Policy search for the optimal control of Markov decision processes: a novel particle-based iterative scheme

Giorgio Manganini, Matteo Pirotta, Marcello Restelli, Luigi Piroddi and Maria Prandini

***Abstract*—Classical approximate dynamic programming techniques based on state space gridding become computationally impracticable for high–dimensional problems. Policy search techniques cope with this curse of dimensionality issue by searching for the optimal control policy in a restricted parameterized policy space. We here focus on the case of discrete action space and introduce a novel policy parametrization that adopts particles to describe the map from the state space to the action space, each particle representing a region of the state space that is mapped into a certain action. The locations and actions associated to the particles describing a policy can be tuned by means of a recently introduced policy gradient method with parameter-based exploration. The task of selecting an appropriately sized set of particles is here solved through an iterative policy building scheme, that adds new particles to improve the policy performance and is also capable of removing redundant particles. Experiments demonstrate the scalability of the proposed approach as the dimensionality of the state space grows.**

***Index Terms*—Markov decision processes; Stochastic optimal control; Approximate dynamic programming; Reinforcement learning; Policy search.**

I. Introduction

In this paper, we address the optimal control of large scale stochastic systems. Stochastic optimal control problems arise in several application domains such as communication networks, manufacturing systems, air traffic management, and power networks. These problems are particularly challenging when the stochastic dynamics of the system is hybrid [1], [2], *i.e.*, it is characterized by a tight coupling between continuous and discrete dynamics. Furthermore, the computational complexity of the problem increases rapidly with the system size, motivating the quest for efficient algorithms.

We consider systems that can be modeled as discrete time Markov Decision Processes (MDPs, see e.g. [3]) with a continuous state space component and a finite control space. MDPs are a powerful modeling framework for addressing problems involving sequential decision making under uncertainty – even in the presence of hybrid dynamics as in stochastic hybrid systems. An MDP is a probabilistic dynamic model where the state evolution is governed by transition probabilities that depend on the control input. A control policy is a rule to determine which control action to apply given the current state of the system. Such a rule can be either deterministic (a function mapping each state into a control action) or stochastic (a function mapping each state into a probability distribution over the control input space). If the objective is to maximize some additive reward function along a future time horizon, an optimal control policy can be characterized through the Dynamic Programming (DP) approach [4]. In the DP framework the maximum expected return that can be obtained starting from any given state is expressed by the optimal value function. This function is calculated by solving the Bellman equation [5], which takes into account the immediate effect of the decisions taken at each stage (through some instantaneous reward), as well as their expected future impact over the residual look-ahead time horizon. In turn, an optimal policy can be computed based on the optimal value function. However, applying DP becomes problematic with large or infinite state spaces, or when the system dynamics are not explicitly modeled. In such cases one may resort to approximate techniques that rely on the simulation of the actual system, and approximate parametric representations of either the value function or the policy. A wide variety of methods have been developed along this line, giving rise to the fields of Approximate/Adaptive Dynamic Programming (ADP) [6], [4], [7], and Reinforcement Learning (RL) [5], [8]. ADP methods typically consist of an iterative scheme for successively improving the quality of the approximation of the optimal value function (value iteration methods, see, *e.g.*, [9], [10]), the optimal control policy itself (policy iteration methods, see, *e.g.* [11], [12], [42]), or both of them (actor–critic methods, see, *e.g.* [13], [14], [15]). RL approaches approximate the expected values involved in the value function computation and in the policy evaluation using empirical averages over data, either taken from available time histories or obtained through the direct interaction between the learning algorithm and the system to be controlled. Such approaches allow to learn optimal or near-optimal policies, even when the system dynamics are unknown (see, *e.g.*, [16], [41]).

Many recent works in RL have focused on algorithms that search directly through a space of parameterized policies (policy search methods) [17]. These are indeed among the most effective learning algorithms for stochastic control problems with continuous state and action spaces, in view of the following aspects: i) policy parameterizations can be chosen according to the task and the exploration can be directly controlled, ii) the policy can be parameterized in a much more compact way than the value function, and iii) tuning algorithms with guaranteed convergence properties are available. Policy search methods have been successfully applied to several real-world tasks [19], [20], [21], [22].

A policy search method is characterized by a parametric policy representation and an optimization algorithm to tune

the policy parameters. Few policy parameterizations discussed in the literature are suitable for the considered framework of MDPs with finite control input spaces. For example, the Gibbs policy [23] associates to every state a Boltzmann distribution over the finite set of actions, using the Q–function to express the relevance (energy) of each action. The Q-function is represented by a parametric function and the policy search process consists in tuning its parameters, the main advantage being that the optimization is carried out over a continuous parameterization. In [24], actions are associated to regions of the state space using Gaussian functions to perform such state aggregation. More specifically, the state space is covered by Gaussian functions, each labeled with a specific action, and the actual action associated to a specific state is given by the Gaussian with the highest value in that state. The policy is tuned by adapting the position and variance of all involved Gaussians. The interesting aspect of this approach is the ability to represent wide areas of the state space using concise information.

In this paper, we present a novel policy parametrization that combines the features of both the mentioned policy parameterizations. The policy is represented through particles positioned over the state space and labeled with different control actions. A set of particles defines a policy by partitioning the state space through the associated Voronoi diagram, so that the control action applied in a given state is the one associated to the closest particle. Particle positions and labels are tuned through the use of suitable parameterized Gaussian and Boltzmann functions, respectively.

Various optimization algorithms can be employed to tune the policy parameters. For example, [24] employs a gradient–free cross-entropy method [25] to tune the parameters of the Gaussian basis functions representing the policy. We focus here in particular on policy gradient methods [26], which perform a local search in the policy space guided by the gradient of the return obtained by simulating the MDP. These methods, as all gradient-based techniques, enjoy local convergence guarantees. Some particularly notable approaches belonging to this family are REINFORCE and G(PO)MDP [27]. However, they are known to suffer from slow convergence problems due to the high variance of the gradient estimates, caused by the repeated sampling from a probabilistic policy which leads to erratic rewards.

Policy Gradient with Parameter-based Exploration (PGPE) [28] is an alternative gradient-based strategy recently introduced in the literature, which overcomes this limitation by focusing on deterministic policies and introducing a probabilistic distribution over the policy parameters. The search in the policy space is then replaced by a search in the space of the parameters of such a distribution, denoted *hyperparameters*. This yields two main advantages over the previously mentioned gradient-based methods: i) the gradient estimates have a low variance [29] (leading to a faster convergence rate), and ii) the PGPE does not require the differentiability with respect to the policy parametrization, since the gradient is evaluated with respect to the hyperparameters.

In view of these features, the PGPE scheme has been here adopted to develop an optimization method for the proposed particle-based policy parametrization. More precisely, a probabilistic distribution of the policy parameters (particle positions and labels) is defined in terms of a multivariate Gaussian density function for the particle locations and a categorical distribution for the action labels. The mean vector and covariance matrix of the Gaussian density function, together with the parameters of the categorical distribution, constitute the hyperparameters to be tuned in the PGPE scheme.

However, as with the Gibbs and Gaussian basis function parameterizations, the optimization of a *fixed* size policy parametrization may yield unsatisfactory results. Indeed, choosing the appropriate number of particles in advance is a difficult task and a wrong choice may lead to poor performance (not enough particles) or very slow convergence rates (too many particles). Starting from this observation, an iterative process has been developed that constructs the policy by changing both its structure and parametrization according to its performance, as evaluated through system simulation. Particles may be added or removed at each iteration, thus refining the current best policy.

The following are the main contributions of this paper:

- Introduction of a particle–based representation of the control policy for MDPs with finite control space;
- Extension of the PGPE to this representation, which also includes a categorical distribution for the selection of the control action;
- Design of an iterative procedure for the structure selection of the particle-based policy parametrization.

A preliminary version of this work has been presented in [30], and is here extended and generalized, in particular by removing the limitation that the policy be characterized statically by a fixed number of particles with pre-assigned action.

The rest of the paper is organized as follows. Section II briefly recalls the concept of MDP and formulates the control policy optimization problem. It also describes the PGPE approach. The policy parametrization with particles is introduced in Section III. Section IV presents the iterative algorithm for particle selection. Its convergence properties and computational characteristics are discussed in Section V. Section VI illustrates the application of the presented method to two benchmark control problems, demonstrating the effectiveness and scalability properties of the iterative PGPE procedure. Finally, some conclusions are provided in Section VII.

II. Preliminaries

In this section we briefly describe the considered optimal control problem for discrete-time MDPs and provide an overview of the PGPE algorithm.

A. Optimal control of Markov Decision Processes

An MDP is a tuple $\langle S, U, f, r, \gamma, D \rangle$, where S is the state space; U is the control (or action) space; $f : S \times S \times U \to \mathbb{R}_+$ is the Markovian transition model with $f(s'|s, u)$ denoting the probability density function governing the one-step evolution from the current state s to the next state s' when the control action u is applied; $r : S \times U \times S \to \mathbb{R}$ is the reward

function, with $r(s,u,s')$ representing the one-step reward obtained when action u is applied in state s and the next state is s'; $\gamma \in [0,1)$ is a discount factor; and D is the distribution of the initial state.

A (stationary) stochastic policy is given by a probability density function $\pi(\cdot|s)$ over the action space U. If U is discrete, $\pi(u|s)$ denotes the probability of taking action u when the state of the MDP is s. When the policy is deterministic, with a slight abuse of notation, we use π to denote the map between states and actions, *i.e.*, $\pi : S \to U$.

We consider control problems where the future rewards are exponentially discounted with γ. The value of state s under policy π is expressed as the expected return when starting in s and following π thereafter[1]:

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{\substack{u(k)\sim\pi \\ s(k)\sim f}} \left[\sum_{k=0}^{T-1} \gamma^k r(s(k),u(k),s(k+1)) | s(0)=s \right],$$

where $\mathbb{E}$ denotes the expectation operator, and the notation $u(k) \sim \pi$ means that the random variable $u(k)$ is drawn from the density π at each step k (a similar interpretation holds for $s(k) \sim f$).

The time horizon length T can either be finite or infinite. Given the initial state distribution D, the policy performance can be evaluated through its expected discounted return:

$$J(\pi) = \mathop{\mathbb{E}}_{s(0)\sim D} \Big[V^{\pi}(s(0)) \Big].$$

Solving an MDP implies finding a policy π^* that maximizes the expected return: $\pi^* \in \arg\max_{\pi\in\Pi} J(\pi)$, where Π is the set of all stochastic policies.

B. Policy search: The PGPE algorithm

Let us consider a class of parameterized policies $\Pi_{\boldsymbol{\theta}} = \{\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \mathbb{R}^d\}$, where d is a positive integer. The problem of finding a policy $\pi_{\boldsymbol{\theta}} \in \Pi_{\boldsymbol{\theta}}$ that maximizes the expected discounted reward can then be addressed via policy gradient approaches, where the policy parameter vector $\boldsymbol{\theta}$ is updated following the direction of the gradient $\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}})$. We define a history to be a sequence $h = \{s_0, u_0, s_1, u_1, \ldots, s_T\}$ of states and actions along the time horizon $[0,T]$. Then, the expected return of a policy $\pi_{\boldsymbol{\theta}}$ can be written as an expectation over all the possible histories H:

$$J(\pi_{\boldsymbol{\theta}}) = \int_H p_{\boldsymbol{\theta}}(h) r(h) \mathrm{d}h, \tag{1}$$

where $r(h) = \sum_{k=0}^{T-1} \gamma^k r(s_k, u_k, s_{k+1})$ is the total cumulative discounted reward of a history h and

$$p_{\boldsymbol{\theta}}(h) = D(s_0) \prod_{k=0}^{T-1} f(s_{k+1}|s_k,u_k) \pi_{\boldsymbol{\theta}}(u_k|s_k)$$

is the probability density function of h when policy $\pi_{\boldsymbol{\theta}}$ is applied. This leads to the following expression for the gradient

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) &= \int_H p_{\boldsymbol{\theta}}(h) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(h) r(h) \mathrm{d}h \\ &= \int_H p_{\boldsymbol{\theta}}(h) \sum_{k=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(u_k|s_k) r(h) \mathrm{d}h, \end{aligned} \tag{2}$$

where the equality $\nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(h) = p_{\boldsymbol{\theta}}(h) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(h)$ has been exploited.

Since solving the integral (2) analytically is generally unfeasible, the gradient can only be estimated, *e.g.*, by means of Monte Carlo simulations as in the REINFORCE algorithm [31]:

$$\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{k=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(u_{k,i}|s_{k,i}) r(h_i),$$

where the histories h_i, $i = 1, \ldots, N$, are independently extracted from H according to $p_{\boldsymbol{\theta}}(\cdot)$. Note that, in the case when the time horizon is infinite, a finite length history is used to approximate the expected return (1) in the first place. This entails that some error is introduced, which, however, can be set arbitrarily low by suitably choosing the history length.

Local convergence properties have been established for REINFORCE [31], but a significant problem with this class of policy gradient algorithms is their slow convergence rate. This is caused by the high variance of the Monte Carlo estimate of the gradient [27], which derives from simulating stochastic policies (the control action is sampled at each step). In addition, the REINFORCE method requires the differentiability of the policy with respect to the parameters $\boldsymbol{\theta}$, which makes the policy design trickier when the control space is finite.

A method that has proved to be successful in mitigating the variance problem and that does not require policy differentiability is the PGPE algorithm [28]. Instead of optimizing over stochastic policies, the PGPE may consider only deterministic ones and moves the stochasticity to a higher level, introducing a probability distribution over the set of the policy parameters $\boldsymbol{\theta}$. An extraction from such probability distribution yields a deterministic policy. The optimization problem is thus reformulated as that of tuning such distribution so as to increase the probability of drawing deterministic policies with higher expected returns. To this aim, distribution $p_{\boldsymbol{\rho}}(\boldsymbol{\theta})$ is parameterized through coefficients $\boldsymbol{\rho}$, denoted "hyperparameters". It is by tuning these hyperparameters that $p_{\boldsymbol{\rho}}(\boldsymbol{\theta})$ is refined. In the specific case of the PGPE, the updating of such hyperparameters follows a classical gradient ascent approach.

Note that restricting the search to deterministic policies is a sensible strategy. Indeed, deterministic policies are more convenient from a practical perspective, since they are easier to implement. Also, under appropriate measurability assumptions, there exists at least one deterministic policy among the optimal ones, [32].

The performance measure in the PGPE is the expected value of $J(\pi_{\boldsymbol{\theta}})$ with respect to the distribution $p_{\boldsymbol{\rho}}(\cdot)$:

$$V(\boldsymbol{\rho}) = \int_{\Theta} p_{\boldsymbol{\rho}}(\boldsymbol{\theta}) J(\pi_{\boldsymbol{\theta}}) \mathrm{d}\boldsymbol{\theta}, \tag{3}$$

[1]The time dependence of stochastic variables is indicated in brackets, *e.g.*, $s(k)$, whereas s_k denotes a possible extracted value of $s(k)$.

and the optimal value for the hyperparameters $\boldsymbol{\rho}$ is given by

$$\boldsymbol{\rho}^* \in \arg\max_{\boldsymbol{\rho}} V(\boldsymbol{\rho}). \tag{4}$$

Ideally, the probability distribution $p_{\boldsymbol{\rho}^*}(\cdot)$ should be a probability mass function attributing probability one to the choice of an optimal deterministic policy $\pi_{\boldsymbol{\theta}^*}$.
The $\boldsymbol{\rho}$ parameters are updated along the gradient ascent direction:

$$\boldsymbol{\rho}(n+1) = \boldsymbol{\rho}(n) + \beta \nabla_{\boldsymbol{\rho}} V(\boldsymbol{\rho}(n)),$$

where

$$\begin{aligned}\nabla_{\boldsymbol{\rho}} V(\boldsymbol{\rho}) &= \int_\Theta p_{\boldsymbol{\rho}}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\rho}} \log p_{\boldsymbol{\rho}}(\boldsymbol{\theta}) J(\pi_{\boldsymbol{\theta}}) \mathrm{d}\boldsymbol{\theta} \\ &= \int_\Theta p_{\boldsymbol{\rho}}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\rho}} \log p_{\boldsymbol{\rho}}(\boldsymbol{\theta}) \int_H p_{\boldsymbol{\theta}}(h) r(h) \mathrm{d}h \mathrm{d}\boldsymbol{\theta}.\end{aligned}$$

A sampling method can be exploited also in this case to estimate the gradient, thus leading to:

$$\nabla_{\boldsymbol{\rho}} V(\boldsymbol{\rho}) \approx \frac{1}{N} \sum_{i=1}^{N} r(h_i) \nabla_{\boldsymbol{\rho}} \log p_{\boldsymbol{\rho}}(\boldsymbol{\theta}_i), \tag{5}$$

where each pair $(\boldsymbol{\theta}_i, h_i)$, $i = 1, \ldots, N$, is extracted independently. The components of each pair are generated according to the following mechanism: parameter $\boldsymbol{\theta}_i$ is drawn first from $p_{\boldsymbol{\rho}}(\cdot)$ and then history h_i is drawn from the conditional distribution $p_{\boldsymbol{\theta}_i}(\cdot)$.

In [29] it is shown that the gradient estimate in PGPE has a lower variance than in REINFORCE since one extracts multiple deterministic policies and for each extracted policy the actions along the history are determined only by the states that are visited and are not extracted at random. The variance in the gradient estimate can be further reduced by subtracting a baseline $b_{\boldsymbol{\rho}}$:

$$\nabla_{\boldsymbol{\rho}} V(\boldsymbol{\rho}) \approx \frac{1}{N} \sum_{i=1}^{N} (r(h_i) - b_{\boldsymbol{\rho}}) \nabla_{\boldsymbol{\rho}} \log p_{\boldsymbol{\rho}}(\boldsymbol{\theta}_i).$$

The optimal baseline has been derived in [29] by minimazing the variance of the above equation with respect to $b_{\boldsymbol{\rho}}$, leading to the following formula:

$$b^*_{\boldsymbol{\rho}} = \frac{\mathbb{E}_{\boldsymbol{\theta},h}\left[r(h) \left\| \nabla_{\boldsymbol{\rho}} \log p_{\boldsymbol{\rho}}(\boldsymbol{\theta}) \right\|^2 \right]}{\mathbb{E}_{\boldsymbol{\theta}}\left[\left\| \nabla_{\boldsymbol{\rho}} \log p_{\boldsymbol{\rho}}(\boldsymbol{\theta}) \right\|^2 \right]}.$$

III. A NOVEL PARTICLE-BASED PGPE PARAMETERIZATION

In this section, we present a new parameterization of a deterministic policy $\pi_{\boldsymbol{\theta}} : S \to U$ with a continuous state space S and a finite control space U, and we incorporate the parameterization into the PGPE framework.

In the literature, the PGPE method has only been applied in combination with policy parameterizations that are linear in the parameters and the adopted probability distribution for the policy parameters $\boldsymbol{\theta}$ is a multivariate Gaussian. Linear regression-type parameterizations are common in the function approximation literature, where they are employed to approximate a function defined over a continuous space, using various families of basis functions. In the policy search context, they can be used to approximate a policy, and they have been shown to be effective in high dimensional continuous problems. However, they are not directly applicable in the case of a finite discrete control space, where the policy turns out to be discontinuous. Inspired by clustering methods, we next introduce a parameterization of a policy that is well suited for the case when the control space is discrete.

A. Policy parameterization via particles

We use *particles* labeled with actions to identify the regions of the state space S that are mapped to different control actions. More precisely, a particle is a point in the state space S with a label in U, and the policy deterministically associates to $s \in S$ the action defined by the label of the particle that is closest to s. Provided a sufficient number of particles is used, one can in principle reproduce the map associated with the optimal policy with arbitrary accuracy.

A deterministic policy $\pi_{\boldsymbol{\theta}}$ is then represented as a set of p labelled particles, collectively described by the parameter vector:

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}^{(1)} \\ \vdots \\ \boldsymbol{\theta}^{(p)} \end{bmatrix},$$

where each particle $\boldsymbol{\theta}^{(i)} = \left[{s^{(i)}}^{\mathrm{T}}, {u^{(i)}}^{\mathrm{T}} \right]^{\mathrm{T}} \in S \times U$ is characterized by a "position" $s^{(i)}$ in the state space and a "label" $u^{(i)}$, that identifies the corresponding action in the control space. Given $s \in S$, the action u associated to s by policy $\pi_{\boldsymbol{\theta}}$ is obtained via the k-NN (k-nearest neighbors) algorithm, with $k = 1$:

$$\pi_{\boldsymbol{\theta}}(s) = u^{(i)}, \quad \text{with } i = \arg\min_j \Delta(s, s^{(j)}),$$

where $\Delta(s, s^{(i)})$ is the Euclidean distance between s and the position $s^{(i)}$ of the i-th particle. As a result, the state space is partitioned into polyhedral sets defining the Voronoi diagram generated by the particle locations, each set defining a region in the state space associated to a specific particle. The idea is to identify the best action and position for each particle in such a way that states in its neighborhood (as induced by the Euclidean norm distance) share the same optimal action.

An example of a Voronoi diagram is given in Figure 1, for a 2-dimensional state space S. Figure 1 plots also the corresponding Delaunay diagram, which is the dual graph of the Voronoi diagram [33].

B. Distribution of the policy parameters

According to the PGPE scheme, the policy parameters are drawn from some probability distribution

$$\boldsymbol{\theta} \sim p_{\boldsymbol{\rho}}(\cdot).$$

The p particles $\boldsymbol{\theta}^{(i)}$, $i = 1, \ldots, p$, are assumed to be independent, and each one has its own distribution over $S \times U$.

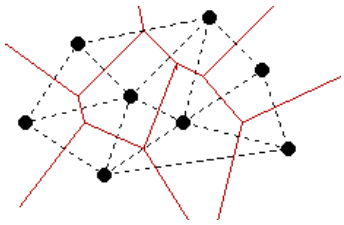


Fig. 1. Pictorial representation of the Voronoi (solid lines) and Delaunay (dashed lines) diagrams associated with a set of particles.

Given a particle $\boldsymbol{\theta}^{(i)} = \left[s^{(i)\mathrm{T}}, u^{(i)\mathrm{T}}\right]^{\mathrm{T}}$, we assume that its position and label are independent, in order to simplify the policy representation. Position $s^{(i)}$ has a Gaussian distribution $s^{(i)} \sim \mathcal{N}(\cdot; \mu^{(i)}, \Sigma^{(i)})$, and label $u^{(i)}$ has a Boltzmann distribution $u^{(i)} \sim \mathcal{B}(\cdot; \alpha^{(i)})$. More specifically, the probability of selecting action u_l in the control space U of cardinality $|U| = m$ is given by

$$\mathcal{B}\left(u_l; \alpha^{(i)}\right) = \frac{e^{\alpha_l^{(i)}}}{\sum_{j=1}^{m} e^{\alpha_j^{(i)}}}.$$

The hyperparameters $\boldsymbol{\rho}$ are then defined as follows:

$$\boldsymbol{\rho} = \begin{bmatrix} \boldsymbol{\rho}^{(1)} \\ \vdots \\ \boldsymbol{\rho}^{(p)} \end{bmatrix}, \text{ where } \boldsymbol{\rho}^{(i)} = \begin{bmatrix} \mu^{(i)} \\ \mathrm{vect}(\Sigma^{(i)}) \\ \alpha^{(i)} \end{bmatrix}.$$

A graphical representation of the particles distribution $p_{\boldsymbol{\rho}}(\cdot)$ is given in Figure 2, together with a policy with parameters extracted from $p_{\boldsymbol{\rho}}(\cdot)$.

In order to reduce the number of parameters, we discard the cross-correlation terms in the covariance matrix, so that $\Sigma^{(i)}$ is diagonal, with the elements on the diagonal parameterized by the logistic function

$$\Sigma_{jj}^{(i)} = \frac{\tau}{1 + e^{-\sigma_j^{(i)}}}$$

so as to prevent the variance from becoming negative [34]. Coefficient τ is a design parameter that determines the asymptotic (maximum) value of the variance. The total number of hyperparameters is then $n_{\boldsymbol{\rho}} = (2n + m) \cdot p$, where n is the dimension of the state space S, m is the number of actions in U, and p is the number of particles.

Each hyperparameter vector $\boldsymbol{\rho}$ defines a probability density function over the set of deterministic policies $\Pi_{\boldsymbol{\theta}}$ parameterized via particles. In the following, we will use the concept of *representative* policy $\bar{\pi}_{\boldsymbol{\rho}}$ associated to hyperparameters $\boldsymbol{\rho}$ to indicate the deterministic policy given by the particles

$$\boldsymbol{\theta}^{(i)} = \begin{bmatrix} \mu^{(i)} \\ \arg\max_{u \in U} \mathcal{B}(u; \alpha^{(i)}) \end{bmatrix}, \; i = 1, \ldots, p.$$

C. Gradient estimation

The partial derivatives of $\log p_{\boldsymbol{\rho}}(\boldsymbol{\theta}) = \sum_{j=1}^{p} \log p_{\boldsymbol{\rho}}(\boldsymbol{\theta}^{(j)})$ with respect to $\mu^{(i)}$, $\Sigma^{(i)}$ and $\alpha^{(i)}$, required to compute the gradient $\nabla_{\boldsymbol{\rho}} V(\boldsymbol{\rho})$ (see expression (5)), are given by:

$$\begin{aligned}
\nabla_{\mu^{(i)}} \log p_{\boldsymbol{\rho}}(\boldsymbol{\theta}^{(j)}) =& \delta_{ij} (\Sigma^{(j)})^{-1} (s^{(j)} - \mu^{(j)}) \\
\nabla_{\sigma_l^{(i)}} \log p_{\boldsymbol{\rho}}(\boldsymbol{\theta}^{(j)}) =& \delta_{ij} \frac{e^{-\sigma_l^{(i)}}}{2\left(1 + e^{-\sigma_l^{(i)}}\right)} \\
& \times \left[(s_l^{(j)} - \mu_l^{(j)})^2 (\Sigma_{ll}^{(j)})^{-1} - 1 \right] \\
\nabla_{\alpha_l^{(i)}} \log p_{\boldsymbol{\rho}}(\boldsymbol{\theta}^{(j)}) =& \tau \delta_{ij} \left(\delta_{u^{(j)} u_l} - \mathcal{B}(u_l; \alpha^{(j)}) \right),
\end{aligned}$$

where $\delta_{ij} = 1$ when $i = j$, and 0 otherwise.

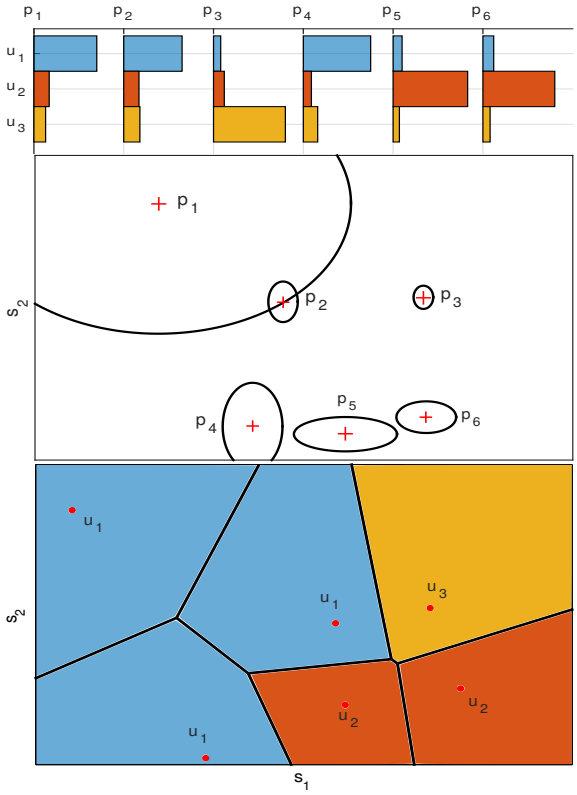


Fig. 2. Illustrative picture of a policy parameter distribution and of an extracted policy in a two dimensional state space ($S \subseteq \mathbb{R}^2$) with 3 actions ($|U| = 3$): Boltzmann (top) and Gaussian (middle) distributions of 6 particles (the bars represent the action probabilities, the plus symbols and the ellipses represent the mean values and the standard deviation isolevel curves, respectively), and Voronoi diagram (bottom) of a deterministic policy extracted from the distributions of the particles.

IV. Iterative PGPE algorithm

In this section, we present an iterative algorithm that refines the policy obtained through the proposed particle-based PGPE approach. In the particle-based parameterization of a policy, a particle is used to identify a connected region of the space that is labeled with an action. In principle, by suitably deciding the number of particles, and their locations and labels, it would be possible to define a particle-based approximation of an optimal deterministic policy up to some desired accuracy. The definition of the particle set—and, in particular, its cardinality—is a critical issue.

The idea proposed here is to set-up an iterative procedure that adds and removes particles until a proper policy parameterization is found. One starts by applying the PGPE approach to some initial particle set. The resulting hyperparameters are referred to as "optimal" hyperparameters and the performance of the representative policy associated with them as "best" performance. At each iteration, one new particle is added with a certain initialization of the related hyperparameters. Then, all hyperparameters are optimized through the PGPE approach. After convergence, redundant particles are removed. The performance of the representative control policy associated to the resulting hyperparameters is evaluated. If such a performance is better than the current best, then it takes its place and the corresponding hyperparameters are marked as optimal. Otherwise, the best performance is not updated and a new iteration starts. The algorithm ends when a maximum number of iterations is reached. The pseudo-code of the iterative procedure iPGPE is reported in Algorithm 1.

Algorithm 1 iPGPE algorithm

Input: $\boldsymbol{\rho}(0)$

$\boldsymbol{\rho}^*(0) \leftarrow \text{PGPE}(\boldsymbol{\rho}(0))$

$[\ \boldsymbol{\rho}^*(0), \bar{\pi}_{\boldsymbol{\rho}^*(0)}\ ] \leftarrow \text{FIX}(\boldsymbol{\rho}^*(0))$

$\pi^\star \leftarrow \bar{\pi}_{\boldsymbol{\rho}^*(0)}$

$\boldsymbol{\rho}^\star \leftarrow \boldsymbol{\rho}^*(0)$

for $k = 1$ to K **do**

  $\boldsymbol{\rho}(k) \leftarrow \boldsymbol{\rho}^\star$

  $\Sigma^{(i)}(k) \leftarrow c\Sigma^{(i)}(k), i = 1, \ldots, p$

  $\boldsymbol{\rho}(k) \leftarrow \begin{bmatrix} \boldsymbol{\rho}(k) \\ \boldsymbol{\rho}^+ \end{bmatrix}$ where $\boldsymbol{\rho}^+ = \begin{bmatrix} \mu^+ \\ \text{vect}(\Sigma^+) \\ \alpha^+ \end{bmatrix}$ with

    $\mu^+ \leftarrow \text{DELAUNAY}(\mu^{(1)}(k), \ldots, \mu^{(p)}(k))$

    $\Sigma^+ \leftarrow \Sigma^{(1)}(0)$

    $\alpha^+ \leftarrow \mathbf{0}$

  $\boldsymbol{\rho}^*(k) \leftarrow \text{PGPE}(\boldsymbol{\rho}(k))$

  $[\boldsymbol{\rho}^*(k), \bar{\pi}_{\boldsymbol{\rho}^*(k)}] \leftarrow \text{FIX}(\boldsymbol{\rho}^*(k))$

  if $J(\bar{\pi}_{\boldsymbol{\rho}^*(k)}) > J(\bar{\pi}_{\boldsymbol{\rho}^\star})$ **then**

    $[\boldsymbol{\rho}^\star, \pi^\star] \leftarrow [\boldsymbol{\rho}^*(k), \bar{\pi}_{\boldsymbol{\rho}^*(k)}]$

  end if

end for

Output: Optimal deterministic policy $\pi^\star$

The algorithm starts by taking as input an initial set $\boldsymbol{\rho}(0)$ of hyperparameters. The PGPE procedure is run to optimize the hyperparameters $\boldsymbol{\rho}(0)$ with respect to the performance index $V(\boldsymbol{\rho})$ defined in (3). Redundant particles are removed from the resulting hyperparameters $\boldsymbol{\rho}^*(0)$ by means of the procedure FIX. This procedure consists of the following steps: *(i)* it builds the representative policy $\bar{\pi}_{\boldsymbol{\rho}^*(0)}$ associated to the optimal hyperparameters $\boldsymbol{\rho}^*(0)$ and *(ii)* it detects the particles that are redundant and can be removed without modifying the representative policy performance.

A particle is redundant if it does not contribute to the definition of the state space partition in regions associated with different actions, since it is surrounded by particles with the same label. This condition is purely geometric and can be easily verified by building the Voronoi diagram. A particle may be redundant also because it belongs to a region of the state space that is not visited when the representative policy is applied.

At the k-th iteration the best parameterization $\boldsymbol{\rho}^\star$ obtained so far (namely, the one associated to the representative policy $\pi^\star$ with the highest expected return J) is chosen as the current hyperparameter vector $\boldsymbol{\rho}(k)$. The variance of the corresponding particles is increased by a factor $c > 1$ to encourage the exploration of the policy space.

Then, a new particle is added to the current parameterization. The hyperparameters $\boldsymbol{\rho}^+$ of the new particle are set as follows: (i) the position μ^+ is set equal to one of the intersection points between the Voronoi diagram and the Delaunay diagram of the current representative policy, selected at random; (ii) the variance Σ^+ of the new particle is set equal to the variance of the first particle in $\boldsymbol{\rho}(0)$; (iii) the weights of the Boltzmann distribution α^+ are set to 0 so as to obtain a uniform probability over the labels. The rationale is that the new particle should serve the purpose of refining the policy map, and hence it is placed on a facet of a polytope in the Voronoi diagram defining the current policy map. The intersection of such facet with the Delaunay diagram provides a convenient candidate position for the new particle, in that it is the nearest point to the particles determining the considered facet, and hence most likely to disrupt significantly the previous solution. Such set of points is identified by the intersections between the Voronoi diagram and the Delaunay diagram. The PGPE and FIX procedures are then executed starting from the new augmented hyperparameter vector. If the representative policy associated to the resulting hyperparameters $\boldsymbol{\rho}^*(k)$ improves over the current best, it is stored in its stead.

The iterative nature of the algorithm carries two favorable properties. The process of insertion and removal of particles helps to alleviate the sensitivity of the algorithm with respect to the hyperparameters initialization and to reduce the risk of being stuck in local optima (which is a typical drawback of gradient–based optimization algorithms). Indeed, the structural modification of the policy parameterization can be reinterpreted as a partial reset of the current solution, that causes the optimization to restart from a different point in the hyperparameters space.

V. Algorithm analysis

A. Convergence properties

The PGPE algorithm belongs to the class of policy gradient algorithms to which the following properties and considerations apply. Specifically, convergence is obtained if the estimated gradients are unbiased and the learning rates β_k satisfy the conditions [5], [27]:

$$\sum_{k=0}^{\infty} \beta_k = \infty, \qquad \sum_{k=0}^{\infty} \beta_k^2 < \infty \tag{6}$$

Although in the general case policy gradient algorithms are guaranteed to converge only to locally optimal policies, in [35] the author shows that policy gradient approaches converge to a globally optimal policy under the same assumptions required by Q-based algorithms—*e.g.*, SARSA(λ) [5]—that are commonly used to solve RL problems. Such assumptions

require that the state and action spaces are countable, while for continuous-domain problems only convergence to a local optimum can be guaranteed. Nonetheless, as suggested in [35], local optima can be avoided by increasing exploration and the representational power of the policy parametrization, which is actually what is pursued by the iterative insertion of new particles in the proposed iPGPE approach.

More in detail, the inner loop of the iPGPE algorithm (step 9 in Alg. 1) amounts to the execution of the PGPE with a fixed structure policy parameterization, which is guaranteed to converge to a local optimum. The outer loop proposes a structure modification of the policy which is accepted if the local optimum it leads to improves over the current best (notice that the two local optima could be defined in different hyperparameter spaces). As such, the iPGPE is also endowed with local convergence properties. An example discussed in Section VI emphasizes the performance improvements achievable by means of the structural modification of the policy parameterization carried out in the outer loop of the iPGPE.

B. Computational complexity

We next analyze the worst-case computational complexity of the iPGPE, providing an asymptotic upper bound expressed in the so called $\mathcal{O}$–notation.

For this purpose, it is necessary to first provide a bound for the plain PGPE algorithm, which is iteratively executed in the inner loop of the iPGPE (step 9 in Alg. 1). The structure of the PGPE algorithm comprises two nested loops, one accounting for the gradient ascent iterations, limited to some M value, and the other for the gradient estimation, which is repeated exactly N times, N being the number of pairs $(\boldsymbol{\theta}_i, h_i)$ of histories and parameters employed. In the gradient estimation loop, two main tasks are performed. The computation of the partial derivatives of $\log p_{\boldsymbol{\rho}}(\boldsymbol{\theta})$ with respect to the hyperparameters has a complexity of order $\mathcal{O}(n_{\boldsymbol{\rho}})$ for the assumed setting (see Section III.C), where $n_{\boldsymbol{\rho}}$ is the number of hyperparameters. The other task, namely the Monte Carlo simulation of the system is of order $\mathcal{O}(T)$, where T is the time horizon length. Each step of the system simulation involves the selection of the control action using the particle policy, which amounts to $\mathcal{O}(np)$ operations, where n is the size of the state space and p is the number of particles, since it requires the calculation of the distance between the current state and all the particles and the selection of the nearest one. In summary, the complexity of the PGPE is of order $\mathcal{O}(MN(Tnp + n_{\boldsymbol{\rho}}))$.

The iPGPE consists of an iterative procedure that adds and removes particles until a proper policy parametrization is found. A PGPE iteration is launched after each structural modification of the policy. The computational bottleneck is represented by the Delaunay triangulation process, for which a worst-case exponential bound $\mathcal{O}(p^{\lceil n/2 \rceil})$ is given in [36]. However, many efficient algorithms are available for this operation (see, *e.g.*, [37]). Notice also that much less costly heuristics can be employed to pick the positions of new particles compared to the Delaunay triangulation, in order to mitigate this source of computational complexity. Finally, since the maximum number of the outer iterations is bounded by K, the overall computational complexity of the iPGPE is of order $\mathcal{O}(KMN(Tnp + n_{\boldsymbol{\rho}}) + Kp^{\lceil n/2 \rceil})$.

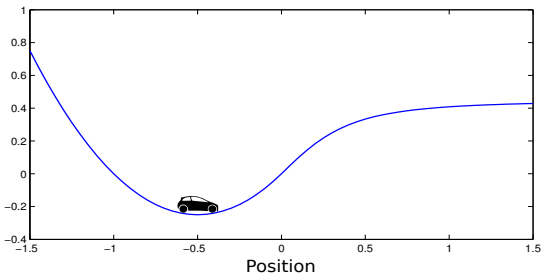


Fig. 3. The "Car on the Hill" picture.

VI. Simulation examples

Two numerical examples are presented in this section. The first example is the classical "Car on the Hill" problem described in [9]. The second example addresses a multi-room heating control problem and is inspired by the benchmark described in [38], and addressed also in [39], [1], [40].

A. Car on the Hill example

An under-powered car is traveling in a valley between two hills (see Figure 3), and the objective is to bring the car to the top of the rightmost hill in minimum time, while preventing the position and the velocity of the car to exit some prescribed set. The control policy decides whether to drive the car to the left, to the right or not to use the engine at all. Since gravity is stronger than the car engine, even at full thrust the car cannot accelerate up the steep slope and the only way to solve the problem is to drive up the opposite hill before reaching the goal to the rightmost hill. The model of the car [9] is described by the following dynamics:

$$\begin{cases} \dot{x} &= v \\ \dot{v} &= \dfrac{u}{m(1+H'(x)^2)} - \dfrac{gH'(x)}{1+H'(x)^2} - \dfrac{v^2 H'(x) H''(x)}{1+H'(x)^2} \end{cases} \tag{7}$$

where $(x, v) \in \mathbb{R}^2$ is the continuous state component of the system, x and $v = \dot{x}$ being respectively the position and the velocity of the car, and $u \in U = \{-4, 0, 4\}$ is the control action representing the horizontal force applied to the car. A binary discrete state q is also present to distinguish the normal mode ($q = 1$), from the terminal conditions where either the car has exited some prescribed set $\{(x, v) : x \geq -2, |v| \leq 4\}$, or it has reached the goal, *i.e.* the target set $A = \{(x, v) : x > 1, |v| \leq 4\}$ ($q = 0$). When one of the two described terminal conditions is reached, q is switched to 0, and the car motion is stopped. The full hybrid state vector is then given by $s = (q, x, v)$.

Function $H(x)$ describes the slope of the hill and is defined by:

$$H(x) = \begin{cases} x^2 + x & \text{if} \quad x < 0 \\ x/\sqrt{1+5x^2} & \text{if} \quad x \geq 0 \end{cases}. \tag{8}$$

Parameters $m = 1$ and $g = 9.81$ are the mass of the car and the gravitational acceleration, respectively.

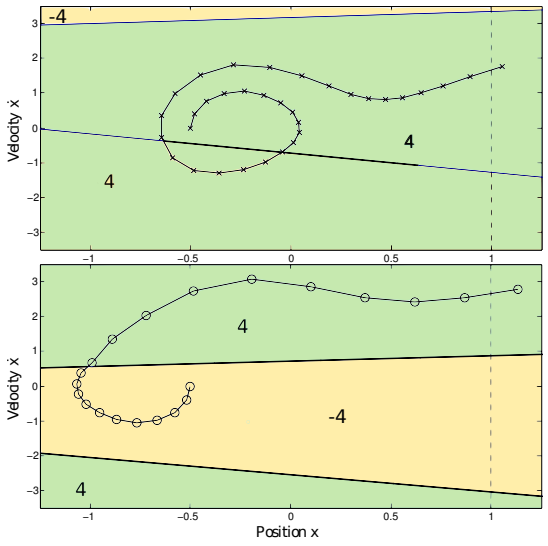


Fig. 4. Car on the Hill problem – Continuous state evolution starting from $[-0.5, 0]$ in the discrete state $q = 1$ obtained with policy $\bar{\pi}_{\boldsymbol{\rho}^*(1)}$ (top) and $\bar{\pi}_{\boldsymbol{\rho}^*(4)}$ (bottom).

System (7) is discretized with a sampling time $\Delta t = 0.1$ and assuming a constant input over each sample time interval. The reward function $r : S \times U \times S \to \{-1, 0, 1\}$ is defined by the following expression:

$$r(s, u, s') = \begin{cases} -1 & \text{if } x' < -2 \text{ or } |v'| > 4 \text{ and } q = 1 \\ 1 & \text{if } x' > 1 \text{ and } |v'| \leq 4 \text{ and } q = 1 \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

where $s = (q, x, v)$ and $s' = (q', x', v')$ are the current and next states, respectively.

A close analysis of the iPGPE algorithm evolution on the Car on Hill problem shows its ability to tune the policy parameters and the number of particles, in order to improve the performance. The initial number of particles is set to 3, one for each possible control action. The particle positions are initialized randomly, with unitary initial variance and $\tau = 6$. The initial car state is $[1, -0.5, 0]$ (the car is initially in the valley). The gradient is evaluated based on 1000 extractions of deterministic policies.

Figure 4 (top) graphically displays the policy $\bar{\pi}_{\boldsymbol{\rho}^*(1)}$ obtained by the algorithm after the 1^{st} iteration. Different colors identify regions of the state space associated to different control actions, according to the Voronoi partition of the representative policy. Although the goal is reached, the policy is not optimal since the car traverses the valley twice, using more time than necessary to reach the goal.

Thanks to the insertion of an additional particle at the 3^{rd} iteration, the iPGPE algorithm is able to find a more rewarding policy, that initially moves the car away from the target position to gain sufficient momentum to reach the top of the hill, as shown in Figure 5. The optimal history obtained at the last iteration of the iPGPE is shown in Figure 4 (bottom).

The process of insertion and removal of particles during the mentioned iterations is summarized in Figure 6. The reader may notice that the particle added at the 2^{nd} iteration did not improve the policy, leading to the same performance of $\bar{\pi}_{\boldsymbol{\rho}^*(1)}$. As a consequence, the iPGPE recovers the hyperparameters $\boldsymbol{\rho}^*(1)$ of the 1^{st} iteration, looking for another location for the insertion of a new particle. At the 3^{rd} iteration the algorithm found a suitable position for the new particle that improves the policy. A close examination of Figures 6.b-c reveals the removal of particles $\boldsymbol{\theta}^{(1)}$ and $\boldsymbol{\theta}^{(2)}$, which turn out to be redundant at the 2^{nd} and 3^{rd} iterations, respectively.

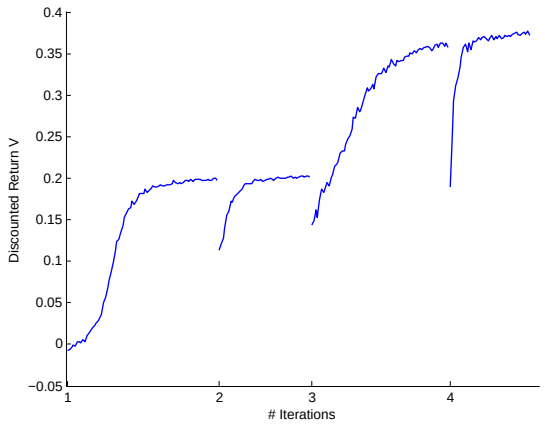
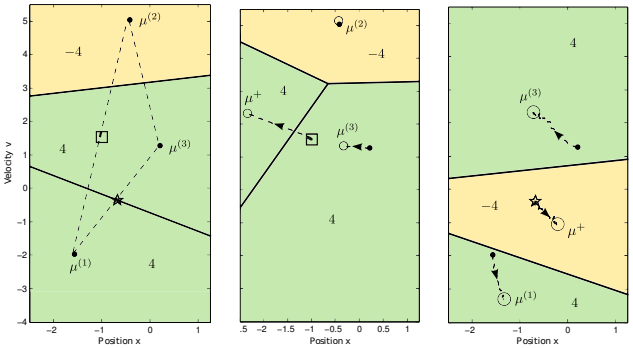


Fig. 5. Car on the Hill problem – Evolution of the performance index $V(\boldsymbol{\rho})$ as a function of the iPGPE iterations.



(a) (b) (c)

Fig. 6. Car on the Hill problem – Insertion and removal of particles: a) representative policy $\bar{\pi}_{\boldsymbol{\rho}^*(1)}$ (the square and the star symbols are the locations where new particles will be added at the 2^{nd} and 3^{rd} iterations), b) representative policy $\bar{\pi}_{\boldsymbol{\rho}^*(2)}$, and c) representative policy $\bar{\pi}_{\boldsymbol{\rho}^*(3)}$.

It is interesting to underline that the problem is characterized by the presence of a useless action ($u = 0$). As shown in Figure 4, the algorithm is able to discard this action from the very start. Furthermore, the car cannot simply drive up the steep slope ($u = 4$), but it must use the features of the landscape to build momentum and eventually escape the valley.

In this example, multiple iterations were necessary for refining the control policy and achieving better results. The process of insertion and removal of particles allowed to escape from local optima and to recover from unfavorable initial conditions of the hyperparameters. More precisely, the iPGPE moves from one local maximum to an improved one, as clearly shown in Figure 5.

For comparison purposes and to assess the quality of the solution found through the iPGPE, the same problem

has been addressed with an alternative state-of-the-art algorithm, namely the Relative Entropy Policy Search (REPS) algorithm [17]. This comparison is all the more interesting considering that a completely different policy representation is adopted, *i.e.* a Gibbs policy with the Q-function approximated using a linear combination of radial basis functions (RBF). In the example, 16 RBFs uniformly distributed in the continuous state space $[-2, 1] \times [-4, 4]$ have been used.

Figure 7 shows the trajectory (originating from the initial state $[1, -0.5, 0]$) obtained by the optimal policy learned with the REPS. Comparing Figures 7 and 4, it appears that both methods converge to the same trajectory, which confirms the capability of the iPGPE scheme to approach optimality. On the other hand the calculated state-action maps appear to be extremely different, which requires some further discussion. Indeed, the task addressed in this particular benchmark problem is not that of finding the full control policy, but exclusively the optimal trajectory from the initial point. This greatly constrains the state exploration process, which is important in that the resulting policies are optimal only with respect to the partial exploration of the space they resulted from. The two policies found with the iPGPE and the REPS, though extremely different, are in fact equivalent in providing the optimal trajectory from the initial point.

B. Multi-room heating problem

The temperature of n rooms is controlled by means of n heaters, one per room. The objective is to design a control policy that keeps the temperature of each room in a desired range activating as few heaters as possible at the same time. The control policy defines which rooms should be heated, depending on the temperature values in all rooms. A nice feature of this benchmark is that it is suitable for testing the scalability of the proposed algorithm, since the problem dimensionality can be easily increased by adding more rooms.

The described system is stochastic and hybrid, the state being defined as $s = (q, x) \in S$, where the discrete state component q identifies the actual combination of rooms being heated, while the continuous state $x = (x_1, \ldots, x_n) \in X = \mathbb{R}^n$ represents the (average) temperature in each room. Accordingly, the discrete state space $\mathcal{Q}$ is defined as the power set of $\{1, \ldots, n\}$ and the control space is given by $U = \mathcal{Q}$. Action $u \in U$ corresponds to the command of heating a certain set of rooms, while $q \in \mathcal{Q}$ indicates the set of rooms that are actually heated.

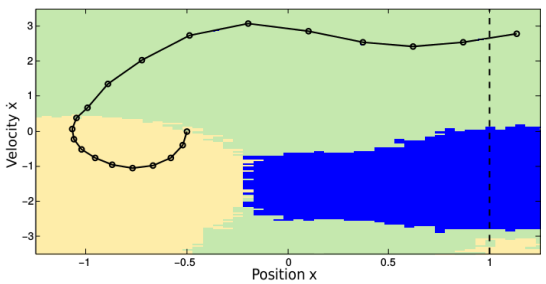


Fig. 7. Optimal Gibbs policy calculated with the REPS algorithm.

The average temperature in room i is governed by the following stochastic difference equation, obtained by Euler discretization of the corresponding continuous time dynamics with constant time step Δt:

$$x_i(k+1) = x_i(k) + [b_i(x_a - x_i(k)) + c_i h_i(k) + \sum_{j=1,\ldots,n; j\neq i} a_{ij}(x_j(k) - x_i(k))]\Delta t + n_i(k),\ i = 1, \ldots, n, \tag{10}$$

where $x_i(k)$ is the average temperature in room i at time k, x_a is the ambient temperature (assumed constant), and $h_i(k)$ is a boolean function equal to 1 when room i is heated, and 0 otherwise. Parameter a_{ij} is the heat exchange coefficient between room i and room j, b_i represents the heat loss rate of room i to the ambient, and c_i is the heat rate supplied by the heater in room i, all these coefficients being normalized with respect to the average thermal capacity of room i. Finally, the disturbance $n_i(k)$ affecting the temperature of room i is assumed to be a sequence of i.i.d. Gaussian random variables with zero mean and variance $\nu^2 \Delta t$, independent of $n_j(k)$, $j \neq i$.

The heaters are controlled by a thermostat that is subject to delay and switching failures. This is modeled through a discrete transition probability function that governs the mode transitions:

$$f_q(q'|q, u) = \begin{cases} 1, & u = q = q' \\ 1 - \alpha, & u \neq q, q' = q \\ \alpha, & u = q', q' \neq q \end{cases}, \tag{11}$$

where q is the current discrete state, q' is the next one, and $\alpha \in [0, 1]$ is the one-step delay/failure probability.

The desired operating region is given by

$$A = [x_l, x_u]^n,$$

where x_l and x_u specify the lower and upper bounds for the temperature in each room.

The control design problem can be formulated as in Section II, by defining the reward function $r : S \times U \times S \to \mathbb{R}$ as:

$$r(s, u, s') = \frac{1}{n}\Big(g(x') - g(x) - \beta\ (|q'| - 1)\Big)$$

where $s = (q, x)$, $s' = (q', x')$, $g(x) = \sum_{i=1}^{n} \mathbf{1}_{[x_l, x_u]}(x_i)$ is the number of rooms whose temperature is within the desired range $[x_l, x_u]$, $|q'|$ is the cardinality of the set $q' \in \mathcal{Q}$, and β is a weight coefficient that penalizes the simultaneous heating of more than one room at a time.

Transitions leading the temperature of k rooms outside $[x_l, x_u]$ are penalized with $-k/n$, and transitions leading the temperature of k rooms back into $[x_l, x_u]$ are rewarded with $+k/n$. All other transitions do not provide neither a penalty nor a reward. The discount factor γ weighs short-term rewards more than long-term ones, favoring a rapid come back into A if some room has exited it.

The parameters are set as follows: $\Delta t = 1/30$, $\nu = 1$, $x_a = 6$, $\alpha = 0.8$, $b_i = 0.25$ and $c_i = 12$ for $i = 1 \ldots n$, $a_{ij} = a_{ji} = 0.33$, for $i = 1, \ldots, n-1$, $j = i + 1$. The "safe" temperature range is $A = [17.5, 22]^n$. The time horizon is $T = 100$, the weighting coefficient is $\beta = 1/100$ and the discount factor is

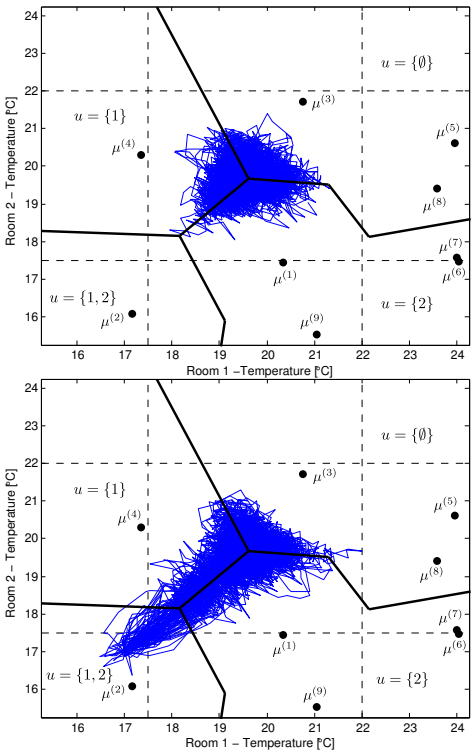


Fig. 8. Multi-room heating problem: 100 temperature trajectories (blue lines) obtained following the representative policy, starting from $x(0) = [19\ 19]^{\mathrm{T}}$ (top) and $x(0) = [17\ 17]^{\mathrm{T}}$ (bottom).

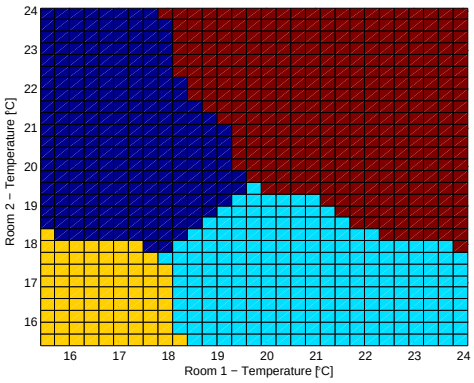


Fig. 9. Optimal policy for the ADP approach.

TABLE I
ADP APPROACH: AVERAGE OF $S(h)$ (IN PERCENTAGE) OVER 10000 HISTORIES.

# rooms	recovery problem	safety problem	general problem
1	95.38	99.97	96.53
2	93.78	99.94	94.98
3	93.00	99.90	93.67

set to $\gamma = 0.95$. The initial state distribution D is characterized by a uniform distribution over the set $[15.25, 24.25]^n$ for the room temperatures, while the discrete state is deterministically set to the condition where no room is heated.

The iPGPE algorithm has been applied to this problem setting the initial number of particles equal to $|U|$, and assigning each action to one of the particles. The positions of the particles are randomly initialized over the domain $[15.25, 24.25]^n$, with variances set to 6 and $\tau = 6$. The Monte Carlo estimates of the gradient are based on 1000 deterministic policies drawn from the distribution described by the current hyperparameters, and the set of initial states is composed by $100n$ states drawn from D.

Figure 8 refers to the 2-room case and shows 100 temperature trajectories obtained using the representative policy resulting from the application of the iPGPE, starting from $x(0) = [19\ 19]^{\mathrm{T}}$ and $x(0) = [17\ 17]^{\mathrm{T}}$.

In the sequel, the iPGPE algorithm is compared with the ADP approach based on state gridding proposed in [40]. The latter approach has been applied using 30 bins to discretize the temperature of each room, uniformly partitioning each dimension of the (continuous) state space. Though well known to scale exponentially with the state dimensions, the gridding scheme provides a reliable reference on the optimal performance. Indeed, the policy calculated in this way tends to the optimal one for decreasing sizes of the grid bins.

Figure 9 shows the optimal policy calculated using the mentioned state gridding ADP approach. Apparently, the policy calculated by the iPGPE is well in agreement with the ADP one, especially in the more densely explored areas, showing the capability of the algorithm to effectively approach optimality.

Tables I and II compare the two approaches in terms of the *safety index*:

$$S(h) = \frac{1}{T}\sum_{k=1}^{T} \mathbf{1}_A(x(k)),$$

where $\mathbf{1}_A(\cdot)$ denotes the indicator function ($\mathbf{1}_A(x) = 1$ if $x \in A$, and 0 otherwise). $S(h)$ represents the percentage of time steps for which the temperature of all rooms is within the safe set A. To emphasize the robustness and reliability of the iPGPE, 10 independent executions have been performed, starting from different initial hyperparameters, and a statistic of the performances of the resulting policies is provided in Table II. Both the ADP policy and the 10 iPGPE policies have been subjected to 10000 tests, starting from different initial states. The reported figures concern the average performance of each policy over all these validation histories h_i, $i = 1, \ldots, 10000$. The initial states have been uniformly extracted over different regions, depending on the specific problem addressed. More precisely, the initial states have been extracted over the set $[15, 17.5]^n$ (*recovery* problem), over A (*safety* problem), and over $[15.25, 24.25]^n$ (*general* problem). In the safety problem, the objective is to keep the temperature within A as long as possible, while in the recovery problem, the goal is to reach A as soon as possible starting from an initial condition outside A.

The ADP approach has been tested only up to $n = 3$ for computational reasons. To deal with larger problem instances the number of bins used for gridding has to be drastically reduced to avoid memory overflow, resulting in dramatic performance losses. Notice also that the ADP approach has

TABLE II
iPGPE ALGORITHM: STATISTICS OF THE AVERAGE $S(h)$ PERFORMANCE FOR 10 POLICIES OBTAINED STARTING FROM DIFFERENT INITIALIZATIONS (MEAN VALUE WITH 95% CONFIDENCE INTERVAL).

# rooms	recovery problem	safety problem	general problem
1	95.38 ± 0.01	99.96 ± 0.01	96.47 ± 0.01
2	94.15 ± 0.04	99.92 ± 0.05	94.84 ± 0.26
3	93.26 ± 0.44	99.85 ± 0.08	93.37 ± 0.54
4	91.12 ± 1.37	99.78 ± 0.04	92.47 ± 0.16
5	89.16 ± 1.31	99.65 ± 0.08	91.30 ± 0.23
10	79.95 ± 2.95	97.87 ± 0.96	82.90 ± 1.42

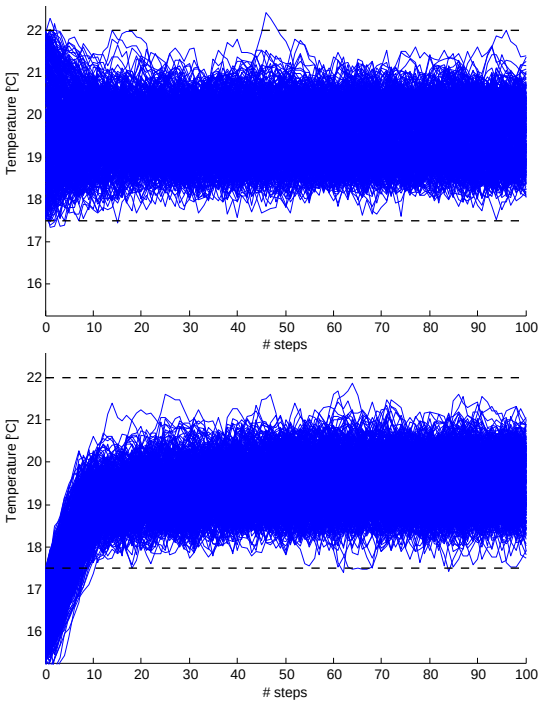


Fig. 10. Multi-room heating problem – Performance of the iPGPE algorithm in the safety (top) and recovery (bottom) problems for the 5-rooms case: 100 histories obtained following the policy $\pi^\star$ starting from 100 initial states drawn in A (top) or in $[15, 17.5]^n$ (bottom).

complexity $|G||U|(2 + |U|)$ per iteration, where $|G|$ is the number of bins in the gridded state space [8].

The iPGPE algorithm yields comparable results in the cases where both approaches are applicable, but scales much more favorably, performing reasonably well even with 10 rooms. Table II also reports the variability of the performance results with respect to different executions of the iPGPE. Apparently, all executions converge to policies that yield comparable performances.

Consider, *e.g.*, the 5-rooms scenario. The histories obtained with the representative iPGPE policy in the safety and recovery problems are shown in Figure 10 (the temperature histories of all rooms are plotted on a single axis for simplicity).

The optimal policy appears to be quite capable of driving and maintaining the temperature of all the rooms inside the safe set efficiently.

It is also worth noting how the weight coefficient β influences the control policy: even if in principle the controller is able to simultaneously heat all the rooms, the optimal policy decides to heat on average 1.25 rooms in the safety problem and 1.64 rooms in the recovery case.

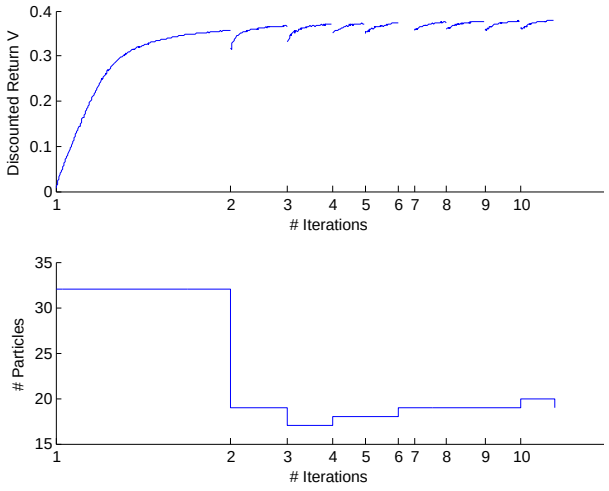


Fig. 11. Multi-room heating problem – Evolution of $V(\boldsymbol{\rho})$ (top) and of the number of particles (bottom) as a function of the iPGPE iterations, in the 5-rooms case.

Figure 11 reports for the same 5-room case the evolution over time of the performance index $V(\boldsymbol{\rho})$, as well as the number of particles per iteration. Notice that, while the improvements obtained through the iterative procedure are smaller than in the Car on the Hill example, the performance index is still improved throughout the process by suitably rearranging and retuning the particles. Notice also that the iPGPE starts with 32 particles, which are reduced to 19 already at the end of the first iteration.

Figure 12 illustrates some basic computational results of the two studied algorithms. The computational complexity of the ADP approach increases exponentially, both in terms of the CPU time and memory occupancy. On the contrary, the computational effort required by the iPGPE algorithm increases much less rapidly with the state space dimension.

Some additional data regarding the efficiency of the iPGPE algorithm are reported in Table III. The complexity of the iPGPE algorithm depends essentially on the number of particles, the number of policies drawn from the hyper-distribution, the number of histories generated for each policy and the number of steps per history. The number of simulation steps performed at each iteration of iPGPE is obtained by multiplying the last three values. Comparable results in terms of J are obtained for increasing values of n, provided that the number of histories is linearly incremented.

VII. CONCLUSIONS

This paper has investigated a policy search technique that applies to stochastic systems with continuous state and finite control spaces, and appears to scale favorably with state space size. The presented approach is based on the PGPE policy gradient technique, endowed with a novel policy parameterization using particles to describe entire areas of the state space

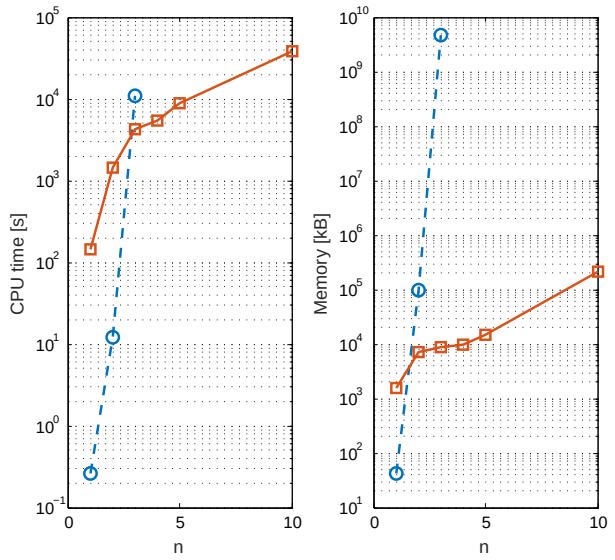


Fig. 12. CPU time (left) and memory occupancy (right) for ADP (circles) and iPGPE (squares) algorithms, for increasing problem dimension (n).

TABLE III
iPGPE ALGORITHM: COMPUTATIONAL COMPLEXITY.

n	# particles	J	# histories	# gradient estimates
1	2	0.3802	$1 \cdot 10^5$	391
2	9	0.3933	$2 \cdot 10^5$	822
3	12	0.3946	$3 \cdot 10^5$	931
4	15	0.3886	$4 \cdot 10^5$	1001
5	19	0.3793	$5 \cdot 10^5$	1092
10	14	0.3204	$10 \cdot 10^5$	1089

associated to the same action. By encapsulating the policy parameterization in the PGPE framework, it is possible to automatically learn both the positions of the particles in the state space and their associated actions. This ability comes at the price of a high number of parameters to be tuned, that scales proportionally to the number of particles. However, the number of samples required by the algorithm in order to estimate the gradient direction is not strictly related to the number of parameters (*i.e.*, particles) and does not increase exponentially with the state dimension.

The *a priori* definition of the particle set to be adapted turns out to be the greatest limitation of the basic PGPE scheme. To overcome this problem an iterative procedure is here suggested that interleaves the particle adaptation task with a particle selection one. More precisely, at each iteration a new particle is added and the whole set of particles retuned. After this tuning, redundant particles are detected and removed and a new iteration is performed.

The proposed iterative PGPE algorithm is applied to two benchmarks problems, to demonstrate its scalability properties and effectiveness. Novel particle insertion rules inspired by actor-critic methods are currently under study, the idea being to improve the iterative algorithm by inferring where the policy must be refined. Also, it will be valuable to investigate other optimization algorithms, like [17], [18], for tuning the particles hyperparameters.

ACKNOWLEDGMENT

The authors gratefully acknowledge financial support by the European Commission project UnCoVerCPS under grant number 643921.

REFERENCES

[1] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems," *Automatica*, vol. 44, no. 11, pp. 2724–2734, 2008.

[2] J. Lygeros and M. Prandini, "Stochastic hybrid systems: a powerful framework for complex, large scale applications," *European Journal of Control*, vol. 16, no. 6, pp. 583–594, 2010.

[3] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. New York (NY), USA: Wiley-Interscience, 1994.

[4] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3^{rd} ed. Athena Scientific, 2007, vol. II.

[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, March 1998.

[6] W. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, 2007.

[7] H. Zhang, D. Liu, Y. Luo, and D. Wang, *Adaptive dynamic programming for control: Algorithms and stability*. Springer Science & Business Media, 2012.

[8] L. Busoniu, R. Babuska, B. D. Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators*. Boca Raton (FL), USA: CRC Press, Inc., 2010.

[9] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, vol. 6, pp. 503–556, 2005.

[10] A. Antos, C. Szepesvári, and R. Munos, "Fitted Q-iteration in continuous action-space MDPs," in *Advances in neural information processing systems*, 2008, pp. 9–16.

[11] D. P. Bertsekas, "Approximate policy iteration: A survey and some new methods," *Journal of Control Theory and Applications*, vol. 9, no. 3, pp. 310–335, 2011.

[12] A. M. Farahmand, M. Ghavamzadeh, S. Mannor, and C. Szepesvári, "Regularized policy iteration," in *Advances in Neural Information Processing Systems*, 2009, pp. 441–448.

[13] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM Journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2003.

[14] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, "Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 10, pp. 1513–1525, 2013.

[15] H. Zhang, Y. Luo, and D. Liu, "Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints," *IEEE Transactions on Neural Networks*, vol. 20, no. 9, pp. 1490–1503, 2009.

[16] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits and Systems Magazine*, vol. 9, no. 3, pp. 32–50, 2009.

[17] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[18] G. Manganini, M. Pirotta, M. Restelli and L. Bascetta, "Following newton direction in policy gradient with parameter exploration", *IJCNN*, 2015.

[19] J. A. Bagnell and J. G. Schneider, "Autonomous helicopter control using reinforcement learning policy search methods," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 2001, pp. 1615–1620.

[20] J. Peters, S. Vijayakumar, and S. Schaal, "Reinforcement learning for humanoid robotics," in 3^{rd} *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids2003)*, Karlsruhe, Germany, Sept. 29–30 2003.

[21] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, April 2004, pp. 2619–2624.

[22] J. Kober, D. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[23] C. Szepesvari, *Algorithms for Reinforcement Learning*. Morgan and Claypool Publishers, 2010.

[24] L. Busoniu, D. Ernst, B. D. Schutter, and R. Babuska, "Cross-entropy optimization of control policies with adaptive basis functions," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 1, pp. 196–209, 2011.

[25] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer, 2004.

[26] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 6, pp. 1291–1307, 2012.

[27] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682–697, 2008.

[28] F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, and J. Schmidhuber, "Parameter-exploring policy gradients," *Neural Networks*, vol. 23, no. 4, pp. 551–559, 2010.

[29] T. Zhao, H. Hachiya, G. Niu, and M. Sugiyama, "Analysis and improvement of policy gradient estimation," *Neural Networks*, vol. 26, pp. 118–129, Feb. 2012.

[30] M. Pirotta, G. Manganini, L. Piroddi, M. Prandini, and M. Restelli, "A particle-based policy for the optimal control of Markov decision processes," in *IFAC World Congress 2014*, Cape Town, South Africa, August 2014.

[31] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3–4, pp. 229–256, 1992.

[32] D. P. Bertsekas and S. E. Shreve, *Stochastic Optimal Control: The Discrete-Time Case*. Athena Scientific, 1996.

[33] A. Okabe, B. Boots, and K. Sugihara, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. New York (NY), USA: John Wiley & Sons, Inc., 1992.

[34] H. Kimura and S. Kobayashi, "Reinforcement learning for continuous action using stochastic gradient ascent," *Intelligent Autonomous Systems (IAS-5)*, pp. 288–295, 1998.

[35] P. Thomas, "Bias in natural actor-critic algorithms," in *31^{st} International Conference on Machine Learning*, 2014, pp. 441–448.

[36] P. McMullen, "The maximum numbers of faces of a convex polytope," *Mathematika*, vol. 17, no. 2, pp. 179–184, 1970.

[37] B. Chazelle, "An optimal convex hull algorithm in any fixed dimension," *Discrete & Computational Geometry*, vol. 10, no. 1, pp. 377–409, 1993.

[38] A. Fehnker and F. Ivančić, "Benchmarks for hybrid systems verifications," in *Hybrid Systems: Computation and Control*, ser. LNCS 2993, R. Alur and G. J. Pappas, Eds. Springer Verlag, April 2004, pp. 326–341.

[39] A. Abate, S. Amin, M. Prandini, J. Lygeros, and S. Sastry, "Computational approaches to reachability analysis of stochastic hybrid systems," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Sciences, A. Bemporad, A. Bicchi, and G. Buttazzo, Eds. Berlin: Springer-Verlag, 2007, no. 4416, pp. 4–17.

[40] M. Prandini and L. Piroddi, "A self-recovery approach to the probabilistic invariance problem for stochastic hybrid systems," in 51^{st} *IEEE Conference on Decision and Control*, Maui (HI), USA, Dec. 2012, pp. 2096–2102.

[41] H. Zhang, C. Qin, L. Busoniu, B. Jiang and Y. Luo, "Online Adaptive Policy Learning Algorithm for H_∞ State Feedback Control of Unknown Affine Nonlinear Discrete-Time Systems", *IEEE T. Cybernetics*, vol. 4, no. 12, pp. 2706–2718, 2014.

[42] K. Senda, S. Hattori, T. Hishinuma, and T. Kohda, "Acceleration of Reinforcement Learning by Policy Evaluation Using Nonstationary Iterative Method", *IEEE T. Cybernetics*, vol. 44, no. 12, pp. 2696–2705, 2014.

Giorgio Manganini was born in Merate, Italy, in 1985. He received the M.S. degree in Computer Engineering in 2010 from the Politecnico di Milano. Between 2010 and 2012 he worked as a software engineer at an ICT international company. From November 2012 he is a Ph.D. student in the Systems and Control Division at the Dipartimento di Elettronica, Informazione e Bioingegneria at Politecnico di Milano. His research interests include Approximate Dynamic Programming, Machine Learning and Randomized Algorithms with applications to building energy efficiency.

Matteo Pirotta received the M.S. degree in computer science from the Politecnico di Milano, Milano, Italy, in 2012. He is actually a Ph.D. student in information technology at the Politecnico di Milano. His current research interests include machine learning (specially reinforcement learning) and robotics.

Marcello Restelli received the Dr.Eng. degree in computer science engineering and the Ph.D. degree in information engineering from Politecnico di Milano, Milan, Italy, in 2000 and 2004, respectively. Since 2008, he is an Assistant Professor with the Politecnico di Milano, where he has held various courses in the areas of machine learning and robotics. His research interests include machine learning, reinforcement learning, multi-armed bandit, and robot learning.

Luigi Piroddi (M'07) was born in London, U.K., in 1966. He received the "Laurea" degree in electrical engineering and the Ph.D. degree in computer science and control theory from the Politecnico di Milano, Milano, Italy, in 1990 and 1995, respectively. Between 1994 and 1999, he was a Professor of fundamentals of automation with the Universit degli Studi di Bergamo, Bergamo, Italy. From 1999 to 2004, he was an Assistant Professor with the Politecnico di Milano. Since 2004, he has been an Associate Professor with the same institution, where he holds various courses in the systems and control area. His research interests include nonlinear model identification, Petri nets, modeling, and control of manufacturing processes.

Maria Prandini received an M.S. degree in Electrical Engineering from Politecnico di Milano, Italy, in 1994 and a Ph.D. degree in Information Technology from the Universit degli Studi di Brescia, Italy, in 1998. From 1998 to 2000 she was a visiting postdoctoral researcher at the University of California at Berkeley. From December 2002 she held positions at Politecnico di Milano, Italy, where she is currently an Associate Professor of Automatic Control at the Dipartimento di Elettronica, Informazione e Bioingegneria.

Her research interests include stochastic hybrid systems, randomized algorithms, constrained control design, system abstraction and verification, distributed and stochastic optimization, system identification, and the application of control theory to air traffic management and power networks.

She currently serves on the editorial boards of IEEE Transactions on Control Systems Technology and Nonlinear Analysis: Hybrid Systems, and previously of European Journal of Control (2007 - 2013) and IEEE Transactions on Automatic Control (2009 - 2013). She is a member of the IFAC Technical Committee on Discrete Event and Hybrid Systems, of both the IEEE CSS and EUCA Conference Editorial Boards. In 2013 she became editor for Electronic Publications of the IEEE Control Systems Society, and she is responsible for the E-Letter on Systems, Control, and Signal Processing.