JMLR: Workshop and Conference Proceedings vol:1–13, 2012 European Workshop on Reinforcement Learning

Direct Policy Search Reinforcement Learning based on Particle Filtering

Petar Kormushev PETAR.KORMUSHEV@IIT.IT
Darwin G. Caldwell DARWIN.CALDWELL@IIT.IT
Department of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego 30, 16163 Genova

Abstract

We reveal a link between particle filtering methods and direct policy search reinforcement learning, and propose a novel reinforcement learning algorithm, based heavily on ideas borrowed from particle filters. A major advantage of the proposed algorithm is its ability to perform global search in policy space and thus find the globally optimal policy. We validate the approach on one- and two-dimensional problems with multiple optima, and compare its performance to a global random sampling method, and a state-of-the-art Expectation-Maximization based reinforcement learning algorithm.

Keywords: reinforcement learning, particle filter, global search, parameterized policy

1. Introduction

Reinforcement learning (RL) is a machine learning approach, in which the goal is to find a policy π that maximizes the expected future return, calculated based on a scalar reward function $R(\cdot) \in \mathbb{R}$. The argument of $R(\cdot)$ can be defined in different ways, e.g. it could be a state s, or a state transition, or a state-action pair, or a whole trial as in the case of episodic RL, etc. The policy π determines what actions will be performed by the RL agent, and is usually state dependent. Sutton and Barto (1998)

Originally, the RL problem was formulated in terms of a Markov Decision Process (MDP) or Partially Observable MDP (POMDP). In this formulation, the policy π is viewed as a direct mapping function ($\pi : s \longmapsto a$) from state $s \in S$ to action $a \in A$.

Alternatively, instead of trying to learn the explicit mapping from states to actions, it is possible to perform *direct policy search*, as shown in Rosenstein and Barto (2001). In this case, the policy π is considered to depend on some parameters $\theta \in \mathbb{R}^N$, and is written as a parameterized function $\pi(\theta)$. The episodic reward function becomes $R(\tau(\pi(\theta)))$, where τ is a trial performed by following the policy. The reward can be abbreviated as $R(\tau(\theta))$ or even as $R(\theta)$, which reflects the idea that the behaviour of the RL agent can be influenced by only changing the values of the policy parameters θ. Therefore, the outcome of the behaviour, which is represented by the reward $R(\theta)$, can be optimized by only optimizing the values θ. This way, the RL problem is transformed into a black-box optimization problem with cost function $R(\theta)$, as shown in Rückstieß et al. (2010) under the name *parameter-based exploration*.

However, it is infeasible to use conventional numeric optimization methods to maximize $R(\theta)$ if we want to apply RL to real-world problems, because the cost function is usually expensive to evaluate. For example, each cost function evaluation requires conducting at least one trial which could involve costly real-world experiments or computationally expensive simulations. Therefore, alternative optimization methods are desired, which are tuned to the specific need of RL to reduce as much as possible the number of reward function evaluations (trials).

This paper aims to bring new ideas into the domain of RL, borrowed from statistics. We propose a novel direct policy search RL algorithm which provides an alternative solution to the RL problem, and new possibilities for RL algorithms in general. Before we can introduce our proposed approach, we need to place it in the proper context, which is done in the following Sections 2 and 3.

2. State-of-the-art RL algorithms for Direct Policy Search

This section contains a non-exhaustive list of direct policy search RL approaches. We focus on policy search methods that attempt to minimize the number of trials. Examples for such RL methods include:

- Policy Gradient based RL - in which the RL algorithm is trying to estimate the gradient of the policy with respect to the policy parameters, and to perform gradient descent in policy space. The Episodic Natural Actor-Critic (eNAC), in Peters and Schaal (2008), and Episodic REINFORCE in Williams (1992) are two of the well-established approaches of this type.

- Expectation-Maximization based RL - in which the EM algorithm is used to derive an update rule for the policy parameters at each step, trying to maximize the lower bound on the expected return of the policy. A state-of-the-art RL algorithm of this type is PoWER (Policy learning by Weighting Exploration with the Returns), in Kober and Peters (2009), as well as its generalization MCEM, in Vlassis et al. (2009).

- Path Integral based RL - in which the learning of the policy parameters is based on the framework of stochastic optimal control with path integrals. A state-of-the-art RL algorithm of this type is PIˆ2 (Policy Improvement with Path Integrals), in Theodorou et al. (2010).

- Regression based RL - in which regression is used to calculate updates to the RL policy parameters using the rewards as weights. One of the approaches that are used extensively is LWPR (Locally Weighted Projection Regression), in Vijayakumar and Schaal (2000).

- Model-based policy search RL - in which a model of the transition dynamics is learned and used for long-term planning. Policy gradients are computed analytically for policy improvement using approximate inference. A state-of-the-art RL algorithm of this type is PILCO (Probabilistic Inference for Learning COntrol), in Deisenroth and Rasmussen (2011).

A major problem with all these existing approaches is that they only perform *local search*, therefore they do not guarantee convergence to the global optimum. Instead, all of them tend to converge to some local sub-optimal solution.

Another major problem is that the result from each method is largely influenced by the initial value of θ. The reason for this is that all of these methods have the inherent notion of 'current policy'. They work by changing this current policy iteratively with a small amount every time using an update rule in the following generic form:

$$\theta_{n+1} = \theta_n + \Delta\theta, \tag{1}$$

where θ_n is the current policy and θ_{n+1} is the new policy. The term $\Delta\theta$ is calculated in different ways, either by some form of gradient estimation (as in gradient-based RL algorithms and PILCO), or using other calculation method (e.g. EM based, path integral based or regression-based). Regardless of the exact way, these approaches are, by definition, local search methods, and they can only guarantee convergence to a local optima.

In contrast, what we propose in this paper is a *global search* method for direct policy search reinforcement learning which is guaranteed not to get stuck at local optima. The principle for realizing this comes from another research domain: *particle filtering*.

3. Particle filters

Particle filters, also known as Sequential Monte Carlo methods Doucet et al. (2000, 2001), originally come from statistics and are similar to importance sampling methods. Particle filters are able to approximate any probability density function, and can be viewed as a 'sequential analogue' of Markov chain Monte Carlo (MCMC) batch methods.

Although particle filters are mainly used in statistics, there are a few other research areas in which particle filters have found application. For example, in the domain of probabilistic robotics Thrun (2002), particle filters are extensively and successfully used, e.g. for performing Monte Carlo localization of mobile robots with respect to a global map of the terrain Fox et al. (1999); Kwok et al. (2003), and also for Simultaneous Localization and Mapping (SLAM) tasks Montemerlo et al. (2003); Sim et al. (2005); Howard (2006).

The potential of applying particle filters in the RL domain appears to be largely unexplored so far. To the best of our knowledge, there are only two partial attempts to apply particle filters in RL in the existing published work, done by Notsu et al and Samejima et al, respectively, as follows.

In Notsu et al. (2011a), they studied traditional RL with discrete state and action spaces. They used the Actor-Critic method for performing value iteration, with state value updates using the TD-error. In this conventional framework, they proposed a particle filter for segmentation of the action space. Their goal was to do what they called 'domain reduction', or trying to minimize the number of discrete actions available at each state, by dividing the action space in segments. Then, in Notsu et al. (2011b), they extended the same idea to traditional continuous RL and used Q-learning with function approximation. They used a similar particle filter approach for segmenting the continuous state and action spaces into discrete sets by particles, and applied it to inverted pendulum tasks.

In Samejima et al. (2004), they studied neurophysiology and created a reinforcement learning model of an animal behaviour. Their goal was to predict the behaviour of a monkey during an experimental task. They used traditional Q-learning, and built a Bayesian network representation of the Q-learning agent. In this framework, particle filtering was used to estimate action probability in order to predict the animal behaviour.

In both of these existing approaches, particle filters were used in a limited way, as a technique to solve some partial problem within a traditional RL framework.

In this paper, we propose a rather different approach. First, we propose a completely new view of the link between particle filters and RL. Then, we propose an entirely novel RL algorithm for direct global policy search, based on particle filters as the core for the RL algorithm itself. In our framework, the search is performed in the policy space defined by the selected policy parameterization, and the process is viewed as a black-box optimization. The particle filter itself is the core of the proposed RL algorithm, and is responsible for guiding the exploration and exploitation, by creating particles, each of which represents a whole policy. Details of the proposed novel view and algorithm follow.

4. Novel view of RL and its link to particle filters

The key to linking particle filters and RL is to make the following observation. The landscape, defined by the reward function $R(\theta) \in \mathbb{R}$ over the whole continuous domain of the parameter space $\theta \in \Theta$, can be viewed as defining an improper probability density function (IPDF)[1]. This is possible even if the reward function $R(\theta)$ has negative values in its range, because we can simply add a constant positive number $L = |\inf_{\theta \in \Theta} R(\theta)|$ to it, and obtain a new reward function $R'(\theta)$ which is non-negative and has exactly the same set of optimizers $\theta^* \in \Theta$ as the original one. Therefore, optimizing $R'(\theta)$ will also optimize $R(\theta)$.

Once we make the assumption that $R(\theta)$ is just an IPDF, then the RL problem can be reformulated from a new point of view. Each trial $\tau(\pi(\theta))$ can be viewed as an independent sample from this unknown IPDF. The RL algorithm can be viewed as a method for choosing a finite number of sampling points for which to obtain the value of the IPDF. Finally, the RL problem can be viewed as the problem of finding the mode (or all modes, in the multi-modal case) of the unknown IPDF, given only a finite number of sampling points with their corresponding values of the IPDF, obtained by the RL algorithm.

This view of RL immediately opens the path for applying particle filters, because they are a method for approximate estimation of an unknown PDF based on a finite number of samples. To complete the link between RL and particle filters, the only thing left to state is that it is trivial to convert an IPDF into a PDF simply by normalizing it (dividing by the integral of it).

5. RL based on particle filters

Using the reformulation of RL from the previous section, here we propose a novel RL algorithm based on Particle Filters (RLPF). The main idea of RLPF is to use particle filtering as a method for choosing the sampling points, i.e. for calculating a parameter vector θ for each trial.

We define a *policy particle* p_i to be the tuple $p_i = \langle \theta_i, \tau_i, R_i, w_i \rangle$, where the particle p_i represents the outcome of a single trial τ_i performed by executing an RL policy $\pi(\theta_i)$, where θ_i is a vector of policy parameter values modulating the behaviour of the RL policy π. The policy particle also stores the value of the reward function evaluated for this trial $R_i = R(\tau_i(\pi(\theta_i)))$. The variable τ_i contains task-specific information recorded during the trial depending on the nature of the task. The information in τ_i is used by the reward

1. IPDF is similar to PDF except that the integral of it does not have to be equal to one.

function to perform its evaluation. The variable w_i is the importance weight of this policy particle, and the way of its calculation is explained below.

Following closely the ideas from particle filters, we make the assumption that the set of particles $\{p_i\}$ is an approximate implicit representation of the underlying unknown IPDF defined by $R(\theta)$. Therefore, in order to select a new particle which obeys the real IPDF distribution, what we can do is to sample from the approximate distribution while correcting for the discrepancy between it and the real distribution. The mechanism for this correction is provided by the importance weights $\{w_i\}$.

Firstly, each policy particle p_i is assigned a scalar importance weight w_i derived from its corresponding reward R_i using a transformation function g, such that: $w_i \propto g(R_i)$. In the simplest case, $g(\cdot)$ could be the identity, but in the general case, it could be an arbitrary non-negative function. We apply the function g in such a way, that the importance weights are normalized, in the sense that: $\forall w_i \quad 0 < w_i < 1$, and also: $\sum w_i = 1$.

Secondly, we construct an auxiliary function $h(u) = \int_{-\infty}^{u} w_u du$, which in our discrete case takes the form $h(k) = \sum_{j=1}^{k} w_j$. This function can be thought of as the (approximate) cumulative density function (CDF) of the unknown PDF. Indeed, due to the way we create the importance weights, it follows directly that $\int_{-\infty}^{+\infty} w_u du = 1$, and thus $h(u)$ is a proper CDF. This is important because, given that $w_i > 0$, it guarantees that $h(u)$ is strictly monotonically increasing and therefore the inverse function h^{-1} exists.

Thirdly, we introduce a random variable z which is uniformly distributed in the interval $(0, 1)$. Now, it can be shown that the random variable y defined as $y = h^{-1}(z)$ is distributed (approximately) according to the desired unknown PDF, see e.g. Bishop (2006).

At this point, there are two variants of particle filters, which accordingly results in two variants for RLPF, and they are: Sequential Importance Sampling (SIS), and Sequential Importance Resampling (SIR). Detailed discussion of the differences between the two is beyond the scope of this paper, and can be found in Gordon et al. (1993); Douc and Cappé (2005). For completeness, we note that the proposed RLPF algorithm is modeled after the SIS variant. However, it is possible to transform RLPF into the SIR variant, by adding an additional resampling step. In SIR, resampling is used in order to avoid the problem of degeneracy of the algorithm, that is, avoiding the situation that all but one of the importance weights are close to zero. The performance of the algorithm can be also affected by proper choice of resampling method.

The pseudo-code for RLPF is given in Algorithm 1. A detailed description of the algorithm follows.

In lines 3-5, at every iteration of the main loop, a choice is made between exploration (which performs global random sampling in the given policy space) and exploitation (which uses particle filtering as a sampling mechanism to select new policy parameters). This choice is controlled by a user-defined function $P_{explore}(n)$, which defines the probability of doing exploration at iteration number $n \in [1, N]$ of the RLPF algorithm. This mechanism allows the user to directly control the exploration/exploitation trade-off. A good practice is, to give high probability of exploration at the beginning, and then reduce it to minimum in order to give preference to exploitation, thus focusing on the most promising areas in the policy space. In the degenerate case when $\forall n \, P_{explore}(n) = 1$, the RLPF algorithm is reduced to Global Random Sampling (GRS).

In lines 9-20, the main particle filtering mechanism is implemented. In lines 11-14, the importance weights of the policy particles are calculated. In lines 15-18, a particle is selected based on the inverse density function mechanism, described earlier in this section. In lines 19-20, a new particle is selected by adding exponentially decayed noise to the previously selected particle.

In lines 22-23, the new policy particle is evaluated based on one or more trials. In the deterministic case, a single policy evaluation is enough to evaluate each policy particle. In the non-deterministic (stochastic) case, multiple evaluations of the policy particle can be performed, and the average obtained return can be used as an unbiased estimate of the expected return from the policy.

Algorithm 1 Reinforcement Learning based on Particle Filters (RLPF)

1: **Input:** parameterized policy π, policy parameter space Θ, reward function $R(\theta)$ where $\theta \in \Theta$, reward transformation function g, total number of trials N, exploration probability function $P_{explore}(n)$, initial noise ϵ_0, noise decay factor λ.

2: Let $S = \{\,\}$ $\quad$ {A set of policy particles}
3: **for** $n = 1$ **to** N **do**
4:   Draw $v \sim U(0,1)$ $\quad$ {Uniform sampling $\in$ (0,1)}
5:   **if** $v < P_{explore}(n)$ **then**
6:     {Exploration step: global random sampling}
7:     Draw $\theta_n \sim U(\Theta)$ $\quad$ {Uniform sampling}
8:   **else**
9:     {Exploitation step: particle filter sampling}
10:     Let $h(0) = 0$
11:     **for** $i = 1$ **to** $|S|$ **do**
12:       $w_i = \frac{g(R_i)}{\sum_{j=1}^{|S|} g(R_j)}$ $\quad$ {Importance weight}
13:       $h(i) = h(i-1) + w_i$ $\quad$ {Aux. function}
14:     **end for**
15:     Draw $z \sim U(0,1)$ $\quad$ {Uniform sampling}
16:     Let $y = h^{-1}(z)$
17:     Let $k = \lceil y \rceil$ $\quad$ {$\lceil \cdot \rceil$ is the ceiling function}
18:     Select policy particle $p_k = \langle \theta_k, \tau_k, R_k, w_k \rangle$
19:     Let $\epsilon_n = \epsilon_0 \lambda^{(n-L-1)}$ $\quad$ {noise with exp. decay}
20:     Let $\theta_n = \theta_k + \epsilon_n$
21:   **end if**
22:   Perform trial(s) $\tau_n(\pi(\theta_n))$
23:   Estimate return R_n
24:   Let $w_n = 0$
25:   Create new policy particle $p_n = \langle \theta_n, \tau_n, R_n, w_n \rangle$
26:   $S = S \cup \{p_n\}$
27: **end for**

6. Analysis of RLPF

RLPF inherits many advantages from particle filters, among which:

- It is very simple to implement and can be realized easily in embedded systems for online learning;
- It is not computationally expensive and has very low memory footprint;
- It can use adaptive computation depending on the available resources (both time- and CPU-wise) by changing the value of the σ parameter;
- It can concentrate the effort of the RL exploration on the most important parts of the policy space, by using function $g(R)$ which increases the relative difference between the rewards, e.g. the function $g(R) = (1 + R)^2$;
- It can exhibit adaptive convergence rate depending on the requirements for precision and time, by changing the initial noise level ϵ_0 and the decay factor λ.

To be fair, we should also mention some disadvantages of RLPF. For example, as a member of global search methods, it generally requires more trials in order to converge, because the scope of the search is the largest possible - the whole policy space. However, using appropriate reward transformation functions, it is possible to bias the exploration towards the highest-reward particles, sacrificing thoroughness for convergence speed.

Another disadvantage is that, particle filters do not have a strict proof of convergence. In theory, we have traded a 'proof for local convergence' with a global search method which has no proof of global convergence, but is at least guaranteed not to get stuck at local optima. In practice, as the experience from particle filters has demonstrated, even without such strict proof of convergence it is possible to obtain excellent results in real-world applications. Especially for tasks where it is important to find all possible ways (or at least all classes) of solving a problem, global search is the only way to go and is more important than having a proof for convergence to a single local optimum.

7. Experimental evaluation of RLPF

First, we evaluate RLPF standalone on a one-dimensional problem, because it is the easiest to visualize and analyze. Figure 1 shows an example run of RLPF on a class of synthetic 1D reward functions with many local optima. It is clearly visible that the generated policy particles by RLPF tend to cluster around the highest 'peaks' in the reward function.

Second, we compare the performance of RLPF with other global policy search RL methods. It is difficult to select baselines against which to compare RLPF to, because there are not any truly global search policy-based RL algorithms. It would not be fair to compare a local search RL, such as policy gradient based RL, to RLPF, because the local search methods will easily get stuck at the local optima. So, instead, for a baseline we use a stochastic global policy search RL algorithm, which is based on Global Random Sampling (GRS) in policy space. The comparison with RLPF is shown in Figure 2, averaged over many runs, and for the same problem as in Figure 1.

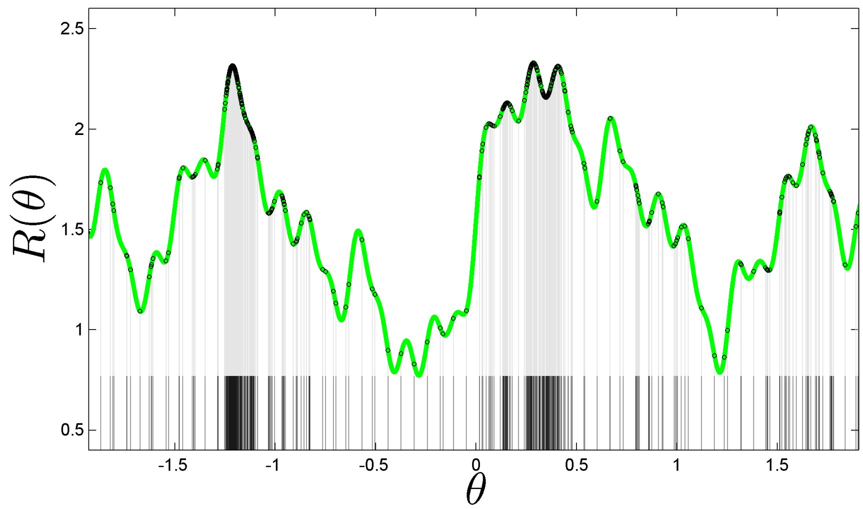


Figure 1: An illustration of a typical run of RLPF on a 1-dimensional problem. The generated policy particles by RLPF are shown with vertical grey stripes. The corresponding reward values are shown with black circles on top of the reward function line, shown in green. The following synthetic reward function was used, because it has many local optima: $R(\theta) = 1.55 + 0.3\sin(2.7\theta) + 0.4\sin(4.5\theta) + 0.2\sin(8.7\theta) + 0.14\sin(20\theta) + 0.08\sin(30\theta) + 0.08\sin(50\theta)$.

Third, we evaluate RLPF on a two-dimensional problem, designed on purpose to visually demonstrate the efficiency of RLPF in policy space exploration. The experiment is shown in Figure 3. It illustrates one of the most important advantages of the proposed global-search based RLPF algorithm over existing local search based RL methods, which is the ability to simultaneously find multiple alternative optimal policies, even if they are far apart from each other in policy space. This advantage is most obvious for problems which have many optimal policies, spread throughout the policy space and grouped in distinct clusters. While existing local search based RL methods tend to find only a single cluster of optimal policies, and are strongly influenced by the initial policy, the proposed RLPF algorithm is able to find simultaneously multiple optimal policies belonging to different clusters, and is not influenced by any initial policy. This advantage of RLPF is clearly visible in Figure 3(a), showing each explored policy particle by RLPF with a blue dot. The reward function for this 2D problem is designed to have multiple optima, and is defined as:

$$R(\theta) = 1000 - \left(\min(|\theta_1 + 25|, |\theta_1 - 25|)\right)^2 . \tag{2}$$

The optimal policies lie on two vertical lines ($\theta_1 = \pm 25$) in policy space, and are illustrated with two vertical red lines in Figure 3(a). In order to quantitatively analyze the exploration performance of RLPF, we divide the 2D policy space in separate tiles on a uniform 100-by-100 grid. We call each tile, which contains at least one policy particle explored by RLPF, an ‘alternative’. By counting the number of alternatives explored by RLPF, and the subset of them which have rewards above a certain threshold, we can quantitatively

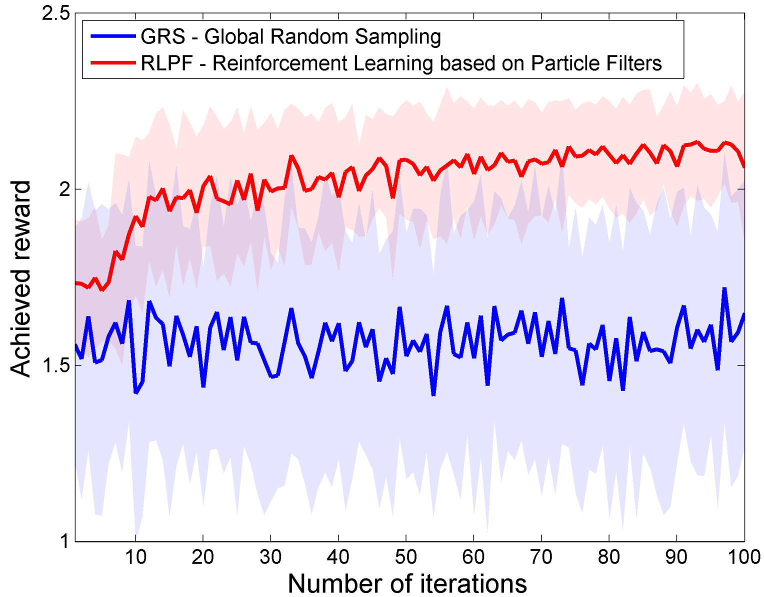
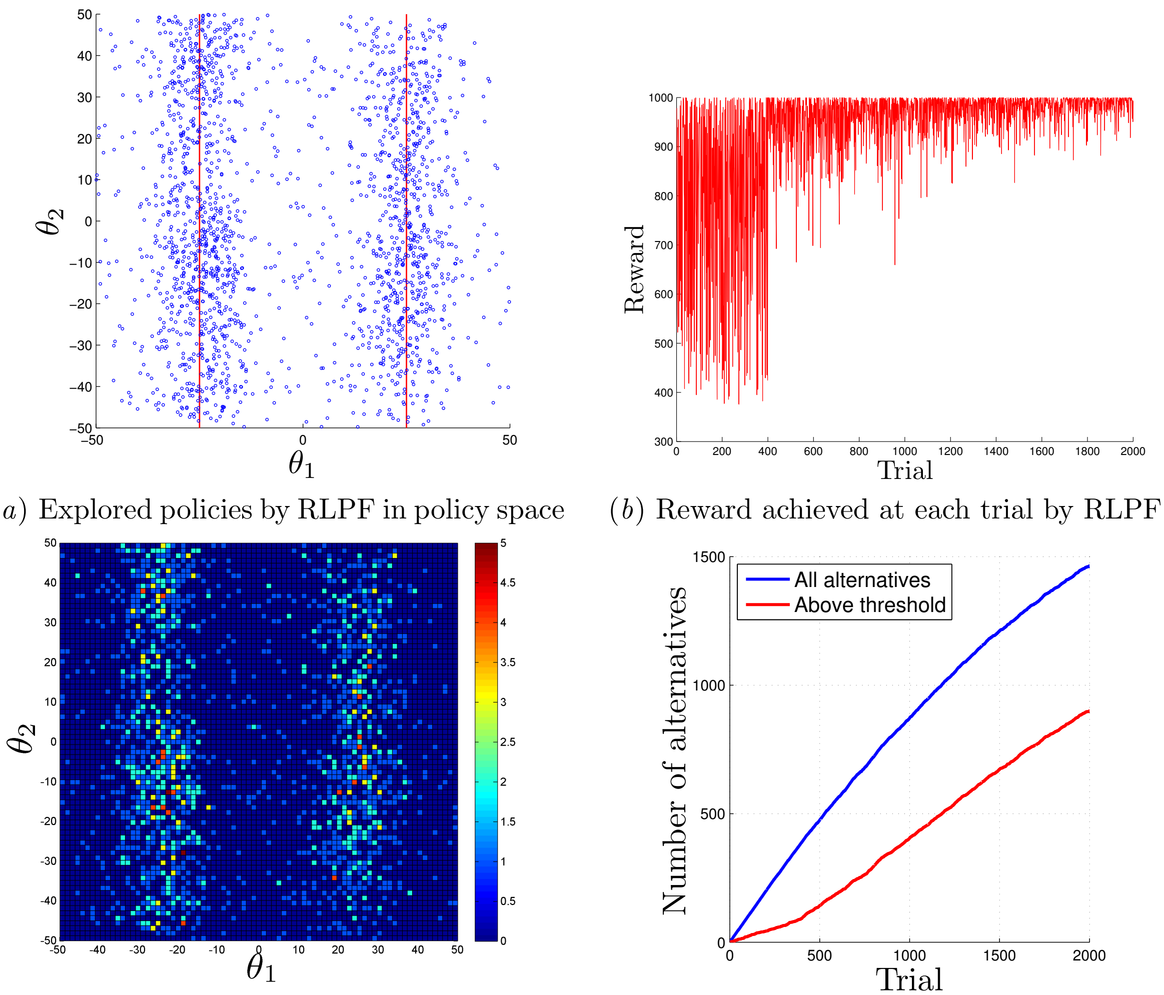


Figure 2: A comparison of the convergence of two global policy search RL algorithms: Global Random Sampling (GRS) policy search RL algorithm, and RLPF. The problem is the same as the one in Figure 1. The results are averaged over 50 runs of each algorithm. Every run has 100 trials. RLPF easily outperforms GRS both in terms of achieved reward and low level of variance.

compare the performance of RLPF to other RL methods. As shown in Figure 3(b), RLPF is run for a total of 2000 trials. The function $P_{explore}(n)$ is defined in such a way that the first 400 trials are exploratory global random trials, and the rest are performed by the particle filter which redistributes the exploration evenly around the most promising policy space regions (the ones with highest rewards). Figure 3(c) shows a 2D histogram, marking the discovered alternatives, and counting the policy particles in each corresponding tile. It clearly shows that RLPF manages to distribute well the exploration effort, focusing on the most promising alternatives. Figure 3(d) plots the temporal evolution of the number of alternatives discovered, versus the number of trials. Out of 2000 total trials, RLPF manages to find almost 1500 alternatives, with around 900 of them above the threshold, which is equal to 95% of the maximum reward.

Fourth, we compare RLPF to a state-of-the-art policy-search RL algorithm. We chose the PoWER algorithm, because of its fast convergence, and small number of open parameters that need to be tuned. Since PoWER is a local search RL method, in order to be fair in the comparison, we run it multiple times starting from a random initial policy every time. Figure 4 shows the results from running PoWER on the same 2D problem as in the previous experiment with RLPF. PoWER is run in batch mode 25 times, each time for 80 trials, adding up to a total of 2000 trials (same as RLPF). Each run is started from a random initial policy, shown with green circles in Figure 4(a). The comparison between Figure 3 and Figure 4 reveals the significant advantage of RLPF over PoWER both in terms of exploration efficiency, and speed of discovery of alternative optimal policies. PoWER manages to find only less than 1000 alternatives in total, with around 700 of them above the threshold. The reason for the low performance of PoWER is the less efficient exploration, caused by redundant policy space exploration (visible from the 2D histogram in Figure 4(c)), and by local policy search which only converges to a single optimal policy.



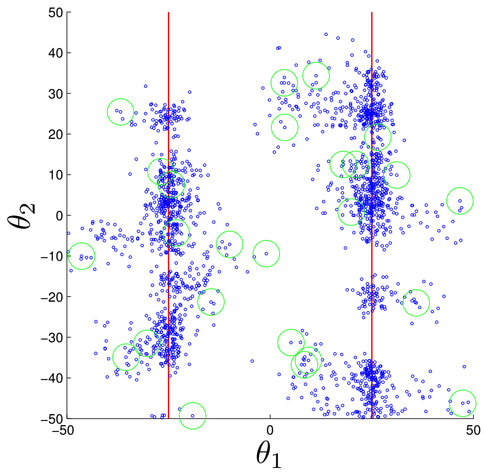
(a) Explored policies by RLPF in policy space

(b) Reward achieved at each trial by RLPF

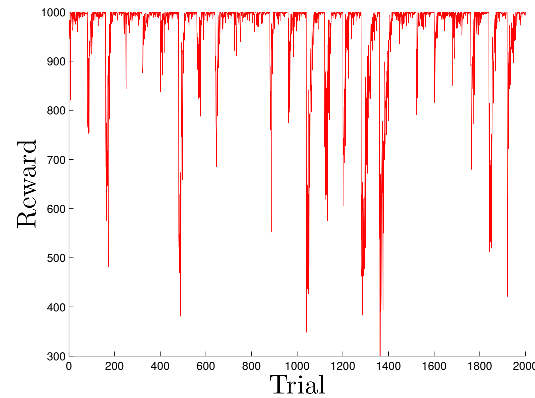
(c) 2D histogram of alternatives discovered by RLPF

(d) Number of alternatives found vs. trials

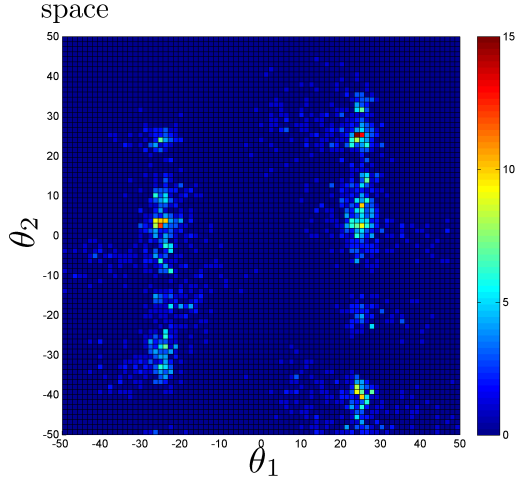
Figure 3: Evaluation of the proposed RLPF algorithm on a 2D problem which has multiple optimal policies.



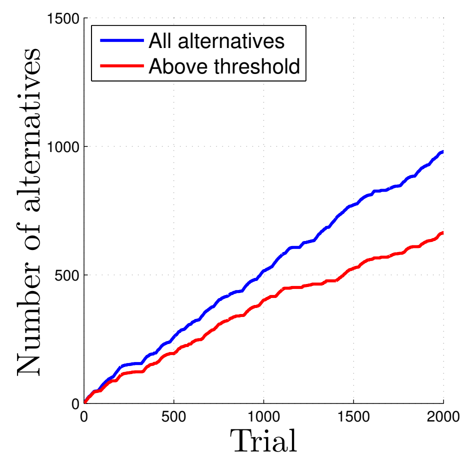
(a) Explored policies by PoWER in policy space



(b) Reward achieved at each trial by PoWER



(c) 2D histogram of alternatives discovered by PoWER



(d) Number of alternatives found vs. trials

Figure 4: Comparative evaluation of PoWER algorithm on the same 2D problem as in Figure 3.

8. Conclusion

This paper introduced ideas from particle filtering and importance sampling into the domain of reinforcement learning. We revealed a link between particle filters and RL, and reformulated the RL problem from this perspective. We proposed a new RL algorithm which is based on particle filters at its core. Due to the ability of particle filters to perform global search, the resulting RL algorithm is also capable of direct global search in policy space, which is significant improvement over traditional local search based policy RL. Since this work opens up a novel research direction in RL, there are many ways in which it can be extended in the future.

References

C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer New York, 2006. (pages 644–646).

Marc P. Deisenroth and Carl E. Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, June 2011.

R. Douc and O. Cappé. Comparison of resampling schemes for particle filtering. In *International Symposium on Image and Signal Processing and Analysis (ISPA 2005)*, pages 64–69. IEEE, 2005.

A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

A. Doucet, N. De Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice.* Springer Verlag, 2001.

D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the National Conference on Artificial Intelligence*, pages 343–349. John Wiley & sons Ltd, 1999.

N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *Radar and Signal Processing, IEE Proceedings F*, volume 140, pages 107–113. IET, 1993.

A. Howard. Multi-robot simultaneous localization and mapping using particle filters. *The International Journal of Robotics Research*, 25(12):1243–1256, 2006.

J. Kober and J. Peters. Learning motor primitives for robotics. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 2112–2118, May 2009.

C. Kwok, D. Fox, and M. Meila. Adaptive real-time particle filters for robot localization. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 2836–2841. IEEE, 2003.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 1151–1156, 2003.

A. Notsu, K. Honda, and H. Ichihashi. Proposed particle-filtering method for reinforcement learning. In *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*, pages 1755–1718. IEEE, 2011a.

A. Notsu, K. Honda, H. Ichihashi, Y. Komori, and Y. Iwamoto. Improvement of particle filter for reinforcement learning. *International Conference on Machine Learning and Applications*, 1:454–457, 2011b.

J. Peters and S. Schaal. Natural actor-critic. *Neurocomput.*, 71(7-9):1180–1190, 2008.

M.T. Rosenstein and A.G. Barto. Robot weightlifting by direct policy search. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 839–846. Citeseer, 2001.

T. Rückstieß, F. Sehnke, T. Schaul, D. Wierstra, Y. Sun, and J. Schmidhuber. Exploring parameter space in reinforcement learning. *Paladyn. Journal of Behavioral Robotics*, 1 (1):14–24, 2010.

K. Samejima, K. Doya, Y. Ueda, and M. Kumura. Estimating internal variables and parameters of a learning agent by a particle filter. In *Advances in Neural Information Processing Systems*, volume 16, pages 1335–1342. NIPS, 2004.

R. Sim, P. Elinas, M. Griffin, and J.J. Little. Vision-based slam using the rao-blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pages 9–16, 2005.

R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction.* Adaptive computation and machine learning. MIT Press, Cambridge, MA, USA, 1998.

Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A Generalized Path Integral Control Approach to Reinforcement Learning. *The Journal of Machine Learning Research*, 11: 3137–3181, December 2010. ISSN 1532-4435.

S. Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

S. Vijayakumar and S. Schaal. Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional spaces. In *Proc. Intl Conf. on Machine Learning (ICML)*, pages 288–293, Haifa, Israel, 2000.

N. Vlassis, M. Toussaint, G. Kontes, and S. Piperidis. Learning model-free robot control by a Monte Carlo EM algorithm. *Autonomous Robots*, 27(2):123–130, 2009.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256, 1992.