2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Policy Gradient Approaches for Multi–Objective Sequential Decision Making

Simone Parisi, Matteo Pirotta, Nicola Smacchia, Luca Bascetta and Marcello Restelli

Abstract— **This paper investigates the use of policy gradient techniques to approximate the Pareto frontier in Multi-Objective Markov Decision Processes (MOMDPs). Despite the popularity of policy gradient algorithms and the fact that gradient ascent algorithms have been already proposed to numerically solve multi-objective optimization problems, especially in combination with multi-objective evolutionary algorithms, so far little attention has been paid to the use of gradient information to face multi-objective sequential decision problems. Two different Multi-Objective Reinforcement-Learning (MORL) approaches, called *radial* and *Pareto following*, that, starting from an initial policy, perform gradient-based policy-search procedures aimed at finding a set of non-dominated policies are here presented. Both algorithms are empirically evaluated and compared to state-of-the-art MORL algorithms on three MORL benchmark problems.**

I. Introduction

Many real-world control problems (e.g., economic systems, water resource problems, robotic systems, just to mention a few) are characterized by the presence of multiple, conflicting objectives. Such problems are often modeled as Multi-Objective Markov Decision Processes (MOMDPs), where the concept of optimality typical of MDPs is replaced by the one of Pareto optimality, i.e., a set of policies providing a compromise among the different objectives. In the last decades, Reinforcement Learning (RL) [1] has established as an effective and theoretically-grounded framework that allows to solve single-objective MDPs whenever either no (or little) prior knowledge is available about system dynamics, or the dimensionality of the system to be controlled is too high for classical optimal control methods. Despite the successful developments in RL theory and a high demand for multi-objective control applications, Multi-Objective Reinforcement Learning (MORL) is still a relatively young and unexplored research topic.

MORL approaches can be divided into two main categories, based on the number of policies they learn [2].
Single-policy algorithms aim at finding the best policy that satisfies a preference among the objectives. The majority of MORL approaches belong to this category and differ in the way in which preferences are expressed.
Multiple-policy approaches aim at learning multiple policies in order to approximate the Pareto frontier. Building the exact frontier is generally impractical in real-world problems, the goal is thus to compute an approximation of the frontier that includes solutions that are accurate, evenly distributed and covering a range similar to the actual one [3].
There are many reasons behind the superiority of the multiple-policy methods: they allow a posteriori selection of the solution and encapsulate all the trade-offs among the multiple objectives. In addition, a graphical representation of the frontier can give better insight into the relationships among the objectives, improving the understanding of the problem and the selection of an appropriate solution.
Few examples of multiple-policy algorithms can be found in literature [4, 5, 6]. The most of such approaches are limited to deterministic policies that often result in scattered Pareto frontiers, while considering stochastic policies gives a continuous range of compromises among multiple objectives [7]. For a recent and complete survey of MORL, we refer the reader to [2, 8, 7].

Shelton [4, Section 4.2.1] was the pioneer both for the use of stochastic mixture policies and gradient approaches in MORL. He solved two well-known problems: simultaneous and conditional objectives maximization.
Consider the case of simultaneous objectives maximization, the algorithm starts with a mixture of policies obtained by applying standard RL techniques to each independent objective. The policy is then improved following a convex combination of the gradients in the policy space that are non-negative w.r.t. all the objectives. An approximation of the Pareto frontier is obtained by performing repeated search with different weights of the reward gradients.
Despite the progresses made by policy-gradient algorithms [9], their advantages in solving complex problems (e.g., problems with continuous-action domains and partially observable states) and the presence in the recent Multi-Objective Optimization (MOO) literature of many techniques exploiting gradient information, either standalone [10, 11] or in combination with evolutionary multi-objective algorithms [12], no further studies have followed [4] in applying policy gradients to MORL problems.

In this paper, we propose two policy-gradient based MORL approaches that, starting from some initial policies, perform gradient ascent in the policy-parameter space in order to determine a set of non-dominated policies.
In the first approach (called *radial* and described in Section IV-A), given the number p of Pareto solutions that are required for approximating the Pareto frontier, p gradient-ascent searches are performed, each one following a different (uniformly spaced) direction within the ascent simplex defined by the convex combination of single-objective gradients.

S. Parisi, M. Pirotta, N. Smacchia, L. Bascetta and M. Restelli are with the Department of Electronics, Information and Bioengineering, Politecnico di Milano, Piazza Leonardo da Vinci, 32, 20133, Milan, Italy (email: {luca.bascetta, matteo.pirotta, marcello.restelli}@polimi.it, {simone.parisi, nicola.smacchia}@mail.polimi.it).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

The second approach (called *Pareto-Following* and described in Section IV-B) starts by performing a single-objective optimization and then it moves along the Pareto frontier using a two-step iterative process: updating the policy parameters following some other gradient-ascent direction, and then applying a correction procedure to move the new solution onto the Pareto frontier.
An extensive empirical analysis of the two proposed approaches, and a comparison with some state-of-the-art MORL algorithms based on three multi-objective domains presenting different challenges, is provided in Section V.

II. Preliminaries

Multi-objective Markov Decision Processes (MOMDPs) are an extension of the MDP model, where several pairs of reward functions and discount factors are defined, one for each objective.
Formally, a MOMDP is described by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathbf{R}, \boldsymbol{\gamma}, D \rangle$, where $\mathcal{S} \subseteq \mathbb{R}^n$ is the continuous state space, $\mathcal{A} \subseteq \mathbb{R}^m$ is the continuous action space, $\mathcal{P}$ is a Markovian transition model where $\mathcal{P}(s'|s,a)$ defines the transition density between state s and s' under action a, $\mathbf{R} = [\mathcal{R}_1, \ldots, \mathcal{R}_q]^\intercal$ and $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_q]^\intercal$ are q-dimensional column vectors of reward functions $\mathcal{R}_i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ and discount factors $\gamma_i \in [0, 1)$, respectively, and D is the initial state distribution from which the initial state is drawn.
In MOMDPs, any policy π is associated to q expected returns $\mathbf{J}^\pi = \left[J_1^\pi, \ldots, J_q^\pi\right]$, where

$$J_i^\pi = E\left\{\sum_{t=0}^{H} \gamma_i^t r_i(t+1) | x_0 \sim D, \pi\right\},$$

being $r_i(t+1) = \mathcal{R}_i(s_t, a_t, s_{t+1})$ the i-th immediate reward obtained when state s_{t+1} is reached from state s_t and action a_t, and H the finite or infinite horizon.

Despite what happens in classical MDPs, in MOMDPs a single policy which dominates all the others usually does not exist; in fact, when conflicting objectives are considered, no policy can simultaneously maximize all the objectives. For these reasons, in Multi-Objective Optimization (MOO) a different dominance concept has been defined.

Definition 2.1: Policy π *dominates* policy π', which is denoted by $\pi \succ \pi'$, if:

$$\forall i \in \{1, \ldots, q\}, J_i^\pi \geq J_i^{\pi'} \ \wedge \ \exists i \in \{1, \ldots, q\}, J_i^\pi > J_i^{\pi'}.$$

Definition 2.2: If there is no policy π' such that $\pi' \succ \pi$, the policy π is Pareto-optimal.

In general, there are multiple Pareto-optimal policies. Solving a MOMDP is equivalent to determine the set of Pareto-optimal policies $\Pi^* = \left\{\pi \,|\, \nexists \pi', \pi' \succ \pi\right\}$, which maps to the so-called Pareto frontier $\mathcal{J}^* = \left\{\mathbf{J}^{\pi^*} | \pi^* \in \Pi^*\right\}$.[1]

[1]As done in [13], we suppose that local Pareto-optimal solutions that are not Pareto-optimal do not exist.

III. Multi-objective Policy Gradient

Consider now the problem of determining a policy that maximizes the expected discounted reward over a class of parametrized policies $\Pi_{\boldsymbol{\theta}} = \left\{\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \mathbb{R}^d\right\}$, where $\pi_{\boldsymbol{\theta}}$ is a compact notation for $\pi(a|s, \boldsymbol{\theta})$. In MOMDPs for each policy parameter $\boldsymbol{\theta}$, q gradient directions are defined

$$\nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}) = \int_{\mathbb{T}} \nabla_{\boldsymbol{\theta}} p(\tau|\boldsymbol{\theta}) r_i(\tau) \mathrm{d}\tau = \mathbb{E}\{\nabla_{\boldsymbol{\theta}} \log p(\tau|\boldsymbol{\theta}) r_i(\tau)\},$$

where $\tau \in \mathbb{T}$ (the space of all possible trajectories) is a trajectory drawn from density distribution $p(\tau|\boldsymbol{\theta})$ with reward $r_i(\tau) = \sum_{t=0}^{H} \gamma_i^t r_{i_{t+1}}$. Each direction $\nabla_{\boldsymbol{\theta}} J_i$ is associated to a particular discount factor-reward function pair $< \gamma_i, \mathcal{R}_i >$. In the following, we will see how to update the policy parameters on the basis of the q gradient vectors to produce an approximation of the Pareto frontier.

A. Ascent Directions

Given the q gradient vectors, we are interested in determining the directions that move the current solution toward the Pareto frontier. As these directions can only be generated as a convex combination of the gradient vectors, we focus our attention on the *ascent simplex*:

$$S(\boldsymbol{\lambda}, \boldsymbol{\theta}) = \sum_{i=1}^{q} \lambda_i \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}) \quad \text{s.t.} \quad \sum_{i=1}^{q} \lambda_i = 1, \ \forall i, \ \lambda_i \geq 0$$

Although any direction in the ascent simplex allows to approach the Pareto frontier, only a subset of such directions allows to *simultaneously* improve all the objectives.

In single objective optimization problems, if the objective is smooth, the level set associated to a particular parametrization $\boldsymbol{\theta}$ divides the parameter space into dominated and non-dominated areas. Locally, using a first order Taylor approximation, this is represented by a line perpendicular to the gradient (i.e., tangent to the level set) that defines two half spaces (Figure 1). In multi-objectives problems, the parameter space is partitioned into, at most, 2^q mutually exclusive directional cones. One cone simultaneously increases all the objectives (*ascent cone*), another cone simultaneously decreases all the objectives (*descent cone*), and $2^q - 2$ cones decrease at least one objective, while increasing another (*diversity cones*) [10] (Figure 2). *Ascent directions* are defined as the directions that lie in the ascent cone. Formally, a direction $\mathbf{l} = [l_1, \ldots, l_d]^\intercal \in \mathbb{R}^d$ is an ascent direction if

$$\mathbf{l} \cdot \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}) \geq 0 \qquad \forall i = 1, \ldots, q. \tag{1}$$

The *Pareto ascent cone* is the intersection of the ascent simplex with the ascent cone. When the solution $\boldsymbol{\theta}$ is sufficiently distant from the Pareto frontier, gradients are likely to be highly correlated and the directions that lie in the ascent simplex will also lie in the ascent cone (Figure 3). However, as the solution approaches the Pareto frontier, gradients become more and more conflicting and the width of the ascent cone decreases (Figure 4). A degenerate case is obtained when gradients are coplanar, in this case the ascent cone cannot be defined and the corresponding $\boldsymbol{\theta}$ is a (eventually local) Pareto-optimal solution.

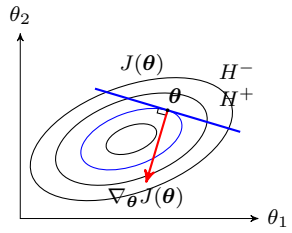

Fig. 1. Definition of the half spaces in a single objective problem. H^+ and H^- denote the non–dominated and dominated spaces, respectively.

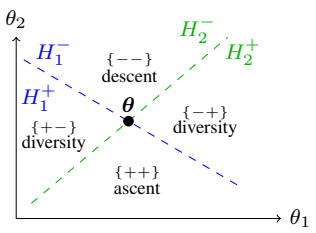

Fig. 2. The set of cones defined by $\boldsymbol{\theta}$ in a 2–objectives problem. Signs denote the change in the objectives J_1 and J_2.

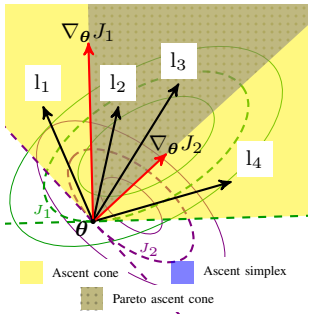

Fig. 3. The ascent cone and simplex in a 2–parameters, 2–objectives problem. The Pareto ascent cone equals the ascent simplex. l_2 and l_3 dominate l_1 and l_4.

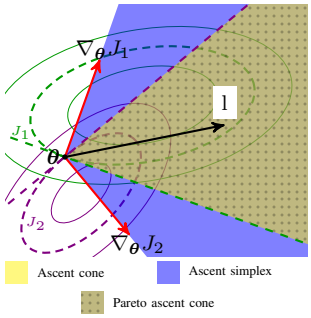

Fig. 4. The ascent cone and simplex in a 2–parameters, 2–objectives problem. The Pareto ascent cone equals the ascent cone. l represents a Pareto ascent direction.

B. Choosing the Best Ascent Direction

Determining the best Pareto ascent direction is an ill-posed problem, since multiple non-dominated solutions exist. The problem can be made unique by searching for the direction that maximizes the minimum improvement among all the individual objective improvements [10].

Let $\mathbb{R}_{++}$ be the set of strictly positive real numbers and $\boldsymbol{G}$ be the $(q \times d)$ Jacobian matrix with entries $\boldsymbol{G}_{ij} = \frac{\partial \mathbf{J}_i}{\partial \boldsymbol{\theta}_j}(\boldsymbol{\theta})$. If a solution $\boldsymbol{\theta}$ is not locally Pareto-optimal then there exists a direction $\mathrm{l} \in \mathbb{R}^d$ such that

$$\boldsymbol{G} \cdot \mathrm{l} \in \mathbb{R}^q_{++},$$

i.e., l is a Pareto ascent direction for the objective function $\mathbf{J}$. The following minimization problem finds the smallest (L2-norm) ascent directions l that maximizes the improvement of the individual objectives

$$\begin{aligned} \min_{\beta, \mathrm{l}} \quad & -\beta + \frac{1}{2}\|\mathrm{l}\|_2^2 \\ s.t. \quad & (\boldsymbol{G} \cdot \mathrm{l})_i \geq \beta \quad \forall i = 1, \ldots, q \end{aligned} \tag{2}$$

where the L2-norm is added as a regularization term. This Quadratic Programming (QP) problem, in $d+1$ variables with q inequality constraints, always admits a unique solution [14]. If $\beta = 0$, the solution is locally Pareto-optimal. On the other hand, by construction $\beta \geq 0$, and when β is strictly positive, l corresponds to a Pareto ascent direction. Moreover, when all the constraints are active, the objectives are improved by the same amount. In this case, assuming that the objectives share the same range, the solution is projected toward the Pareto front at $45°$ in objective space.

IV. Multi-Objective Policy Gradient Pareto Approximation

This Section introduces two algorithms, based on the policy gradient method, to compute an approximate Pareto frontier. In the first approach (called *radial*), given the number p of Pareto solutions required to approximate the Pareto frontier, p gradient-ascent searches are performed, each one following a different (uniformly spaced) direction within the ascent simplex. The second approach (called *Pareto-Following*), instead, performs a single-objective optimization and then moves along the Pareto frontier using a two-step iterative process: updating the policy parameters along another gradient-ascent direction and applying a correction to force the new solution onto the Pareto frontier.

A. Radial Algorithm (RA)

Following any Pareto ascent direction, a solution belonging to the Pareto frontier can be determined. Though an approximation of the Pareto frontier can be computed adopting an approach based on multiple starting points, solutions are unlikely to be evenly distributed on the frontier.

Consider the ascent simplex $S(\boldsymbol{\lambda}, \boldsymbol{\theta})$, following the extreme directions (individual steepest ascent directions) one converges to the solution that maximizes one objective, neglecting the others. Any other direction in the ascent simplex will simultaneously increase at least two objectives, ignoring the others. As a consequence, a uniform sampling of the ascent simplex results in evenly distributed directions pointing to the Pareto frontier with the goal of generating Pareto-optimal solutions as evenly distributed as possible.

Every direction in the ascent simplex intrinsically defines a preference over the objectives through $\boldsymbol{\lambda}$. Therefore, in order to generate evenly distributed directions in the ascent simplex, we need to uniformly sample the q-dimensional space $\mathbb{R}^q$ associated to the vector $\boldsymbol{\lambda}$.

Let $p \geq q$ be the granularity of the sampling. The algorithm starts with computing the individual gradients at a single point $\boldsymbol{\theta}^{(0)}$ and identifies the set $\{\mathrm{l}_i\}_{i=1}^p$ that defines the uniform partition of the ascent simplex. For every direction l_i, a new candidate solution $\boldsymbol{\theta}_i^{(1)}$ is generated according to the following equation

$$\boldsymbol{\theta}_i^{(1)} = \boldsymbol{\theta}^{(0)} + \alpha \mathrm{l}_i, \quad i = 1, \ldots, p.$$

Since every direction l_i is specified by a preference vector $\boldsymbol{\lambda}_i$, any point $\boldsymbol{\theta}_i^{(1)}$ can be associated to the preference vector $\boldsymbol{\lambda}_i$. At each iteration $t > 0$, each candidate solution $\boldsymbol{\theta}_i^{(t)}$ is associated to the preference vector $\boldsymbol{\lambda}_i$, and the following update rule is applied

$$\boldsymbol{\theta}_i^{(t+1)} = \boldsymbol{\theta}_i^{(t)} + \alpha \mathrm{l}_i^{(t)}, \qquad \mathrm{l}_i^{(t)} = S\left(\boldsymbol{\lambda}_i, \boldsymbol{\theta}_i^{(t)}\right).$$

Note that, p points are created at the first iteration, and are successively updated according to the associated preference vectors until they reach the Pareto frontier, as shown in Figure 5. The pseudo code is reported in Algorithm 1.

Algorithm 1 Radial Algorithm (RA)

Input: $\boldsymbol{\theta}^{(0)}$
$\{\boldsymbol{\lambda}_i\}_{i=1}^{p} \leftarrow$ uniform sampling of $\mathbb{R}^d$
$\mathbf{d}_i^{(0)} \leftarrow S\left(\boldsymbol{\lambda}_i, \boldsymbol{\theta}^{(0)}\right)$
for $i = 1, \ldots, p$ **do**
  $t = 1$
  while $\boldsymbol{\theta}_i^{(t-1)}$ not Pareto-optimal **do**
    $\boldsymbol{\theta}_i^{(t)} \leftarrow \boldsymbol{\theta}_i^{(t-1)} + \alpha \mathbf{d}_i^{(t-1)}$
    $\mathbf{d}_i^{(t)} \leftarrow S\left(\boldsymbol{\lambda}_i, \boldsymbol{\theta}_i^{(t)}\right)$
    $t \leftarrow t + 1$
  end while
end for

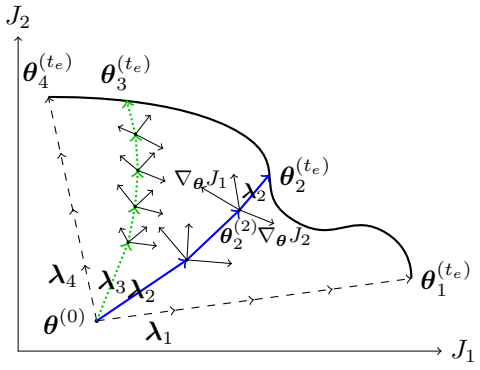

Fig. 5. Behaviour of the radial algorithm in a 2–objectives problem. Four preferences $\boldsymbol{\lambda}_i$ are selected from the ascent simplex in $\boldsymbol{\theta}^{(0)}$. At every step t the point $\boldsymbol{\theta}_i^{(t)}$ is updated according to the associated preference, until a Pareto–optimal point is reached.

B. Pareto-Following Algorithm

As far as we know, the Pareto-Following Algorithm is the first MORL algorithm that implements the idea of directed optimization on Pareto frontier. Similar approaches have been presented in MOO literature combined with genetic algorithms [10] or user preferences [15].
The concept of directed optimization on Pareto frontier is related to the ability of the search algorithm to reside on a neighbourhood of the frontier throughout the optimization process. Clearly, a first optimization stage that moves outside the frontier is required in order to reach a Pareto-optimal solution.

Moving the solution over the Pareto frontier improves some objectives and degrades other ones, according to the followed path. The main problem of directed algorithms is the choice of the search path. In the case of a 2-objective problem, the Pareto frontier is a line in the objective space, thus only two search directions exist. When $q \geq 3$, the Pareto frontier is represented by a multi-dimensional surface, and there are infinite directions that lie on the surface and along which a solution can be moved

The idea of the Pareto-Following Algorithm is to build a uniform approximation of the Pareto frontier by optimizing one objective at a time. In this way the choice of the search path is made unique for every parametrization $\boldsymbol{\theta}$. At every step, the Pareto-Following Algorithm accounts for two problems: improving the value of an objective and maintaining the point on the Pareto frontier.
Note that, starting from a solution on the Pareto frontier and following the steepest ascent direction of some objective may produce a dominated solution. To go straightly back to the frontier, a Pareto ascent direction can be followed. In particular, the minimal length projection, named also correction, is obtained by repeatedly solving the QP problem in (2). Figure 6 illustrates the procedure in a 2-objective problem.

The Pareto-Following Algorithm starts searching for an extreme point of the Pareto frontier, by optimizing the first objective $J_1(\boldsymbol{\theta})$. When such a Pareto-optimal solution is reached, it starts optimizing all the other objectives.
Let $\boldsymbol{\theta}^{[i]}$ be a solution on the Pareto frontier obtained by considering only the i-th objective in the last step

$$\boldsymbol{\theta}^{[i]} = \Gamma\left(\boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta})\right),$$

where $\boldsymbol{\theta}$ is a solution on the Pareto frontier and $\Gamma(\mathbf{x})$ is a function that given a candidate solution $\mathbf{x}$ returns a solution on the Pareto frontier (e.g., Γ is the repeated solution of the QP problem in (2)). Then, Pareto-Following Algorithm evaluates at most $(q - i + 1)$ ascent directions associated to the i-th and to the other objectives (see Figure 7 for a 3-objective example). This means that for any solution $\boldsymbol{\theta}^{[i]}$, the Pareto-Following Algorithm generates $(q-i+1)$ points $\left\{\overline{\boldsymbol{\theta}}^{[k]}\right\}$ such that

$$\overline{\boldsymbol{\theta}}^{[k]} = \Gamma\left(\boldsymbol{\theta}^{[i]} + \alpha \nabla_{\boldsymbol{\theta}} J_k\left(\boldsymbol{\theta}^{[i]}\right)\right) \qquad \forall k = i, \ldots, q. \quad (3)$$

Then it recursively applies the same procedure to any solution $\overline{\boldsymbol{\theta}}^{[k]}$, neglecting solution $\boldsymbol{\theta}^{[i]}$ if it is optimal w.r.t. the i-th objective. The pseudo code of the recursive Pareto-Following Algorithm is reported in Algorithm 2.

Algorithm 2 Pareto-Following Algorithm (PFA)

Input: the candidate solution $\boldsymbol{\theta}$, the index of the last gradient followed i, the points of the Pareto frontier $\mathcal{F}$

$c \leftarrow i$
if $\boldsymbol{\theta}$ is optimal w.r.t. the i-th objective **then**
  $c \leftarrow i + 1$
end if
for $k = c, \ldots, q$ **do**
  $\overline{\boldsymbol{\theta}} \leftarrow \Gamma\left(\boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} J_k\left(\boldsymbol{\theta}\right)\right)$
  $\mathcal{F} \leftarrow$ Pareto-Following Algorithm $\left(\overline{\boldsymbol{\theta}}, k, \mathcal{F}\right)$
end for
return $\mathcal{F} \cup \{\boldsymbol{\theta}\}$

V. Experiments

In this section, results related to the numerical simulations of the proposed algorithms, in continuous and discrete domains, are presented. In particular, the performance are compared against some existing algorithms [16, 17].

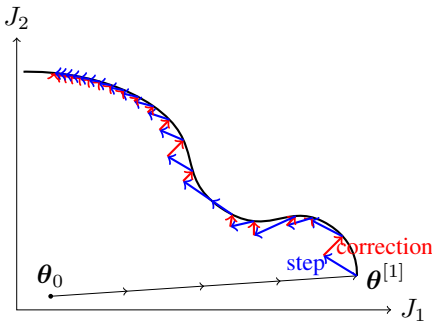

Fig. 6. Behaviour of the Pareto following algorithm in a 2–objectives problem. Once the first objective has been maximized (solid black line), the second objective is considered. After a step along the gradient (blue dashed line) is performed, the correction phase is performed (solid red line) in order to move the point on the frontier (solid curve).

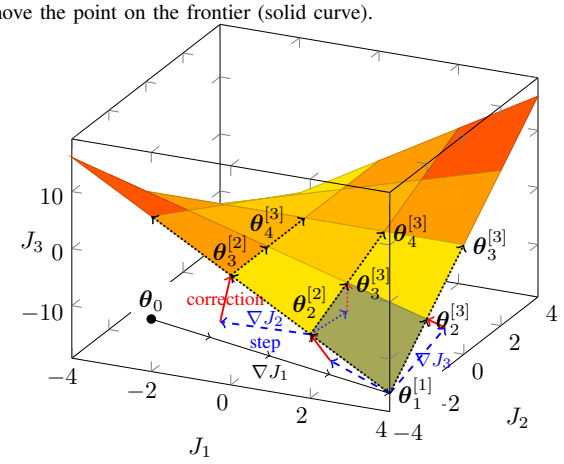

Fig. 7. Behaviour of the Pareto following algorithm in a 3–objectives problem. Blue densely dotted lines denote the step along the single gradient, while red lines denote the correction phase. Dotted lines denote the ideal steps on the frontier.

The Section is organized as follows: the domains are first described, then the numerical results are reported.

A. Domains

To illustrate the performance of the algorithms, we consider the following MDPs.

1) LQG: The first case of study is a discrete-time Linear-Quadratic Gaussian regulator (LQG) with multidimensional and continuous state and action spaces [18]. The LQG problem is defined by the following dynamics

$$s_{t+1} = As_t + Ba_t, \quad a_t \sim \mathcal{N}(K \cdot s_t, \Sigma)$$
$$r_t = -{s_t}^{\mathrm{T}} Q s_t - {a_t}^{\mathrm{T}} R a_t$$

where s_t and a_t are n-dimensional column vector ($n = m$), $A, B, Q, R \in \mathbb{R}^{n \times n}$, Q is a symmetric semidefinite matrix and R is a symmetric positive definite matrix. Dynamics are not coupled, that is, A and B are identity matrices. The policy is Gaussian with parameters $\boldsymbol{\theta} = vec(K)$, where $K \in \mathbb{R}^{n \times n}$. Finally, a constant covariance matrix $\Sigma = I$ has been chosen.

The LQG can be easily extended to account for multi-conflicting objectives. In particular, the problem of minimizing the distance from the origin w.r.t. the i-th axis has been taken into account, considering the cost of the action over the other axes

$$\mathcal{R}_i(s, a, s') = -s_i^2 - \sum_{i \neq j} a_j^2.$$

Since the maximization of the i-th objective requires to have null action on the other axes, objectives are conflicting.
As this reward formulation violates the positiveness of matrix R_i, we change the reward adding an ξ-perturbation

$$\mathcal{R}_i(s, a, s') = -(1-\xi)\left(s_i^2 + \sum_{i \neq j} a_j^2\right) - \xi\left(\sum_{j \neq i} s_j^2 + a_i\right),$$

where ξ is sufficiently small.

The values of the parameters used for all the experiments are the following ones: $\gamma = 0.9, \Sigma = I, \xi = 0.1$ and the initial state $s_0 = [10, 10]^{\mathrm{T}}$. For the estimation of the gradient a total of 100 episodes of 50 steps are collected.

2) Deep Sea Treasure: The deep sea treasure problem was first proposed in [19]. The environment is a 10×11 grid where some cells are not accessible. The agents can move in the four cardinal directions, with exception of the actions against the boundary that are feasible but do not change the position of the agent.
Unlike the LQG, this domain is episodic and each episode ends when the agent finds a treasure. Multiple treasures with varying values are placed on the grid, and there are two objectives: to maximize the value of the treasure and to minimize the time spent to reach it. For the first objective, the agent gets an immediate reward that equals the value of the cell it moves in (empty cells have 0 value), for the second objective it receives a penalty of -1 at each step. The initial state is always $(1, 1)$ and the discount factor is set to 1.
Since the problem has discrete states and actions a Gibbs policy is used

$$\pi(a|s) = \frac{e^{\tau f(s,a)}}{\sum_{a' \in \mathcal{A}} e^{\tau f(s,a')}},$$

where $f(s, a) = \phi(s, a)^{\mathrm{T}} \boldsymbol{\theta}$ is a preference function over state-action pairs, and $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ is the set of basis functions.

For the estimation of the gradient and for the evaluation of the policy a total of 500 episodes by 50 steps were used.

3) Water Reservoir: A water reservoir can be modelled as a MOMDP with a continuous state variable s representing the water volume stored in the reservoir, a continuous action a that controls the water release, a state-transition model that depends also on the stochastic reservoir inflow ϵ, and a set of conflicting objectives. For a complete description of the problem, the reader can refer to [17].

In this work we consider three objectives: flooding along the lake shores, irrigation supply and hydropower supply. The immediate rewards are defined by

$$\mathcal{R}_1(s_t, a_t, s_{t+1}) = -\max(h_{t+1} - \bar{h}, 0)$$
$$\mathcal{R}_2(s_t, a_t, s_{t+1}) = -\max(\bar{\rho} - \rho_t, 0)$$
$$\mathcal{R}_3(s_t, a_t, s_{t+1}) = -\max(\bar{e}_t - e_{t+1}, 0).$$

TABLE I
EXPERIMENTAL SETUPS

	LQG				Deep		Water	
	ex. PFA	PFA	ex. RA	RA	PFA	RA	PFA	RA
Step to reach the frontier	0.1	0.1	0.1	0.1	0.1	0.5	4	4
Step on the frontier	$5 \cdot 10^{-4}t$	$5 \cdot 10^{-4}t$	-	-	0.1	-	4	-
Step for correction	0.1	0.1	-	-	0.1	-	4	-
N. of simplex directions	-	-	101	101	-	66	-	21

where $h_{t+1} = {}^{s_{t+1}}/_{S}$ is the reservoir level (in the following experiments $S = 1$), $\bar{h}$ is the flooding threshold ($\bar{h} = 50$), $\rho_t = \max(\underline{a}_t, min(\bar{a}_t, a_t))$ is the release from the reservoir, $\bar{\rho}$ is the water demand ($\bar{\rho} = 50$), $\bar{e}_t$ is the demand for electricity ($\bar{e}_t = 4.36$) and e_{t+1} is the electricity production:

$$e_{t+1} = \psi \; g \; \eta \; \gamma_{H_20} \; \rho_t \; h_{t+1},$$

where $\psi = {}^{10^{-6}}/_{3.6}$ is a dimensional conversion coefficient, $g = 9.81$ the gravitational acceleration, $\eta = 1$ the turbine efficiency and $\gamma_{H_20} = 1,000$ the water density.
$\mathcal{R}_1$ denotes the negative of the cost due to the flooding excess level, $\mathcal{R}_2$ is the negative of the deficit in the water supply and $\mathcal{R}_3$ is the negative of the deficit in hydropower production.

Due to the fact that the transition function limits the action in the range of admissible values ($a_t \in [\underline{a}_t, \bar{a}_t]$), there are infinite policies with equal performance that allow the agent to release more than the reservoir level or less than zero. A penalty term in the reward $p = -\max(a_t - \bar{a}_t, \underline{a}_t - a_t)$ is thus introduced, in order to prevent such infeasible actions. Moreover, in order to be able to compare the results with the original work, the penalty is considered only in the learning phase and not in the evaluation of the policy.

Like in the original work, the discount factor is set to 1 for all the objectives. However, different settings are used for learning and evaluation. In the learning phase $1,000$ episodes by 10 steps with initial state $s_0 \sim Unif(0, 160)$ are used, while the evaluation phase exploits 100 episodes by 100 steps with initial state drawn from a finite set.

Since the problem is continuous we exploit a Gaussian policy model

$$\pi(a|s, \boldsymbol{\theta}) = \mathcal{N}\left(\phi(s)^{\mathrm{T}}\kappa, \sigma\right),$$

where $\phi : \mathcal{S} \rightarrow \mathbb{R}^{d-1}$ are the basis functions. In order to prevent the variance from becoming negative, the parametrization presented by [20] is presented, where σ is represented by a logistic function parametrized by ω

$$\sigma = \frac{\tau}{1 + e^{-\omega}},$$

where τ is the maximum variance allowed (in the experiments $\tau = 50$) and $\omega \in \mathbb{R}$. Unlike the LQG, the variance σ is variable and will be learned by the agent, leading to an overall set of parameters $\boldsymbol{\theta} = [\kappa, \omega]^{\mathrm{T}}$.

Since the optimal policies for the objectives are not linear in the state variable, a radial basis approximation is used: $\phi(s) = \left[e^{-\|s - c_i\|_2^2/w_i}\right]_{i=1}^{d-1}$, where the centres c_i are placed at 0, 50, 120 and 160, and the widths are 50, 20, 40 and 50.

The complete set of parameters used in the learning is shown in Table I.

B. Numerical results

For each of the previous domains, simulation results are reported here, comparing them with the results obtained using other algorithms. The gradient is estimated as in [18].

1) LQG: The LQG problem is useful because the exact gradient is known and the precision of the proposed algorithms can be analysed using both the exact and the estimated gradient. Further, since the Pareto frontier is convex, a weighted sum method can be exploited in order to obtain a reference frontier used to compare the two approaches.

In order to obtain a robust solution of the problem (2), gradients are normalized in the correction phase. Normalization of the objectives is not necessary since the initial state s_0 ensures they have the same order of magnitude.

Figure 8(c) shows the approximated front obtained with the Pareto-Following algorithm. It can be noticed that the approximation is not completely uniform as it lacks solutions near the optimum $\boldsymbol{\theta}_2^*$ for the second objective (bottom right corner) and it is more dense near the optimum $\boldsymbol{\theta}_1^*$ for the first objective (top left corner). This is explained by the fact that the algorithm first optimizes J_2 and then J_1. Starting from $\boldsymbol{\theta}_2^*$, i.e., the optimum w.r.t. to J_2, the magnitude of the gradient $\nabla_{\boldsymbol{\theta}} J_1(\boldsymbol{\theta})$ w.r.t. the single objective J_1 decreases as the solution approaches the optimum $\boldsymbol{\theta}_1^*$, and this produces smaller steps. Better approximations can be obtained adapting the learning rate with time [21], but this does not ensure to get a uniform approximation of the frontier.

Figure 8(a) shows the approximated Pareto frontier obtained with the Radial Algorithm. Compared with the previous one, this is more uniformly distributed and more precise, but has more solutions concentrated near the knee of the front. Radial Algorithm proved to be faster since it can use a higher learning rate and still obtain a uniform front. It is interesting that, even starting near to the center of the front, Radial Algorithm is still able to reach its extremity.

Figure 8(b) and 8(d) show the same algorithms using the estimated version of the gradient [18]. The former figure reports the frontier obtained with the Radial Algorithm. Compared to the previous frontier, solutions are more sparse.

Concerning computational costs, the exact version in significantly faster (e.g., for the PFA $23s$ versus $3,078s$) than the approximate version because it does not require to perform sampling in order to compute the gradient.

2) Deep Sea Treasure: This problem is interesting because the Pareto frontier obtained using deterministic policies is concave [19]. Thus, methods that exploits weighted sum, e.g., [17], are not able to correctly approximate the frontier. However, using stochastic policies, e.g., mixture policies, all the deterministic policies, except the extrema, are dominated. The real Pareto frontier is obtained as convex combination of the optimal policies of the single objectives. Since the proposed algorithms are able to exploit stochastic policies, they are able to correctly approximate the real frontier.

Figure 9 shows the frontiers obtained by the Pareto-Following algorithm using different approximations of the preference function $f(s, a)$. The algorithm begins with a

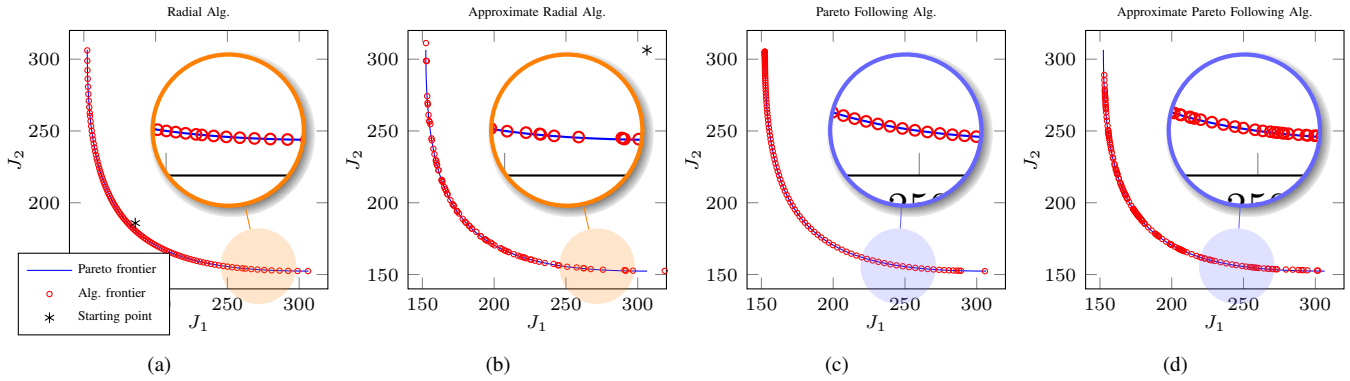

Fig. 8. Approximate Pareto frontier obtained by RA and PFA in the LQG problem. Figures (a) and (b) represents the RA approximation in the exact and approximate scenario. Initial parametrization are $\boldsymbol{\theta}_0 = [-0.5, 0, 0, -0.5]^{\mathrm{T}}$ and $\boldsymbol{\theta}_0 = [-0.17, 0, 0, -0.17]^{\mathrm{T}}$, respectively. PFA approximation is reported in figures (c) and (d).

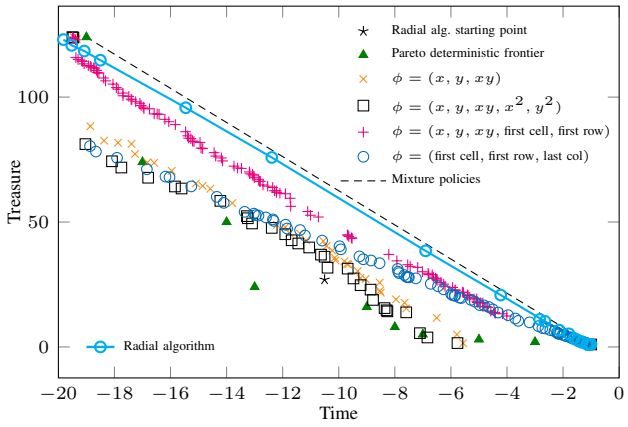

Fig. 9. Comparison between frontiers obtained with different features using FPA and RA. Since the environment is a grid the state is represented by two variables x (rows) and y (columns).

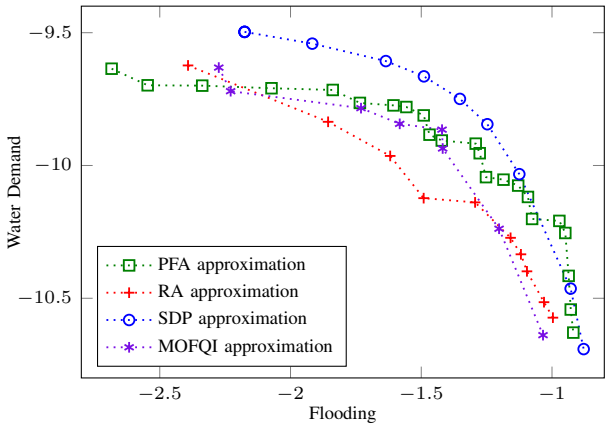

Fig. 10. Approximate Pareto frontier for the 2-objectives water reservoir.

completely random policy ($\boldsymbol{\theta} = \mathbf{0}$) and the first objective to be optimized is the time. Exploiting gradient ascent a deterministic policy is found, that forces the agent to immediately perform a step down in order to reach the first treasure. Then, the treasure value is optimized starting from such policy. Since the policy is deterministic a randomization is necessary for the correct estimation of the gradient. The randomization is performed by reducing the temperature τ of the Gibbs policy proportionally to its entropy H (evaluated during the sampling): $\tau_{new} = \tau_{old} \cdot H$. This explains why the algorithm finds few solutions near the optimal deterministic policy for the first objective (bottom right corner). We faced the same problem starting from the inverse ordering of the objectives because the optimal policy for the treasure value is also deterministic (at least in cell $(1, 1)$ in order to avoid the choice of the worse treasure).

Figure 9 also shows a front obtained with the Radial Algorithm (solid bullet line) with only one set of features (first cell, first row without first and last cell and last column), since with the other ones the agent was not able to learn, or solutions were concentrated only near the optimal single objective solutions. Unlike for the LQG experiment, the front is not uniformly distributed, even if directions were chosen uniformly in the simplex of the gradients. Furthermore, even if we are able to use a higher learning rate, the algorithm performs significantly slower than the Pareto-Following algorithm and obtains less solutions (dominated solutions are removed in post processing). Since the randomization of the policy is not necessary, solutions are not sparse near the single objective optimums.

3) Water Reservoir: To evaluate the effectiveness of the proposed algorithms we have analysed their performances against the solution found by Stochastic Dynamic Programming (SDP) using the weighted sum method [17], multi-objective FQI [17] (using $20,000$ samples with a dataset of $200,000$ tuples) and the standard FQI. Both MOFQI and FQI have been trained using $10,000$ samples with a dataset of $50,000$ tuples for the 2–objectives problem and $20,000$ samples with a dataset of $500,000$ tuples for the 3–objectives problem. FQI exploits a scalarization of the objectives according to the SDP weights. Since the objectives $\mathbf{J}$ have different magnitude, they have been normalized.

Figure 10 shows a graphical representation of the Pareto points obtained by the algorithms when only the first two objectives are considered. While a graphical comparison is meaningful in 2-objectives problem, such representation is meaningless or infeasible when $q > 2$.

For the comparison of the algorithms, we consider an extension of a previously defined metric [17] that measures the loss of an approximation of the Pareto frontier and

TABLE II

WATER RESERVOIR: ALGORITHM COMPARISON

Algorithm	Loss (2-obj.)	Loss (3-obj.)
Radial	0.0945 ± 0.0049	0.0123 ± 0.0011
Pareto following	0.0811 ± 0.0072	0.0162 ± 0.0021
MOFQI [17]	0.1870 ± 0.0090	0.0540 ± 0.0061
FQI [16]	0.1910 ± 0.0100	0.0292 ± 0.0010

TABLE III

WATER RESERVOIR: COMPUTATIONAL COSTS

	Water 2-obj.		Water 3-obj.	
	PFA	RA	PFA	RA
Time	$408 \pm 52s$	$2,789 \pm 204s$	$13,128 \pm 286s$	$9,414 \pm 1040s$
#Iterations	347.5 ± 46.2	974.5 ± 42.7	$2,096 \pm 330.2$	$2,441 \pm 125.9$
#Solutions	16.9 ± 1.5	10.4 ± 0.5	154.3 ± 10.5	54.4 ± 0.8

the frontier itself. Since the optimal Pareto frontier $\mathcal{J}^*$ is not available, we exploit the solution of the SDP as reference frontier and the loss function is approximated using the discrete set of weights exploited to compute the SDP frontier [17]. Furthermore, given that it is not possible to associate weights with policies found by the gradient algorithms, for each weight of the SDP, the policy that minimizes the loss function is selected.

Table II reports the loss achieved by the algorithms w.r.t. the SDP approximation. Gradient algorithms result to be more effective when the number of objectives increases. Numerical results are reported in Table III. All the results are averaged over 10 runs.

VI. CONCLUSIONS

Multi-objective problem are relevant in many real world applications. However, few attention has been posed on MOMDPs in the field of RL. This paper has investigated multi-objectives policy gradients methods that are able to overcome limitations of weighted sum methods. The key issue is how to compute the update in an appropriate ascent or diversity direction. The Radial Algorithm defines directions in the ascent simplex and computes p solutions according to them. The Pareto-Following algorithm, instead, is able to navigate through nearly Pareto-optimal solutions.

Subjects for further investigation include assessing how it is possible to guarantee a uniform covering of the Pareto frontier and how close are the solutions to the real frontier.

REFERENCES

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

[2] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker, "Empirical evaluation methods for multiobjective reinforcement learning algorithms," *Machine Learning*, vol. 84, no. 1-2, pp. 51–80, 2011.

[3] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE T EVOLUT COMPUT*, vol. 7, no. 2, pp. 117–132, 2003.

[4] C. R. Shelton, "Importance sampling for reinforcement learning with multiple objectives," Ph.D. dissertation, Massachusetts Institute of Technology, August 2001.

[5] D. J. Lizotte, M. Bowling, and S. A. Murphy, "Linear fitted-q iteration with multiple reward functions," *J MACH LEARN RES*, vol. 13, pp. 3253–3295, 2012.

[6] A. Castelletti, F. Pianosi, and M. Restelli, "A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run," *Water Resources Research*, vol. 49, no. 6, pp. 3476–3486, 2013.

[7] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *JAIR*, vol. 48, pp. 67–113, 2013.

[8] C. Liu, X. Xu, and D. Hu, "Multiobjective reinforcement learning: A comprehensive overview," *IEEE T SYST MAN CY C*, vol. PP, no. 99, pp. 1–13, 2013.

[9] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE T SYST MAN CY C*, vol. 42, no. 6, pp. 1291–1307, 2012.

[10] M. Brown and R. E. Smith, "Directed multi-objective optimization," *Int. J. Comput. Syst. Signal*, vol. 6, no. 1, pp. 3–17, 2005.

[11] S. N. Kazuhiro Izui, Takayuki Yamada, "A gradient-based multiobjective optimization technique using an adaptive weighting method," *10th World Congress on Structural and Multidisciplinary Optimization*, 2013.

[12] O. Schütze, G. Sanchez, and C. A. C. Coello, "A new memetic strategy for the numerical treatment of multi-objective optimization problems," in *GECCO*. ACM, 2008, pp. 705–712.

[13] K. Harada, J. Sakuma, and S. Kobayashi, "Local search for multiobjective function optimization: pareto descent method," in *GECCO*, 2006, pp. 659–666.

[14] J. Fliege and B. F. Svaiter, "Steepest descent methods for multicriteria optimization," *Math. Meth. of OR*, vol. 51, no. 3, pp. 479–494, 2000.

[15] V. Sevastyanov, "Directed optimization on pareto frontier," *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, 2010.

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

[17] F. Pianosi, A. Castelletti, and M. Restelli, "Tree-based fitted q-iteration for multi-objective markov decision processes in water resource management," *Journal of Hydroinformatics*, vol. 15, no. 2, pp. 258–270, 2013.

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

[19] P. Vamplew, J. Yearwood, R. Dazeley, and A. Berry, "On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts," in *Advances in Artificial Intelligence*. Springer, 2008, pp. 372–378.

[20] H. Kimura and S. Kobayashi, "Reinforcement learning for continuous action using stochastic gradient ascent," in *IAS-5*, 1998, pp. 288–295.

[21] M. Pirotta, M. Restelli, and L. Bascetta, "Adaptive step-size for policy gradient methods," in *NIPS 26*. Curran Associates, Inc., 2013, pp. 1394–1402.