Reinforcement Learning Through Global Stochastic Search in N-MDPs

Matteo Leonetti[1], Luca Iocchi[1], and Subramanian Ramamoorthy[2]

[1] Department of Computer and System Sciences,
Sapienza University of Rome,
via Ariosto 25, Rome 00185, Italy
[2] School of Informatics,
University of Edinburgh,
10 Crichton Street, Edinburgh EH8 9AB, United Kingdom

Abstract. Reinforcement Learning (RL) in either fully or partially observable domains usually poses a requirement on the knowledge representation in order to be sound: the underlying stochastic process must be Markovian. In many applications, including those involving interactions between multiple agents (e.g., humans and robots), sources of uncertainty affect rewards and transition dynamics in such a way that a Markovian representation would be computationally very expensive. An alternative formulation of the decision problem involves partially specified behaviors with choice points. While this reduces the complexity of the policy space that must be explored - something that is crucial for realistic autonomous agents that must bound search time - it does render the domain Non-Markovian. In this paper, we present a novel algorithm for reinforcement learning in Non-Markovian domains. Our algorithm, Stochastic Search Monte Carlo, performs a global stochastic search in policy space, shaping the distribution from which the next policy is selected by estimating an upper bound on the value of each action. We experimentally show how, in challenging domains for RL, high-level decisions in Non-Markovian processes can lead to a behavior that is at least as good as the one learned by traditional algorithms, and can be achieved with significantly fewer samples.

Keywords: Reinforcement Learning

1 Introduction

Reinforcement Learning (RL) in its traditional formulation has been successfully applied to a number of domains specifically devised as test beds, while scaling to large, and more realistic applications is still an issue. RL algorithms, either *flat* or hierarchical, are usually grounded on the model of Markov Decision Processes (MDPs), that represent controllable, fully observable, stochastic domains. When the environment is not observable and not so well understood, however, traditional RL methods are less effective and might not converge. For this reason, the

vast majority of work on RL has focused on Markovian domains, and the algorithms for abstracting over the state space, creating a more compact knowledge representation, are designed to ensure the Markov property is maintained.

If the application does not allow the designer to create a reliable description of the state space, such that the representation is Markovian, MDPs are no longer suitable. Partially Observable MDPs overcome part of this limitation: they allow the actual state space to be accessed only through *observations*, but still require the specification of such a space so that an underlying Markovian process exists. Where applicable, POMDPs scale to increasingly larger domains, but there are tasks in which observability is not the only concern. For instance, if other agents (including humans) influence the task in ways that cannot be characterized upfront, then the typical MDP formulation or even versions (e.g., interactive or decentralized) of POMDPs do not capture this issue [19]. In such scenarios, it may be more useful to synthesize the overall behavior as a composition of partially specified local behaviors - tuned to a localized interaction context - with choices between them in order to adapt to the changing environment and task [7]. The composition of local behaviors may depend on high level observations, and an approach that acknowledges the Non-Markovian nature of the problem is required.

In the literature, three methods have been used in Non-Markovian domains. The first one consists in applying direct RL to observations, relying on *eligibility traces* [13]. The second one is a local optimization algorithm, MCESP [11], made sound by a specific definition of the equation used for value prediction. The third one is pure search in policy space, dominated by policy gradient [2], a category of methods that search in the continuous space of stochastic policies. The first two methods make use of value functions, while the last one avoids them altogether. In this paper, we introduce a novel Reinforcement Learning algorithm to learn in Non-Markovian domains, that performs a global stochastic search in policy space. As such, it belongs to the third category, but it is also based on an action-value function, and uses it to store an estimated upper bound of each action's value to bias the search. Our control method makes decisions locally, and separately at each choice point, while the prediction method takes into account the long-term consequences of actions. In general N-MDPs (as well as in stochastic processes on top of POMDPs' observations) the reward, and therefore the decisions, may depend on the whole chain of observations and actions. In problems of practical interests, however, the choices can be efficiently made separately in many cases. The prediction method, nonetheless, must take the effect of subsequent choices into account.

We show experimentally how the space of deterministic policies can be searched effectively by exploiting action values' upper bounds locally and evaluating policies globally. We first introduce two simple domains, taken from the literature of partially observable processes, that allow for the comparison with other methods. Then, we apply our methodology to Keepaway [15], a challenging and more realistic domain for RL. Here, instead of relying on function approximation, we consider a partially specified behavior with coarse-grained choices. While

function approximation attempts to learn the best decision for each state, generalizing from *similar* ones (whose definition depends on the specific approximator), our method aims at learning, at specific choice points among high-level actions, which option performs best across all the situations that actually happen, biased by their frequency. Even if RL (in MDPs) provably converges to the optimal behavior, on Keepaway it does not do so in any reasonable time. We show how a behavior better than the ones reported in the literature can be obtained in a number of samples an order of magnitude smaller. The results suggest that, when the domain does not allow a compact Markovian description, giving up the Markov property in order to reduce the representation might, with the appropriate algorithms, provide a good behavior (possibly optimal in the given representation) much faster than methods that achieve optimality in the underlying MDP.

2 Background and Related Work

In the following, we first define the notation, and then describe the methods most closely related to our own.

2.1 Notation

A Markov Decision Process is a tuple $MDP = \langle S, A, T, \rho \rangle$ where:

- S is a set of *states*
- A is a set of *actions*
- $T : S \times A \times S \to [0, 1]$ is the transition function. $T(s, a, s') = Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that the current state changes from s to s' by executing action a. Since $T(s, a, \cdot)$ is a probability distribution, then $\sum_{s' \in S} T(s, a, s') = 1 \ \forall \ s \in S$ and $a \in A$. If $T(s, a, s') = \{0, 1\}$ the system is said to be *deterministic*, otherwise it is *stochastic*.
- $\rho : S \times A \times \mathbb{R} \to [0, 1]$ is the reward function. $\rho(s, a, r) = Pr(r_{t+1} = r | s_t = s, a_t = a)$ is the probability to get a reward r from being in state s and executing action a. Analogously to the transition function, $\rho(s, a, \cdot)$ is a probability density function and $\int_{\mathbb{R}} \rho(s, a, r) dr = 1$. If the reward function is defined over a discrete subset $P \subset \mathbb{N}$, ρ is a probability distribution and the reward is said to be *deterministic* if $\rho(s, a, r) = \{0, 1\} \ \forall s \in S$, $a \in A$, and $r \in P$.

The behavior of the agent is represented as a function $\pi : S \times A \to [0, 1]$ called a *stationary policy*, where $\pi(s, a)$ is the probability of selecting action a in state s. If $\pi(s, a) = \{0, 1\} \ \forall \, s \in S$ and $a \in A$ the policy is *deterministic*.

A policy π and an initial state s_0 determine a probability distribution $\mu(\omega)$ over the possible sequences $\omega = (\langle s_t, a_t, r_{t+1} \rangle, t \geq 0)$. Given such a sequence, we define the *cumulative discounted reward* as

$$R(\omega) = \sum_{t \geq 0} \gamma^t r_{t+1} \tag{1}$$

where $0 < \gamma \leq 1$ is the *discount factor*. If $\gamma = 1$ the reward is *undiscounted*, which is allowed only if the MDP is episodic (it has at least an absorbing state, which is never left once entered) otherwise the total reward could diverge.

Analogously to Markov Decision Processes, a Partially Observable MDP, is a a tuple $\langle S, A, T, \rho, Z, O\rangle$, where $\langle S, A, T, \rho\rangle$ is an underlying MDP whose current state is not directly accessible. Instead of perceiving an element from S, the agent is given an element of O, the set of *observations*, which relates to the underlying state through the function $Z : O \times A \times S \rightarrow [0, 1]$ such that $Z(o, a, s) = Pr(o|s, a)$ is the probability of observing o when executing a in s. We consider the case of POMDPs in which S, T, and Z are unknown. Ignoring the actual state space and considering the controllable stochastic process $\langle O, A, T_o, \rho_o\rangle$ over observations, the unknown distribution

$$T_o(o, a, o') = \sum_{s \in S} Pr(s|o) * \sum_{s' \in S} T(s, a, s')Z(o', a, s)$$

depends not only on the current observation, but also on the underlying actual state and on the system's dynamics. The actual state in turn depends on the initial state and on the policy followed. Analogously, the reward obtained after each observation does not only statistically depend on the observation alone, and is therefore Non-Markovian.

Given an N-MDP $= \langle O, A, T_o, \rho_o\rangle$, we aim at computing the deterministic stationary reactive policy $\pi(o) = a$ that maximizes the expected value of the cumulative discounted reward:

$$R = \sum_{s_0} \sum_{\omega} \mu(\omega|\pi, s_0)Pr(s_0)R(\omega)$$

In the following we summarize what has been proved for direct RL methods in these circumstances.

2.2 Previous results for N-MDPs

In Markov Decision Processes sub-optimal policies are never fix-points of policy iteration, so that each step produces not only a different policy, but a better one. MDPs are, therefore, well suited to hill-climbing, since optimal policies, and only those, are equilibrium points in MDPs, while this is not true in general for N-MDPs. hPOMDPs constitute a class of POMDPs in which the history of observations and actions is enough to determine the current state. Pendrith and McGarity [10] analyze the stability of TD(λ) and first-visit Monte Carlo [16]. They prove the following results:

- if a first-visit MC is used for an hPOMDP where $\gamma = 1$, then the optimal observation-based policies will be learning equilibria.
- the previous result does not apply to $\gamma = [0, 1)$
- if a TD(λ) method of credit assignment is used for direct RL on a N-MDP, then for $\gamma < 1$ it is not guaranteed that there exists an optimal observation-based policy representing a learning equilibrium.

Perkins and Pendrith [12] carry this analysis further, and include the exploration policy explicitly. They prove that there exists a learning equilibrium for 1-step TD methods if the exploration policy is *continuous* in the action values, while most of the former analysis had been conducted with ϵ-greedy which is discontinuous. So, for instance, following SoftMax, that assigns to every action a probability according to a Boltzmann distribution:

$$Pr(a|s) = \frac{e^{Q(s,a)/\tau}}{\sum_{a' \in A} e^{Q(s,a')/\tau}} \tag{2}$$

both Sarsa and Q-learning have at least one action-value function that is a learning equilibrium. The parameter τ in Eq. 2 balances exploration and exploitation: the higher τ the more the agent is likely to select a sub-optimal action (according to the *current* value function). The results prove that there exists a fixed point with respect to the update rule and a continuous exploration policy, but do not prove that such a fixed point can actually be reached. Moreover, the presented results do not consider that the exploration may change, for instance letting τ tend to zero.

2.3 A sound local algorithm

Perkins [11] redefined the value function to overcome the above difficulties with discounted problems.

Let $\mu_\pi(\omega)$ be the probability distribution over the possible trajectories determined by a policy π. The author splits the reward with respect to an observation o from one of these trajectories ω in:

$$\begin{aligned} V^\pi &= \mathbb{E}^\pi_{\omega \sim \mu} [R(\omega)] \\ &= \mathbb{E}^\pi_{\omega \sim \mu} [R_{pre-o}(\omega)] + \mathbb{E}^\pi_{\omega \sim \mu} [R_{post-o}(\omega)] \end{aligned} \tag{3}$$

where $R_{pre-o}(\omega)$ is the cumulative discounted reward before o is encountered in ω for the first time, while $R_{post-o}(\omega)$ is the reward after the first occurrence of o. In the following, we shall omit the subscript $\omega \sim \mu_\pi(\omega)$, but all traces are extracted from μ if not otherwise noted. The value of an observation-action pair $\langle o, a \rangle$, with respect to a policy π, is the value of the policy when π is followed everywhere except for o, in which a is executed instead. Such a policy is represented as $\pi \leftarrow \langle o, a \rangle$, and clearly $\pi = \pi \leftarrow \langle o, \pi(o) \rangle$. Its value is:

$$Q^\pi(o, a) = \mathbb{E}^{\pi \leftarrow \langle o,a \rangle} [R_{post-o}(\omega)] \tag{4}$$

This definition differs from the usual definition for MDPs in two important respects: (1) every time (and not just the first one) the observation o is encountered, the agent executes the action a; (2) the value of an observation-action pair is not the discounted return following o, but the expected discounted reward following *o at that point* of the trace.

While in MDPs the optimal policy is greedy with respect to the action-value function, this is not necessarily true for POMDPs. With the definition of the

value function just given, this property is retained to some extent. In particular, for all π and $\pi' = \pi \leftarrow \langle o, a \rangle$

$$V^{\pi} + \epsilon \geq V^{\pi'} \iff Q^{\pi}_{o,\pi(o)} + \epsilon \geq Q^{\pi}_{o,a}$$

Monte Carlo Exploring Starts for POMDPs (MCESP) is a family of algorithms, that make use of the value function of Equation 3 to compute a *locally optimal* policy. The gained capability to hill-climb brings about the theoretical guarantee of local optimality, at the cost of updating one state-action pair at a time. For this reason MCESP proved to be slower than $Sarsa(\lambda)$ in the domain presented in the original paper.

We define a new algorithm for stochastic search that retains some of the ideas behind MCESP, while attempting a biased global search. Our algorithm relies on the ability of Monte Carlo policy evaluation to estimate the current policy, and performs a form of branch and bound related to *confidence bounds* [1] for N-MDPs. Although stochastic policies could, in general, perform better on N-MDPs, we only search in the space of deterministic policies as we are interested in getting *good* policies in the shortest number of episodes possible. It has been shown how deterministic policies can often provide such behaviors in practice [8], and we provide more examples in the experimental section.

3 The Algorithm: SoSMC

The main idea behind the algorithm, Stochastic Search Monte Carlo (SoSMC), is based on the intuition that often a few bad choices disrupt the value of all the policies that include them. Taking those policies as if they were as valuable as any other just wastes samples. We would rather like to realize that those actions are not promising and not consider them unless we have tried all the other options. The strategy would consider all the policies with a probability proportional to how *promising* they are, which we believe is beneficial in at least two ways: (1) the algorithm reaches the optimal policy earlier; (2) during the phase of evaluation of those *promising* but suboptimal policies, the behavior is as good as the current information allows.

The algorithm (cf. Algorithm 1) is constituted by two parts: the *exploratory* phase and the *assessing* phase, as described in the next section.

3.1 Exploration: gathering information

The exploration initializes the Q-function to drive the execution in the subsequent phase. The aim of the initial exploration is to determine an upper bound for each state-action pair. For a number of episodes *exp_length* the agent chooses a policy according to some strategy Σ (e.g., uniformly at random), and in each pair $\langle o, a \rangle$ stores the highest value that any policy, going through $\langle o, a \rangle$, has ever obtained.

Consider the simple example of the N-MDP in Figure 1(a). This N-MDP has three states and four actions with a total of four policies. Let the reward returned

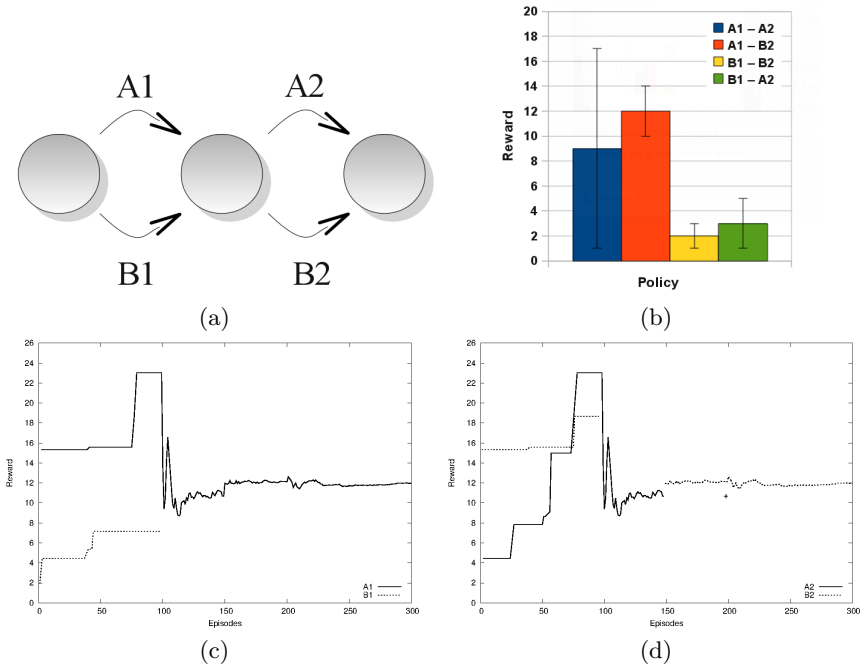


Fig. 1. A simple example of an N-MDP (a). The four policies return a reward normally distributed whose means and standard deviations are shown in (b). The evolution of the Q-function for the first state (actions A1 and B1) is represented in Figure (c), while for the second state (actions A2 and B2) is represented in Figure (d).

Algorithm 1 SoSMC

exp_length ← number of episodes in the exploratory phase
$n \leftarrow$ current episode
$t \leftarrow$ last exploratory episode
$\alpha(n, o, a) \leftarrow$ learning step parameter
initialize $Q(s, a)$ pessimistically
{Exploratory phase}
for $i = 1$ to *exp_length* **do**
  generate a trajectory ω according to a policy π extracted from a strategy Σ
  for all $o \in O$, $a \in A$ $s.t.\langle o, a\rangle$ is in ω **do**
    $Q(o, a) = max(Q(o, a), R_{post-o}(\omega))$
  end for
end for
{Assessing phase}
for all other episodes : n **do**
  if n is such that the current estimate is considered accurate **then**
    $t = n$
    $\pi \leftarrow$ a policy chosen from Σ
  else
    $\pi \leftarrow$ the last policy chosen
  end if
  {Possible policy change after an exploratory episode}
  if $n = t + 1$ **then**
    $\pi \leftarrow$ the policy that greedily maximizes Q
  end if
  generate a trajectory ω from π
  if $n = t$ **then**
    for all $o \in O$, $a \in A$ $s.t.\langle o, a\rangle$ is in ω **do**
      $Q(o, a) = max(Q(o, a), R_{post-o}(\omega))$
    end for
  else
    for all $o \in O$, $a \in A$ $s.t.\langle o, a\rangle$ is in ω **do**
      $Q(o, a) = (1 - \alpha(n, o, a))Q(o, a) + \alpha(n, o, a)R_{post-o}(\omega)$
    end for
  end if
end for

by each of those policies be normally distributed, with means and standard deviations represented in Figure 1(b). Figure 1(c) and 1(d) show the value of the Q-function for each action during a particular run. The first 100 episodes belong to the exploratory phase, in which the actions A1 and A2 obtain the highest reward, making the policy A1-A2 look particularly promising. An action is considered as *promising* as the highest value of the reward that choosing that action has ever given. In the case of A1-A2, its good result is due to the high variance, rather than the highest mean. This aspect will be addressed by the second phase of the algorithm.

The number of episodes in the exploratory phase should ideally allow for the sampling of each policy above its mean at least once. Depending on the particular strategy Σ and the shape of the distributions of the policies, such a number for *exp_length* might be computable. In practice, unless the problem is effectively hierarchically broken into much smaller problems, the number of episodes required is hardly feasible. In those cases, the exploration has to be shorter than what would be required to complete, and the algorithm will start with an upper bound for the limited number of policies visited, and keep exploring during the second phase.

If the domain does not allow for the estimation of a helpful upper bound, for instance because every action can potentially give high rewards, the first phase can be skipped initializing all actions optimistically. We conjecture that this may happen on synthetic domains in which the stochasticity is artificially injected, but it is rarer in real-world applications.

3.2 Assessment

We want to maximize the *expected* cumulative discounted reward, rather than the maximum obtainable one, therefore an evaluation of the *promising* policies is needed.

We rely on the main result behind stochastic search, reported (for minimization, but valid for maximization problems as well) in the following theorem [14]:

Theorem 1. *Suppose that θ^* is the unique minimizer of the loss function L on the domain Θ where $L(\theta^*) > -\infty$, $\theta_{new}(k)$ is the sample probability at iteration k, $\hat{\theta_k}$ is the best point found up to iteration k, and*

$$\inf_{\theta \in \Theta, \|\theta - \theta^*\| \geq \eta} L(\theta) > L(\theta^*)$$

for all $\eta > 0$. Suppose further that for any $\eta > 0$ and for all k there exists a function $\delta(\eta) > 0$ such that

$$Pr(\theta_{new}(k) : L(\theta_{new}(k)) < L(\theta^*) + \eta) > \delta(\eta)$$

Then, executing a random search with noise-free loss measurements, $\hat{\theta_k} \to \theta^*$ almost surely as $k \to \infty$.

In order to guarantee that the conditions expressed by the theorem are met we: (1) limit the search space to the *finite* set of deterministic stationary policies; (2) require immediate rewards to be bound; (3) require that the strategy Σ according to which the policies are picked assigns a non-zero probability to each of them. In particular, the second condition holds as the search space is finite, and the probability to land arbitrarily close to the optimal policy is non-zero, as such is the probability to land directly on it. If the optimal solutions are more than one, the algorithm might not converge on any single of them, but rather oscillate among optimal solutions. Since the value of optimal solutions

is, in general, not known in advance, there is no obvious stopping criterion. In practice, and as it is common in stochastic search, we may define a threshold above which we accept any solution.

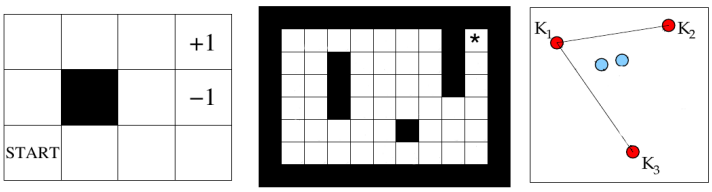
In the second phase the algorithm picks a policy according to Σ, evaluates it with first-visit Monte Carlo, and stops if the policy's reward is above a threshold. Monte Carlo methods wait until the end of the episode to update the action-value function, therefore the task needs to be episodic. The novel aspect of SoSMC is the way in which the search is biased. Reinforcement learning algorithms can traditionally be considered as composed by prediction and control. While the prediction part is borrowed from the literature (first-visit MC) the control part is based on the estimate of upper bounds, their storage in the action-value function, and their use to generate the next policy to try. Moreover, differently from MCESP, it performs a global search. It also employs a *consistent* exploration [3], that is, during the same episode, every time the agent perceives an observation it performs the same action.

If the reward is deterministic a single evaluation per policy is sufficient. Such a case may, for instance, occur on POMDPs in which the underlying MDP is deterministic and there is a single initial state. If the reward is stochastic, on the other hand, the capability to have an accurate estimate of the reward depends on the distribution. For some distributions it may be possible to compute confidence bounds and ensure that the reward returned by a policy is higher than the threshold with some probability. In general, the estimated mean cannot be guaranteed to be correct after any finite number of samples. In such cases, we use a fixed number of samples k which empirically proves to be reliable. Although losing some of the theoretical guarantees, we experimentally show how SoSMC can outperform the best results in the literature for different domains. While an inaccurate estimation of a policy may deceive the stopping criterion, if the threshold is not too tight on the optimal value a *good* policy is in practice always found in the domains we have used.

The example of Figure 1 shows an assessment phase with $k = 50$. Beginning with episode 101, in each state the action with the highest value is locally selected. The policy A1-A2 is initially executed for k episodes and the values of the actions converge to their means. Every k episodes a new policy is generated at random, with its probability depending on action values, and executed to be evaluated. Notice how in the second phase only half of the policies are actually sampled, as the action B1 is never executed after the initial phase.

3.3 Exploration strategies

Different choices are possible for the exploration strategy Σ, making SoSMC a family of algorithms. We have used both ϵ-greedy and SoftMax (cf. Equation 2). In the case of ϵ-greedy (where we refer to the algorithm as ϵ-SoSMC), the choice has been made for each state the first time it is encountered, and then remembered throughout the episode. Notice that this has not been applied to Sarsa, in which a separate decision is made if the same state is encountered multiple times in the same episode. The policies closest to the current optimal one



(a) (b) (c)

Fig. 2. The three domains used. (a) Parr and Russell's grid world, (b) Sutton's grid world, (c) Keepaway

are more likely to be selected, and become less and less probable as the distance from the optimal policy increases. As for SoftMax, again the current optimal policy is the most likely to be selected, but the neighbourhood is considered not just in the distance from such a policy, but also in the value of its actions, evaluated locally. SoSMC with SoftMax, referred to as Soft-SoSMC, performs particularly well in those domains in which the combinatorial aspect is minimal, and the choices can often be made separately.

4 Experimental Evaluation

In this section, we show the results of the application of our algorithm to three different domains. The first domain is the same one used for MCESP. In all of the domains, we have compared our algorithm with Sarsa as it is still the most effective method based on value functions on N-MDPs [9, 11]. As they are not based on value functions and search for a stochastic policy, rather than a deterministic one, we also leave policy gradient methods out of the evaluation.

4.1 Parr and Russell's Grid World

This small grid world has been used as a test domain by several authors [9, 11]. It has 11 states (Figure 2 (a)) in a 4 by 3 grid with one obstacle. The agent starts at the bottom left corner. There is a target state and a penalty state whose rewards are +1 and -1 respectively. Both are absorbing states, that is when the agent enters them the episode terminates. For each action that does not enter a target state, the agent receives a reward of -0.04. The actions available in every state are `move north`, `move south`, `move east`, and `move west` which succeed with probability 0.8. With probability 0.1 the agent moves in one of the directions orthogonal to the desired one. In all of the previous cases if the movement is prevented by an obstacle the agent stays put. In any state the agent can only observe the squares east and west of it, having a total of four possible observations. In order to make the task episodic, the maximum number of actions allowed is 100, after which the environment is reset.

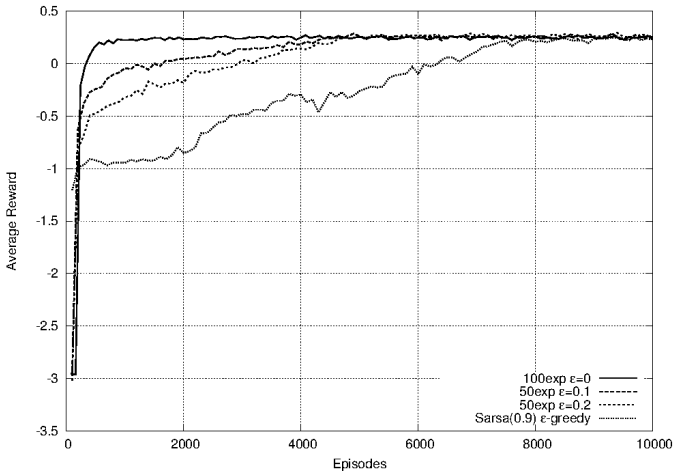


Fig. 3. Results for Parr and Russell's domain

Figure 3 shows the results of ϵ-SoSMC with an initial phase of 100 episodes and no exploration afterward (employing a constant $\alpha = 0.05$ and selecting the greedy policy at any time), with an initial phase of 50 episodes and exploration in the second phase ($\epsilon = 0.1$), and finally Sarsa(λ) with ϵ-greedy, in which ϵ starts at 0.2 and decreases to zero in 80000 actions as described by Loch and Singh [9]. The threshold has been posed close to the optimum at 0.2. Each point is the average of the reward collected by the agent in the previous 100 episodes, and over 100 runs, including the exploration. It can be noted how SoSMC converges much faster than Sarsa and reaches the optimal solution reliably.

4.2 Sutton's Grid World

Sutton's grid world is a 9 by 6 grid with several obstacles and a goal state in the top right corner. It is accessible to the agent only through its 8 neighboring states, making it a POMDP. Only 30 possibly observations are actually possible in the grid world, and the initial state is chosen at every episode uniformly at random. The actions are the same as the previous domain, but they are deterministic. For this reason, only those that do not run directly into an obstacle are available. In order to make the task episodic we set the maximum number of actions in any given episode to 20. After each action the agent receives a reward of -1, except for when it enters the goal state, in which case it is 0. Every 200 episodes we pause the learning and take a sample, from each initial state, of the current best policy, whose average reward per episode is plotted in Figure 4.

In this domain ϵ-SoSMC and Soft-SoSMC obtained similar results, therefore we only show Soft-SoSMC. We used SoftMax with no initial phase. The value function has been optimistically initialised and SoSMC launched from its second

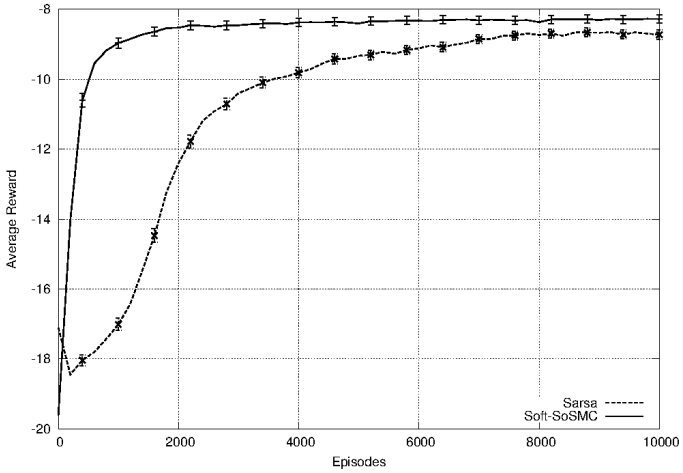


Fig. 4. Results for Sutton's domain

phase. In this experiment $\tau = 4$ and the algorithm explores every 20 episodes. The threshold is at -8.3. The results show how Soft-SoSMC finds, on average, a policy better than Sarsa. Sarsa obtains a value slightly, but statistically significantly smaller, as shown by the 95% confidence intervals which have been plotted on the graph every three points in order to not clutter the image.

4.3 Keepaway

Keepaway is a subtask of RoboCup Soccer in which one team, the *keepers*, must keep possession of the ball in a limited region as long as possible while another team, the *takers*, tries to gain possession. The task is episodic, and one episode ends whenever the takers manage to catch the ball or the ball leaves the region. We conducted our experiments on the 3 vs 2 task, i.e., with three keepers and two takers. Two different procedures are defined: `hold` and `pass(k)`. `Hold` keeps possession of the ball until the next decision can be made, while `pass(k)` passes the ball to the k-th team mate, where the team mates are sorted by their distance to the agent. A reward of 1/10 is collected every 1/10 of second, so that the agent maximizes the duration of the episode. We devise a representation with three variables: the two distances between the takers and the lines of pass towards the team mates, and the distance between the agent and the closest taker. Moreover, for each variable we consider only one threshold, having 8 observations in total. The simulator is provided with a policy, written by hand, meant as a benchmark. On our system, such a policy holds the ball for almost 16 seconds on average. We set the threshold for the stochastic search at 18 seconds. We also fix the behavior of two agents at a policy that waits for the takers to be withing a certain distance, and passes the ball to the team mate whose line of pass is

farther from the opponents. The third agent is the one that is going to learn. Figure 5 shows the mean hold time per episode of Soft-SoSMC and Sarsa(λ).

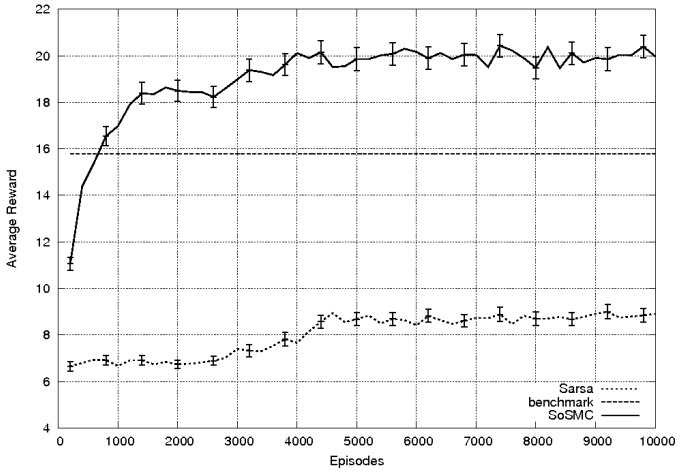


Fig. 5. Results for Keepaway. The average hold time per episode (y axis) is expressed in seconds

Each point is the average of the previous 200 episodes over 20 runs and *includes* the exploration (it is not just the best policy at any given time). Since this simulation is quite time consuming, we could not perform the same number of runs as the previous domains, in which we made sure that the confidence interval for each point was negligible. This plot is therefore less smooth, and as with the previous domain, we show the 95% confidence interval explicitly. The initial phase of Soft-SoSMC has a length of 100 episodes. While Sarsa(λ) only learns a behavior that on average holds the ball for 9 seconds, despite the small number of states, our algorithm reaches 16 seconds in less than 1000 episodes and goes up to 20s in the next 4000 episodes. This behavior outperforms several other methods on this domain, including policy search algorithms [18, 17, 4–6]. This is not just due to the small size of the representation, as Sarsa(λ) is not able to improve the agent's behavior of more than about 3 seconds. Note that the representation is certainly non-Markovian, as we have run Q-learning choosing actions at random, and instead of off-policy learning the optimal action-value function - as it is proved it would on an MDP - the value of all actions collapsed and where almost the same. In this N-MDP then, a direct RL method does not succeed in learning, and a specific algorithm like SoSMC can instead leverage the representation nonetheless.

5 Conclusions

In this paper, we have analyzed the advantages of using Non-Markovian processes in order to reduce the representation space and to achieve fast learning. Our work leads us to conclude that, in the domains analyzed in this paper, the attempt to enrich the domain description to make it Markovian can introduce high complexity for the learning process, such that algorithms that are proved to converge to optimal solutions in the long term do not provide any good results when the number of available samples is limited. On the other hand, smaller representations can indeed be more effective, since good solutions can be found within the limit of the available samples, with an adequate learning method.

References

1. P. Auer, T. Jaksch, and R. Ortner. Near-optimal regret bounds for reinforcement learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 89–96. 2009.
2. J. Baxter and P. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15(4):319–350, 2001.
3. P. A. Crook. Learning in a state of confusion: Employing active perception and reinforcement learning in partially observable worlds. Technical report, University of Edinburgh, 2006.
4. T. Jung and D. Polani. Learning robocup-keepaway with kernels. In *JMLR: Workshop and Conference Proceedings (Gaussian Processes in Practice)*, volume 1, pages 33–57, 2007.
5. S. Kalyanakrishnan and P. Stone. An empirical analysis of value function-based and policy search reinforcement learning. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '09, pages 749–756, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.
6. S. Kalyanakrishnan and P. Stone. Learning Complementary Multiagent Behaviors: A Case Study. In *Proceedings of the 13th RoboCup International Symposium*, pages 153–165, 2009.
7. M. Leonetti and L. Iocchi. Improving the performance of complex agent plans through reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 723–730, 2010.
8. M. L. Littman. Memoryless policies: Theoretical limitations and practical results. In *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 238–247, 1994.
9. J. Loch and S. Singh. Using eligibility traces to find the best memoryless policy in partially observable Markov decision processes. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 323–331, 1998.
10. M. D. Pendrith and M. McGarity. An analysis of direct reinforcement learning in non-markovian domains. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, pages 421–429, 1998.
11. T. J. Perkins. Reinforcement learning for POMDPs based on action values and stochastic optimization. In *Proceedings of the National Conference on Artificial Intelligence*, pages 199–204, 2002.

12. T. J. Perkins and M. D. Pendrith. On the existence of fixed points for Q-learning and Sarsa in partially observable domains. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 490–497, 2002.
13. S. Singh and R. Sutton. Reinforcement learning with replacing eligibility traces. *Recent Advances in Reinforcement Learning*, pages 123–158, 1996.
14. J. C. Spall. *Introduction to Stochastic Search and Optimization.* John Wiley & Sons, Inc., New York, NY, USA, 1 edition, 2003.
15. P. Stone, R. S. Sutton, and G. Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.
16. R. Sutton and A. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.
17. M. E. Taylor, S. Whiteson, and P. Stone. Transfer via inter-task mappings in policy search reinforcement learning. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, AAMAS '07, pages 37:1–37:8, New York, NY, USA, 2007. ACM.
18. S. Whiteson, N. Kohl, R. Miikkulainen, and P. Stone. Evolving soccer keepaway players through task decomposition. *Machine Learning*, 59:5–30, 2005. 10.1007/s10994-005-0460-9.
19. H. Young. *Strategic learning and its limits.* Oxford University Press, USA, 2004.