sa};
2. Initialize variables:
- for all (s,a): $\Delta_{sa} \leftarrow 0$;
- $\mathcal{K} \leftarrow 0$;

3. For $i = 1$ **to** n: (executes n learning trials)
- Beginning of a trial:
  - for all (s,a): $N_s \leftarrow 0$, $N_{sa} \leftarrow 0$;
  - $R \leftarrow 0$;
  - $K \leftarrow 1$;
  - $h \leftarrow (s_0)$;
- At each time-step t of the trial:
  - with probability ϵ: $a_t \leftarrow \mu_e(h)$;
    with prob. $1-\epsilon$: draw a_t at random following $\mu(s_t, \cdot)$;
  - if $a_t = \mu_e(h)$: $K \leftarrow K\mu(s_t, a_t)/(\epsilon + (1-\epsilon)\mu(s_t, a_t))$;
    else: $K \leftarrow K/(1-\epsilon)$;
  - $N_{s_t} \leftarrow N_{s_t} + 1$, $N_{s_t a_t} \leftarrow N_{s_t a_t} + 1$;
  - execute an action a_t, receive s_t, r_t from environment;
  - $R \leftarrow R + \gamma^t r_t$;
  - append the triple (a_t, r_t, s_{t+1}) to h;
- End of the trial:
  - for all (s,a): $\Delta_{sa} \leftarrow \Delta_{sa} + KR(N_{sa} - \mu(s,a)N_s)/\theta$;
  - $\mathcal{K} \leftarrow \mathcal{K} + K$;

4. Update policy:
- for all (s,a): $w_{sa} \leftarrow w_{sa} + \alpha\Delta_{sa}/\mathcal{K}$;

5. Loop: return to 2.

Figure 5: Algorithm 5: an off-policy implementation of REINFORCE based on weighted importance sampling.

4 Complexity Issues

As we said in the introduction, *off-policyness* offers many advantages, including the possibility to optimize several controllers simultaneously (Sutton et al., 1998), and the possibility to reuse past experience (Kearns et al., 1999). In this section, we show that off-policy implementations can also be used to reduce the sample complexity of REINFORCE, as measured by two variables: the number of leaning trials necessary to reach a given performance level, and the length of learning trials learning.

4.1 Number of Learning Trials

The technique of IS was originally designed to increase the accuracy of Monte Carlo estimates by reducing their variance. In the context of REINFORCE, a more accurate estimate of the gradient at the end of each trial would imply that less trials are needed to reach a given performance. Therefore, the first motivation for choosing the sampling policy is speeding up learning in terms of the number of trials.

A classical result of the theory of Monte Carlo estimation (Kahn & Marshall, 1953) implies that the optimal sampling policy, that is, the policy that minimizes

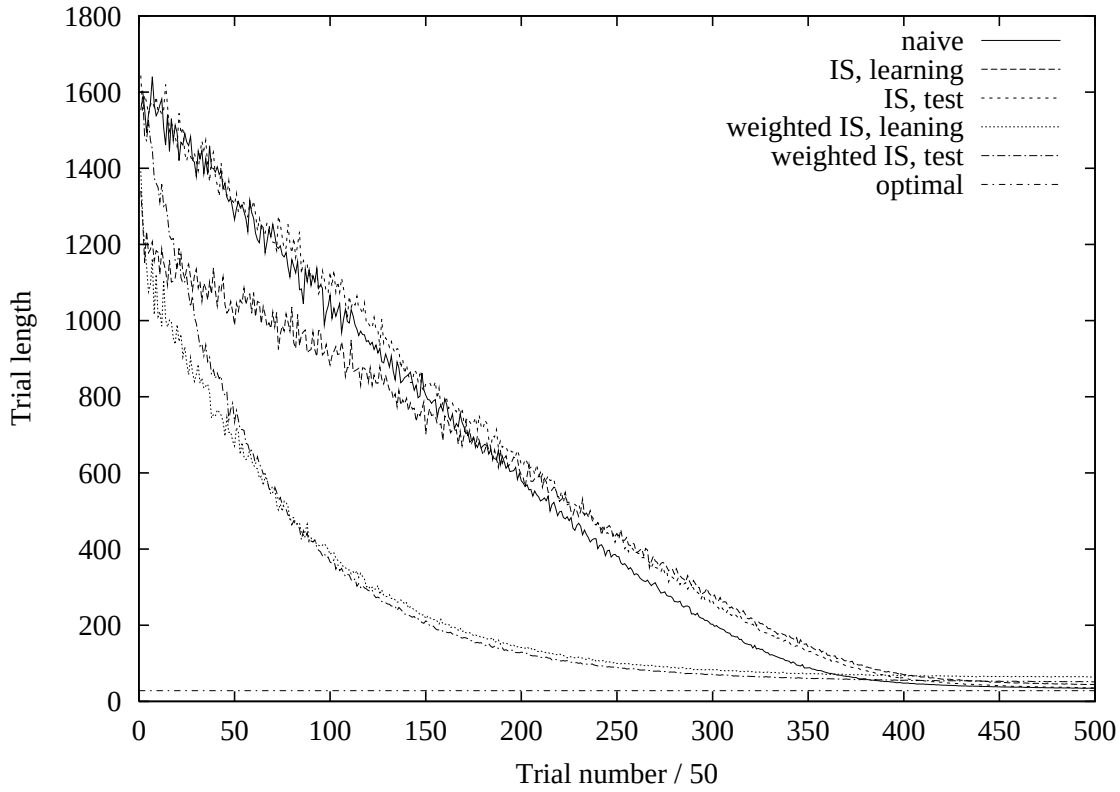

Figure 6: Learning curves obtained in the 15×15 partially-observable no-reset variant of the problem, using Meuleau and Bourgine's global exploration policy ($\epsilon = 0.1$, goal-reward, $\gamma = 0.95$, $\alpha = 0.01$, $n = 3$, $\theta = 1$, average of 30 runs)

the variance of IS estimate, should give to a sequence $\bar{h}_T$ a probability proportional to its *importance* $\Pr(\bar{h}_T \mid \mu)|\bar{C}_{sa}(\bar{h}_T)|$. Unfortunately, it may not be possible to implement this optimal sampling policy of $\bar{C}_{sa}$ for all (s, a) at the same time. Moreover, to implement this sampling scheme in REINFORCE, the agent needs to know the dynamics of the environment (transition probabilities, reward function). This contradicts assumptions in RL. There are techniques which allow to approximate the optimal distribution, by changing the sampling distribution during the trial, while keeping the resulting estimates unbiased via reweighting of samples (see for example *adaptive sampling* (Oh & Berger, 1992)).

Another issue is that bias estimates may sometimes be more efficient that unbiased estimates. In general, there is a bias/variance dilemma in Monte Carlo estimation. Zlochin and Baram (Zlochin & Baram, 2000) showed that the biased estimate obtained by using a sampling distribution different from the target distribution, but not correcting the estimate with importance coefficients, may have a smaller estimation error than IS's unbiased estimate when the number of samples is small. This is due to the big variance of simple IS's estimate. Our simulation results show that the best algorithm is algorithm 5, which uses a biased estimate. In a stochastic gradient

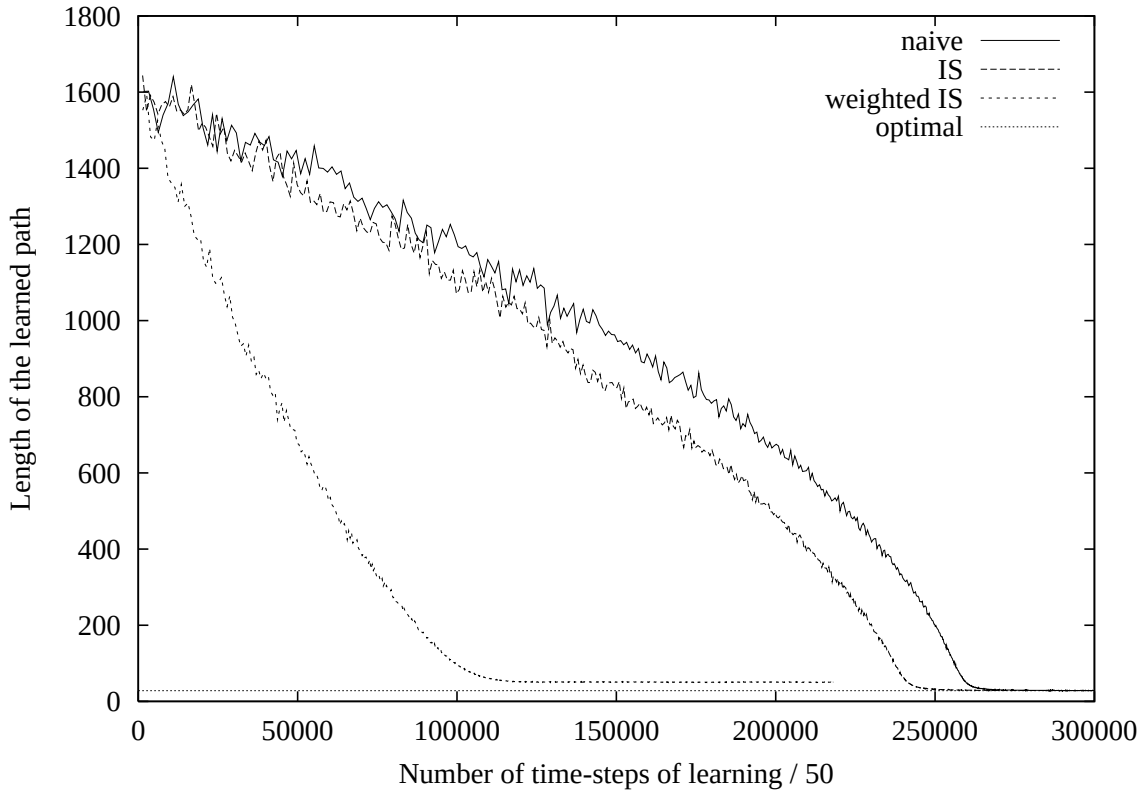

Figure 7: Sample complexity of the algorithms measured in the conditions of fig. 6. $\epsilon = 0.1$.

algorithm, a bias estimate may also be interesting because the bias is efficient in some sense. For instance, the bias may help get out of (bad) local optima. Although, our simulation results do not demonstrate this ability.

4.2 Length of Learning Trials

The number of trials does not perfectly reflect the sample complexity of learning. The total number of interactions with the environment depends also on the length of learning trials. It constitutes a second consideration to take into account when choosing the sampling policy. The issues involved here are very similar to early work on the complexity of reinforcement learning.

The first studies on the complexity of reinforcement learning focused on the length of Q-learning's first trial in goal-achievement problems. First Whitehead showed that the expected length of the QL's first trial can grow exponentially with the size of the environment (Whitehead, 1991). This is basically because QL dynamics may result in a random walk during the first trial, and random walks can take exponential time. Typically, difficult problems are those where more actions take you away from the goal than bring you closer, or where in each state a reset action brings you back to the starting state (reset problems). Thrun showed later that the

use of directed (counter-based) exploration techniques could alleviate this problem in QL, guaranteeing polynomial complexity of the first trial in any deterministic MDP (Thrun, 1992). Simultaneously, Koenig and Simmons showed that some changes in the parameters, such as the initial Q-values and the reward function, could have the same effect in any MDP (Koenig & Simmons, 1996).

It is easy to see that REINFORCE faces the same difficulties as undirected QL in the same kind of environments. Depending on the way the controller is initialized in the beginning of learning, the complexity of the first trial(s) may be very bad due to initial random walk of the algorithm. For instance one may show that there is a finite MDP such that, if the controller is initialized by drawing the action probabilities $\mu(s, \cdot)$ uniformly in the simplex, then the expected length of REINFORCE's first trial is infinite (Meuleau, 2000). Also, as the update consecutive to one trial may not change the policy a lot, one may expect a very bad performance during several trials in the beginning of learning, not only the very first one. It is clear that changing the reward model—as suggested by Koenig and Simmons for QL—may not reduce the expected length of REINFORCE's very first trial: as we do not update the weights *during* a trial, the length of the first trial depends only on the initial controller. However, a change in the reward model implies a change of objective function. Therefore, it will be perceptible after the first weight-update. The off-policy implementations of REINFORCE proposed in the previous section can be used with efficient directed exploration policies to avoid initial random walk. Algorithms 3 to 5 can be used with μ_e set to many directed exploration policies (which are often deterministic and non stationary), including Thrun's counter-based (1992) and Meuleau and Bourgine's global exploration policy (1999). Moreover, the algorithms can easily be adapted to stochastic exploration policies. It can be shown that, if we mix the current policy with Thrun's counter-based exploration policy , then the expected length of the first trial of algorithms 3 to 5 in any deterministic MDP tends to a polynomial in the size of the MDP, when ϵ tends to 1 (provided that the controller is initialized uniformly) (Meuleau, 2000).

There is then double motivation when choosing the sampling policy to execute during learning trials. These objectives may be contradictory or compatible. In the next section, we present simulations results that indicate that it is possible to find sampling policies that reduce both the number of learning trials and their length.

5 Numerical Simulations

We tested our off-policy algorithms on a simple grid-world problem consisting of an empty square room where the starting state and the goal are two opposite corners. Four variants of this problem were tried: there may or may not be a reset action, and the problem can be fully observable or partially observable. When the problem is partially observable, the agent cannot perceive its true location, but only the presence or absence of walls in its immediate proximity. This problem was not designed to

be hard for the algorithms, and every version of REINFORCE converges easily to the *global* optimum. However, it allows to compare well the different variants in terms of learning speed.

We used the algorithms presented in figures 1 to 5. Controllers were initialized by setting $\mu(s, \cdot)$ to the uniform distribution on actions, for all states s. We tried several parameter settings and environments of different sizes. In the choice of the sampling policy, we focused on the objective of reducing the trials length: we tried several directed exploration policies as μ_e, including greedy counter-based, Thrun's counter-based (1992) and an indirect (QL) implementation of a global counter-based exploration policy proposed by Meuleau and Bourgine (Meuleau & Bourgine, 1999). Exploration strategies designed for fully observable environments where naively adapted by replacing states by observations in the formulas, when dealing with the partially observable variants of the problem. These (empirical) exploration policies derived in the goal of minimizing the length of trials, appear empirically to have a beneficial influence on the number of trials too, when they are implemented efficiently as in algorithm 5.

In an on-policy algorithm, the observed length of learning trials represents well the quality of the policy learned so far. However, it is not the case in off-policy algorithm, where the length of the learning trials depends only partially on the current policy. To measure the quality of the learned policy, we perform after each learning trial of an off-policy algorithm, a test trial in which all the variables are frozen and (exclusively) the current policy is executed. The evolution of the length of both the learning trials and the test trials is recorded.

The results of these experiments are qualitatively independent of the variant and the size of the problem (although we were unable to run experiments in reset problems of reasonable size, due to the exponential complexity of random walk in these problems). The best performances were obtained using Meuleau and Bourgine's global exploration policy. In general *naive* sampling is very stable and slow. With small values of ϵ, simple IS allows reducing the length of learning trials without affecting the quality of policy learned. However, it rapidly becomes very unstable and systematically jumps to very bad policies as ϵ increases. Algorithm 5 is by far the most efficient algorithm. It stays stable when ϵ approaches 1, even with relatively small number of learning trials ($n = 5$). It can thus be used with high values of ϵ, which allows dramatic reduction of the trials' length. Our experiments show that it always comes with a considerable improvement of the quality of the policy learned over several trials. Figures 6, 8 and 7, 9 present sample results. Figures 6, 8 contain classical learning curves, that is, plots of the evolution of the length of learning trials as learning progresses. The evolution of the length of the test trials of off-policy algorithms is also represented. It shows how the quality of the policy learned by these algorithms evolves. The same data is re-plotted in a different form in fig. 7, 9. The graph presented here represent the evolution of the quality of the policy learned as a function of the total number of time-steps of interactions with the environment. The quality of the policy learned is measured by the length of learning trials in the

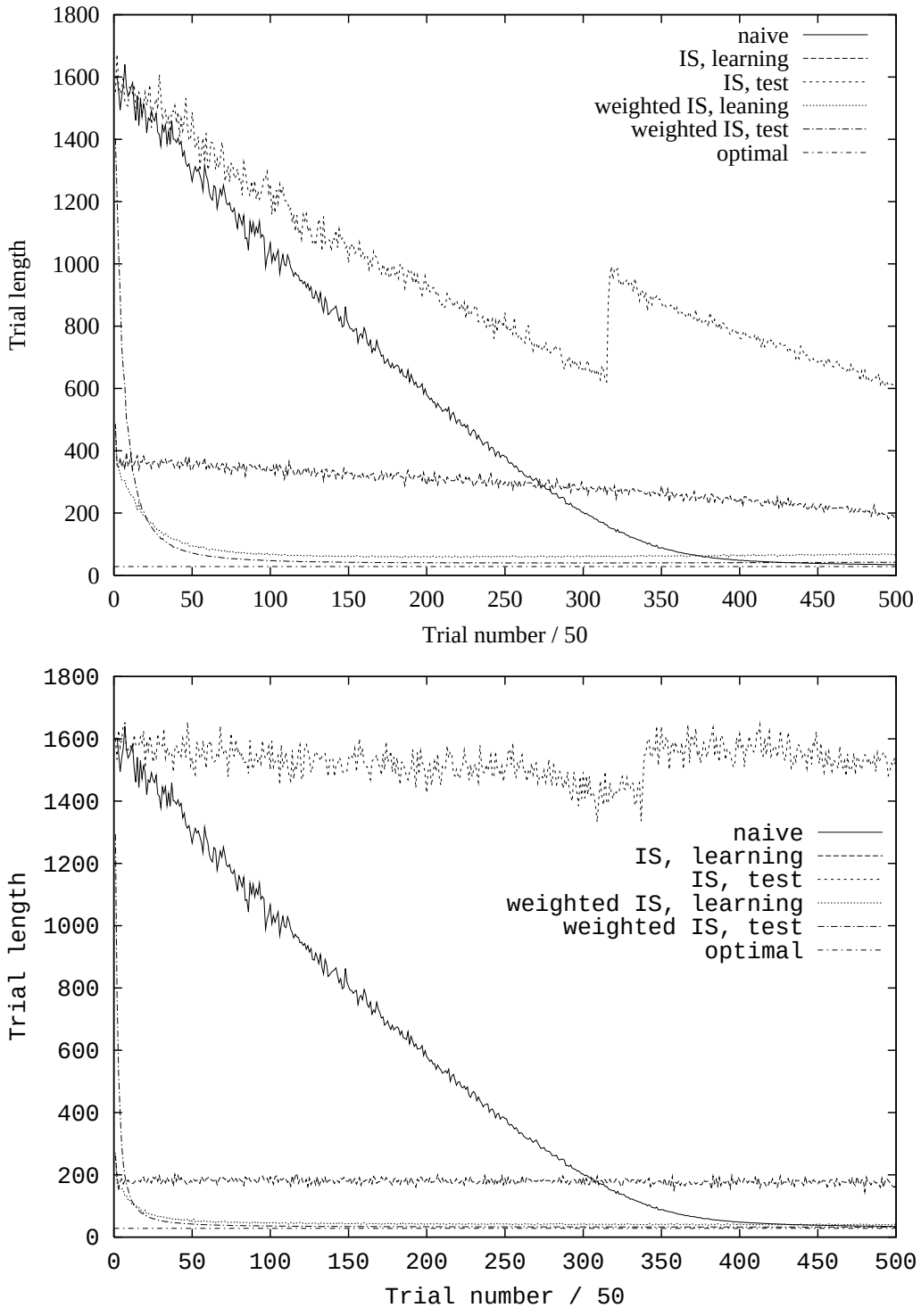

Figure 8: Learning curves obtained in the 15×15 partially-observable no-reset variant of the problem, using Meuleau and Bourgine's global exploration policy. Top: $\epsilon = 0.3$; bottom: $\epsilon = 0.5$. (goal-reward, $\gamma = 0.95$, $\alpha = 0.01$, $n = 3$, $\theta = 1$, average of 30 runs)

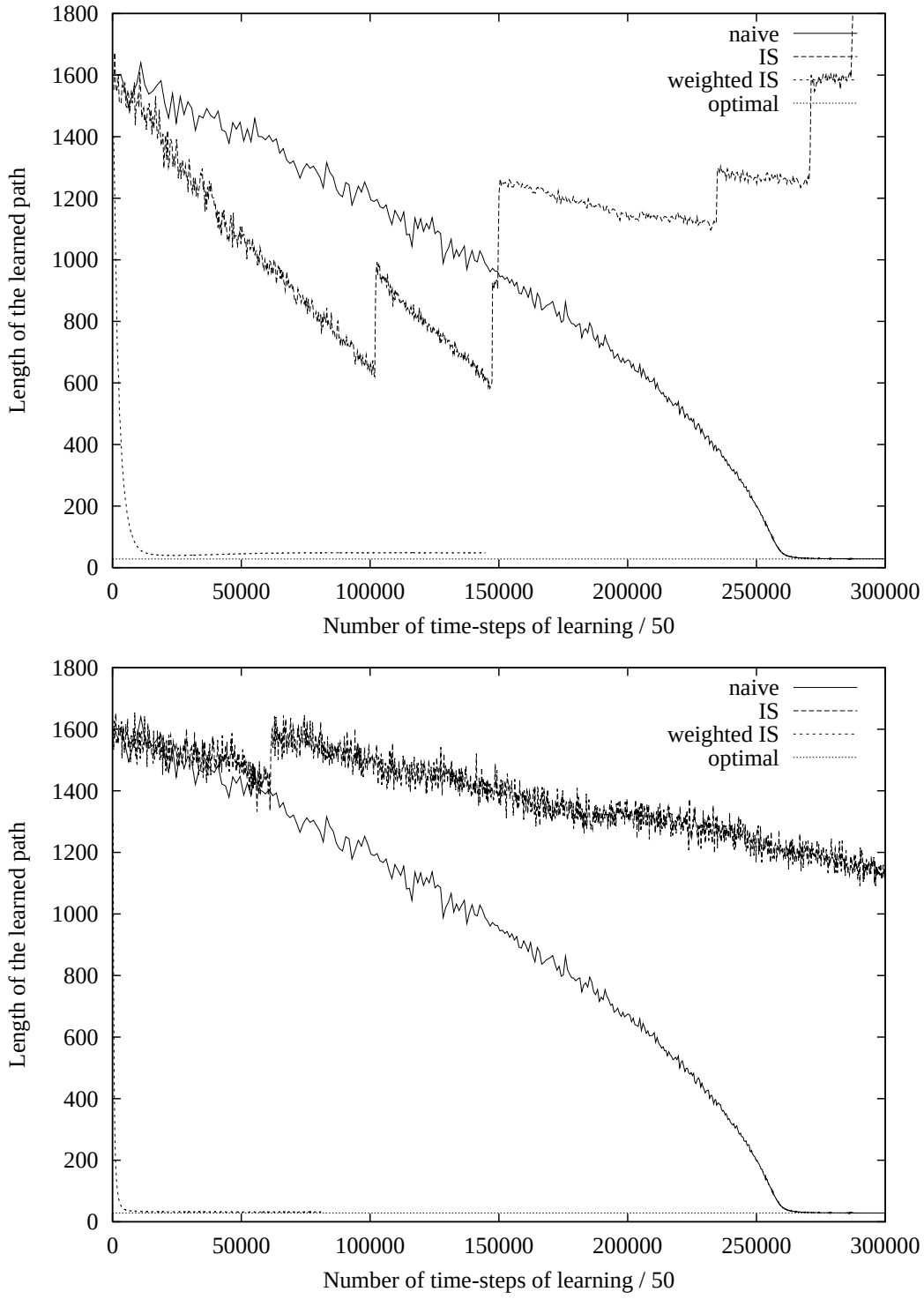

Figure 9: Sample complexity of the algorithms measured in the same conditions as in fig. 8. Top: $\epsilon = 0.3$; bottom: $\epsilon = 0.5$.

case of the on-policy algorithm, and by the length of test trials in the case of off-policy algorithms. Therefore, the graphs of fig. 7, 9 represent accurately the sample complexity of the algorithms.

6 Conclusion

We have proposed off-policy implementations of REINFORCE based on the technique of importance sampling. We have argued that these algorithms may be used to reduce the sample complexity of learning. Finally, we presented simulation results where off-policy implementations allow reducing the number of learning trials as well as their length, which results in a dramatic acceleration of learning.

References

Baird, L., & Moore, A. (1999). Gradient descent for general reinforcement learning. In *Advances in neural information processing systems, 12*. Cambridge, MA: MIT Press.

Bartlett, P., & Baxter, J. (2000). *A biologically plausible and locally optimal learning algorithm for spiking neurons* (Technical Report). CSL, Australian National University.

Baxter, J., & Bartlett, P. (2000). Reinforcement learning in POMDP's via direct gradient ascent. *Proceedings of ICML-00*.

Kaelbling, L., Littman, M., & Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*.

Kahn, H., & Marshall, A. (1953). Methods of reducing sample size in Monte Carlo computations. *Journal of the Operations Research Society of America*, *1*, 263–278.

Kearns, M., Mansour, Y., & Ng, A. (1999). Approximate planning in large pomdps via reusable trajectories. *Advances in Neural Information Processing Systems, 12*. Cambridge, MA: MIT Press.

Kim, K., Dean, T., & Meuleau, N. (2000). Approximate solutions to factored markov decision processes via greedy search in the space of finite controllers. *Proceedings of AIPS-00*, 323–330.

Koenig, S., & Simmons, R. (1996). The effect of representation and knowledge on goal-directed exploration with reinforcement learning alogrithms. *Machine Learning*, *22*.

Littman, M. (1994). Memoryless policies: Theoretical limitations and practical results. *Proceedings of SAB-94*.

Marbach, P., & Tsitsiklis, J. N. (2001). Simulation-based optimization of Markov reward processes. *IEEE Transactions on Automatic Control*, *To appear*.

Meuleau, N. (2000). *The complexity of the first trial in REINFORCE.* Unpublished manuscript.

Meuleau, N., & Bourgine, P. (1999). Exploration of multi-state environments: Local measures and back-propagation of uncertainty. *Machine Learning*, *35*, 117–154.

Meuleau, N., Peshkin, L., Kim, K., & Kaelbling, L. (1999). Learning finite-state controllers for partially observable environments. *Proceedings of UAI-99*, 127–136.

Oh, M., & Berger, J. (1992). Adaptive importance sampling in monte carlo integration. *Journal of Statistical Computing and Simulation.*, *41*, 143–168.

Peshkin, L., Kim, K., Meuleau, N., & Kaelbling, L. (2000). Learning to cooperate via policy search. *Proceedings of UAI-00*, 489–496.

Peshkin, L., Meuleau, N., & Kaelbling, L. (1999). Learning policies with external memory. *Proceedings of ICML-99*, 307–314.

Precup, D., Sutton, R., & Singh, S. (2000). Eligibility traces for off-policy policy evaluation. *Proceedings of ICML-00*, 759–766.

Puterman, M. (1994). *Markov decision processes: Discrete stochastic dynamic programming.* New York, NY: Wiley.

Rubinstein, R. (1981). *Simulation and the Monte Carlo method.* New York, NY: Wiley.

Singh, S., Jaakkola, T., Littman, M., & Szpesvari, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, *39*, 287–308.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction.* Cambridge, MA: MIT Press.

Sutton, R., Precup, D., & Singh, S. (1998). Intra-option learning about temporally abstract actions. *Proceedings of ICML-98*, 556–564.

Thrun, S. (1992). *Efficient exploration in reinforcement learning* (Technical Report CS-92-102). Carnegie Mellon University, Pittsburgh, PA.

Watkins, J., & Dayan, P. (1992). Technical note: Q-learning. *Machine Learning*, *8*, 279–292.

Whitehead, S. (1991). A complexity analysis of cooperative mechanisms in reinforcement learning. *Proceedings of AAAI-91.*

Wiering, M., & Schmidhuber, J. (1997). HQ-Learning. *Adaptive Behavior*, *6*, 219–246.

Williams, R. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.

Zlochin, M., & Baram, Y. (2000). The bias-variance dilemma of the Monte Carlo method. *Machine Learning*, Submitted.