BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **12**, No 3

Sofia • 2012

Combining Local and Global Direct Derivative-free Optimization for Reinforcement Learning

Matteo Leonetti, Petar Kormushev*, Simone Sagratella***

**Department of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego, 30, 16163 Genova*
***Department of Computer, Control, and Management Engineering, Sapienza University of Rome, via Ariosto 25, 00185, Roma*
Emails: matteo.leonetti@iit.it petar.kormushev@iit.it sagratella@dis.uniroma1.it

Abstract: *We consider the problem of optimization in policy space for reinforcement learning. While a plethora of methods have been applied to this problem, only a narrow category of them proved feasible in robotics. We consider the peculiar characteristics of reinforcement learning in robotics, and devise a combination of two algorithms from the literature of derivative-free optimization. The proposed combination is well suited for robotics, as it involves both off-line learning in simulation and on-line learning in the real environment. We demonstrate our approach on a real-world task, where an Autonomous Underwater Vehicle has to survey a target area under potentially unknown environment conditions. We start from a given controller, which can perform the task under foreseeable conditions, and make it adaptive to the actual environment.*

Keywords: *Reinforcement learning, policy search, derivative-free optimization, robotics, autonomous underwater vehicles*

1. Introduction

Reinforcement Learning (RL) is the learning paradigm in which an agent improves its behavior on a given task by exploring the environment through trial and error. Its mathematical formulation consists in the maximization of a *reward function*, which measures the agent's performance. As an optimization problem it is a peculiar one, because many aspects of the task on which the performance must be optimized are

unknown to the agent. In particular, the function to be maximized has often no analytical expression, and must be sampled by acting in the environment. Each function evaluation costs the agent resources, which is especially critical in robotics. Robotic agents act in real-world environments, consume power, are subject to wearing and tearing, and work in real time. This imposes a careful use of trials, favoring on-line learning to off-line, batch, learning.

Policy gradient methods have largely been used in robotics [17] owing to characteristics that make them particularly suitable. In high-dimensional domains, finding good value function representations is difficult, while structuring the policy may come naturally from the task. Encoding the policy appropriately, policy search methods can benefit from previous knowledge. Usually fewer parameters are needed than with value function approximators, and the methods have strong theoretical foundations. Moreover, policies can be changed gradually, so that the behavior of the robot can be ensured withing the operation limits. Nonetheless, estimating the gradient is still a costly procedure, that requires many trials around the current policy. Noisy environments can make gradient estimation extremely difficult, either affecting the estimate greatly, or requiring a large number of trials.

While the local aspect of policy-gradient methods is favorable to robots, it is also a double-edged sword. On the one hand, policy-gradient approaches a local optimum slowly and smoothly, producing behaviors that don't deviate sharply from one another. On the other hand, there's no guarantee about the presence of global optima elsewhere. A broader exploration can be performed in simulation, but the so called *reality gap* [4, 11], the inevitable difference between the real and simulated domains, makes learning in simulation more brittle than in the actual environment.

In this paper, we explore direct derivative-free optimization algorithms for policy search in episodic reinforcement learning. Reinforcement learning imposes particular constraints on optimization algorithms. For instance, industrial applications are usually parallelized, while this is not possible for on-line RL. We combine the benefits of local methods for on-line learning, with a global search in simulation to obtain good starting points. While being able to benefit from all the advantages mentioned for policy-gradient algorithms, local derivative-free ones do not have the burden to estimate the gradient, which is particularly relevant with noisy reward functions. Derivative-free algorithms used in RL so far are pure global optimization algorithms, for the largest part evolutionary [1, 9]. We chose to combine two different derivative-free algorithms: a global stochastic search [3], and a line search [16]. The former allows to identify a policy in whose neighborhood the global optimum is most likely to be. The latter then refines this policy, going through its neighborhood without attempting to estimate the gradient. Learning in simulation is therefore coarse-grained, for the environment is only an approximation of the one the agent will really face. A more refined learning is performed on-line on the actual environment, in a local and smooth fashion. Different combinations of global and local methods are possible, and more sophisticated methods can be employed.

We implemented our approach on a real-world problem, where an Autonomous Underwater Vehicle (AUV) has to survey an area under unpredictable disturbances.

The AUV is equipped with a controller able to perform the task under normal operation conditions. When unexpected environment disturbances make it fail, the agent learns a policy that corrects the given controller. The experiments have been performed on a realistic simulator, taking into account real battery life. The method introduced in this paper proves to be able to learn a policy to perform the task in a few tens of trials, taking a time largely within the robot mission duration.

2. Background and notation

In this section we provide the background behind policy search in RL.

A Markov Decision Process is a tuple $MDP = \langle S, A, T, \rho \rangle$ where: S is a set of *states*, A is a set of *actions*, $T : S \times A \times S \to [0,1]$ is the transition function. $T(s,a,s') = Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that the current state changes from s to s' by executing action a. $\rho : S \times A \times \mathbb{R} \to [0,1]$ is the reward function. $\rho(s,a,r) = Pr(r_{t+1} = r | s_t = s, a_t = a)$ is the probability to get a reward r from being in state s and executing action a. In our setting both states and actions are continuous, while time is discrete.

The behavior of the agent is represented as a function $\pi : S \times A \to [0,1]$ called a (stationary) *policy*, where $\pi(s,a)$ is the probability of selecting action a in state s. A policy π and an initial state s_0 determine a probability distribution d^π over the possible sequences $\omega = (\langle s_t, a_t, r_{t+1} \rangle, t \geq 0)$. Given such a sequence, we define the *cumulative discounted reward* as

$$R = \sum_{t \geq 0} \gamma^t r_{t+1}$$

where $0 < \gamma \leq 1$ is the *discount factor*. The reward is accumulated by executing a in s and following π thereafter.

A state is said to be *absorbing* if once entered cannot be left. An MDP with absorbing states is said to be *episodic*. In the rest of this paper we focus on episodic MDPs, where the duration of the episode is given by the first time step in which an absorbing state is entered. Therefore, from now on we set $\gamma = 1$ and sum over an unknown finite duration T.

Policies are parametrized through a vector θ. This allows to add structure to the policy, which is then changed by modifying the parameters. The value of a policy π is defined, through its parameters, as:

$$J(\theta) = \int_S d^\pi(s) \int_A \pi(s,a) r(s,a) ds da \tag{2}$$

where $r(s,a)$ is extracted from $\rho(s,a,\cdot)$.

Policy gradient methods try to estimate the gradient of the function in Equation 2, in order to make an update to the parameters along it:

$$\theta_{i+1} = \theta_i + \alpha \nabla_\theta J(\theta_i) \tag{3}$$

Direct derivative-free algorithms, on the other hand, perform hill-climbing updates without estimating the gradient. The function our method will maximize, over a finite horizon and from a single initial state, is:

$$J(\theta) = \sum_{t=0}^{T} \int_S d_t^{\pi}(s) \int_A \pi(s,a) r(s,a) ds da \tag{4}$$

3. Related work

A large number of policy search methods have been developed in the field, although only few of them have found successful application in robotics [14]. Most of those are local algorithms, the vast majority of which is gradient-based. Episodic REINFORCE [26] is one of the earliest gradient-based learning algorithm for episodic RL. As previously mentioned, gradient-based methods estimate the gradient at a given policy, and perform an update along its direction according to Equation 3. More recent policy gradient [22] RL algorithms are G(PO)MDP [2], natural policy gradient [12], and natural actor-critic [19] methods. All of the mentioned policy gradient methods differ in how the gradient is estimated. A different approach, based on Expectation Maximization (EM), has been proposed and applied to several algorithms, the most recent of which is PoWER [18, 14]. In EM-based methods, the lower bound on the expected cumulative return is maximized. Another notable local algorithm is Policy Improvements with Path Integrals (PI^2) [23], which belongs to the category of path integrals method applied to stochastic optimal control.

The algorithm we are going to apply for local search is a line search, as in gradient-based methods, but the direction used is not the gradient. Therefore, it can avoid the extremely costly operation of estimating it. Nonetheless, the algorithm is guaranteed, under appropriate conditions, to converge to the optimal solution.

On the side of global algorithms, a huge range of methods has been employed. Simulated Annealing [13] and Ant Colony Optimization [6] are well-known global methods, although the most studied one is the set of evolutionary approaches, in particular genetic algorithms [8]. Tabu search [7] attempts to escape local minima by performing temporary worsening moves, and using history to prevent steps that would lead back to the minimum just left. Last, we mention RLPF [15] and pattern search [24] belonging to the class of global random optimization algorithms used in RL. The global algorithm we are going to use is a clustering algorithm similar to pattern methods, where the search is random and controlled. In our random search, an initially random point cloud tends to gather around the global optimum, as will be described in Section 4.1.

4. Derivative-free algorithms

It is well known that extensive useful information is contained in the derivatives of any function one wishes to optimize. For a variety of reasons, however, there have

always been many instances where (at least some) derivatives are unavailable or unreliable. Nevertheless, under such circumstances it may still be desirable to carry out optimization. Consequently, a class of nonlinear optimization techniques called derivative-free methods has been developed in the literature. In the following, we describe the methods we are going to combine, highlighting the features that make them suitable for the different learning phases. The first method, a random search, was developed in the original paper with both a global and a local search. We removed the local part, and substituted it with the line search described in Section 4.2. This line search fits better with the RL setting, according to the arguments on locality introduced in Section 1. In the following, the algorithms will be described for minimization problems, in order to comply with the literature on optimization. Since in our RL implementation we maximize the reward (instead of minimizing costs) the sign of the reward function will simply be inverted.

4.1. Random global serach

Many algorithms have been proposed in the literature to solve unconstrained global optimization problems [10, 25]. In this paper, however, we are interested to tackle the particular difficult case of a problem in which:

- the evaluation of the objective function is very expensive;
- the values of the objective function can be affected by the presence of noise;
- the derivatives of the objective function are not available.

The method we use is a version of the Controlled Random Search Algorithm [20] in which the efficiency is improved by using a weighted centroid and a weighted reflection [3].

4.1.1. Controlled Random Search Algorithm

Data: a positive integer $m \geq n+1$, $\varepsilon > 0$

Step 0. Set $k = 0$ and compute the initial set:

$$S^k = \{\theta_1^k, \ldots, \theta_m^k\}$$

where the points θ_i^k, $i = 1, \ldots, m$ are chosen at random over a box D; evaluate J at each point θ_i^k, $i = 1, \ldots, m$.

Step 1. Determine the points θ_{max}^k, θ_{min}^k and the values J_{max}^k, J_{max}^k such that:

$$J_{max}^k = J(\theta_{max}^k) = \max_{\theta \in S^k} f(\theta)$$
$$J_{min}^k = J(\theta_{min}^k) = \min_{\theta \in S^k} f(\theta);$$

if $J_{max}^k - J_{min}^k \leq \varepsilon$ then STOP

Step 2. Choose at random $n+1$ points $\theta_{i_0}^k, \theta_{i_1}^k, \ldots, \theta_{i_n}^k$ over S^k, where

$$J(\theta_{i_0}^k) \geq J(\theta_{i_j}^k), \quad j = 1, \ldots, n;$$

determine the centroid

$$c^k = \sum_{j=0}^{n} w_j^k \theta_{i_j}^k,$$

and determine the trial point $\bar{\theta}^k$ given by

$$\bar{\theta}^k = c^k - \alpha^k(\theta^k_{i_0} - c^k),$$

where

$$w^k_j = \frac{\eta^k_j}{\sum_{r=0}^n \eta^k_r}, \quad \eta^k_j = \frac{1}{J(\theta^k_{i_j}) - J^k_{min} + \phi^k}, \quad \alpha^k = 1 - \frac{J(\theta^k_{i_0}) - \sum_{j=0}^n w^k_j J(\theta^k_{i_j})}{J^k_{max} - J^k_{min} + \phi^k},$$

and

$$\phi^k = n\frac{(J^k_{max} - J^k_{min})^2}{J^0_{max} - J^0_{min}};$$

if $\bar{\theta}^k \notin D$ go to Step 2; otherwise compute $J(\bar{\theta}^k)$.

Step 3. If $J(\bar{\theta}^k) \geq J^k_{max}$ then take

$$S^{k+1} = S^k;$$

set $k = k+1$ and go to Step 2

Step 4. If $J(\bar{\theta}^k) < J^k_{max}$ then take

$$S^{k+1} = S^k \cup \{\bar{\theta}^k\} - \{\theta^k_{max}\};$$

set $k = k+1$ and go to Step 1

The initial population of points is expected to cluster in the region where the global optimum is most likely to be. In practice, the possibility of locating a global minimum point rests on the fact that the number of points randomly chosen at the initial step is not small, and that global minimum points do not have narrow region of attraction. From this point of view, it appears clearly that the described algorithm is a heuristic. In this respect, we refer to section 7.2 of Torn and Zilinskas' book [25], where it is reported a thorough discussion on why heuristics are necessary and on how many the heuristics elements in global optimization are.

4.2. Local refinement

Numerical experience seems to indicate that the algorithm described in Section 4.1 is efficient enough at the global search, while it is not able to perform a sufficiently fast local minimization when the algorithm has produced an estimate $\tilde{\theta}$ "good enough". A local refinement procedure seems to be a better way to compute the global minimizer when a point $\tilde{\theta}$ near the global minimum is provided by the global method. In fact derivative-free local methods have better convergence properties than global ones and the risk to compute a local minimizer that is not global is unlikely by starting the local method from $\tilde{\theta}$.

In this paper, for the local refinement, we use a direct-search method that is a line-search version of the coordinate-search algorithm. Direct-search methods are derivative-free methods that sample the objective function at a finite number of points. At each iteration, they decide which actions to take solely based on those function

values, and without any explicit or implicit derivative approximation or model building [5]. Two particular subclasses of globally convergent direct-search methods are pattern-search methods and line-search methods. Pattern-search methods present the distinguishing feature of evaluating the objective function on specified geometric patterns. Line-search methods, on the contrary, draw their inspiration from the gradient-based minimization methods and perform one dimensional minimization along suitable directions. These two classes of methods present different interesting features. Pattern search methods can accurately sample the objective function in a neighborhood of a point. Hence, they can identify a "good" direction, namely, a direction along which the objective function decreases significantly. Line search algorithms can perform large steps along the search directions and, therefore, can exploit to a large extent the possible goodness of the directions. The algorithm used here combines these approaches in order to determine "good" directions and to perform "significant" step lengths along such directions [16].

4.2.1. Coordinate-Search Algorithm With Line-Search Expansions

Data: $\theta^0 \in \mathbb{R}^n$, $\tilde{\alpha}_1^0, \ldots, \tilde{\alpha}_n^0 > 0$, $\sigma \in (0,1)$, $\gamma \in (0,1)$, $\delta > 1$, $\varepsilon > 0$

Step 0. Set $k = 0$

Step 1. If $\max_{i=1,\ldots,n} \{\tilde{\alpha}_i^k\} \leq \varepsilon$ then STOP; set $i = 1$, $y_1^k = \theta^k$, $x^k = \theta^k$

Step 2. If $J(y_i^k + \tilde{\alpha}_i^k e_i) \leq J(y_i^k) - \gamma(\tilde{\alpha}_i^k)^2$ and $J(y_i^k + \tilde{\alpha}_i^k e_i) < J(x^k)$ then
set $\alpha_i^k = \tilde{\alpha}_i^k$ and $x^k = y_i^k + \alpha_i^k e_i$
while $J(y_i^k + \delta\alpha_i^k e_i) \leq J(y_i^k) - \gamma(\delta\alpha_i^k)^2$ and $J(y_i^k + \delta\alpha_i^k e_i) < J(x^k)$
set $\alpha_i^k = \delta\alpha_i^k$ and $x^k = y_i^k + \alpha_i^k e_i$
end while
set $\tilde{\alpha}_i^{k+1} = \alpha_i^k$
else set $\alpha_i^k = 0$ and $\tilde{\alpha}_i^{k+1} = \sigma\tilde{\alpha}_i^k$

Step 3. Set $y_{i+1}^k = y_i^k + \alpha_i^k e_i$

Step 4. If $i < 2n$ then set $i = i + 1$ and go to Step 2

Step 5. Set $\theta^{k+1} = x^k$, $k = k + 1$ and go to Step 1

5. Experimental evaluation

We carried out our experiments on a simulator of the vehicle Girona500 [21]. Girona-500 is a reconfigurable autonomous underwater vehicle designed for a maximum operating depth of up to 500 m. The vehicle has passive stability in pitch and roll, making it suitable for imaging surveys. The most remarkable characteristic of Girona 500 is its capacity to reconfigure for different tasks. On its standard configuration, the vehicle is equipped with typical navigation sensors (DVL, AHRS, pressure gauge and USBL) and a basic survey equipment (profiler sonar, side scan sonar, video camera and sound velocity sensor). In addition to these sensors, almost half the volume of the lower hull is reserved for mission-specific payload such as a stereo imaging system or an electric arm for manipulation tasks. The same philosophy has been applied to the propulsion system. The basic configuration has 4 thrusters, two vertical to actuate the heave and pitch and two horizontal for the yaw and surge. However, it is

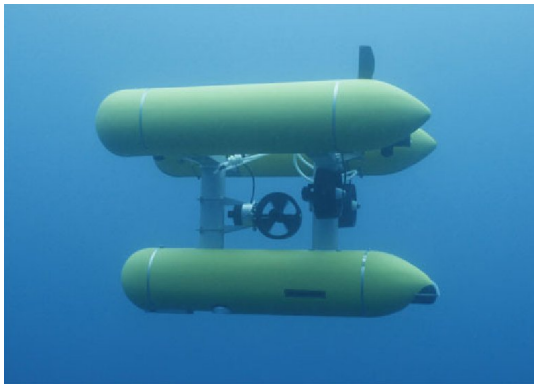

Fig. 1. The robot Girona500

possible to reconfigure the vehicle to operate with only 3 thrusters (one vertical and two horizontal) and with up to 8 thrusters to control all the degrees of freedom.

We performed our experiment using the Under Water Simulator (UWSim)[1] and the Robot Operating System (ROS)[2].

The robot's mission is a survey task, in which the agent has to reach a black box on the bottom of the seabed to take images and compose a mosaic. The target of the robot is to stay within the distance of 1m from the given point as long as possible, despite the disturbances in the environment. A current with velocity 0.6m/s has been simulated at depth higher than 2m, while the target point is 4.8m deep. Therefore, the agent cannot avoid the current and has to navigate trough it. The setting is represented in Figure 2. The current is too strong to be fully compensated

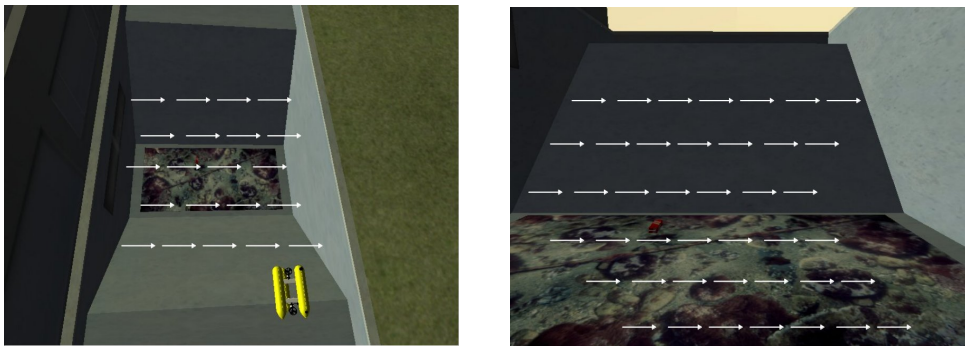

Fig. 2. The experimental setting. A strong current pushes the vehicle below 2m depth

by the thrusters, and once entered the robot slowly drifts away from the target point. We assume the agent is provided with an initial controller, able to perform the task

[1] http://www.irs.uji.es/uwsim/

[2] http://www.ros.org

under normal operation conditions. In our experiment this was a simple PD controller. The trajectory produced by the given controller is shown in Figure 3.

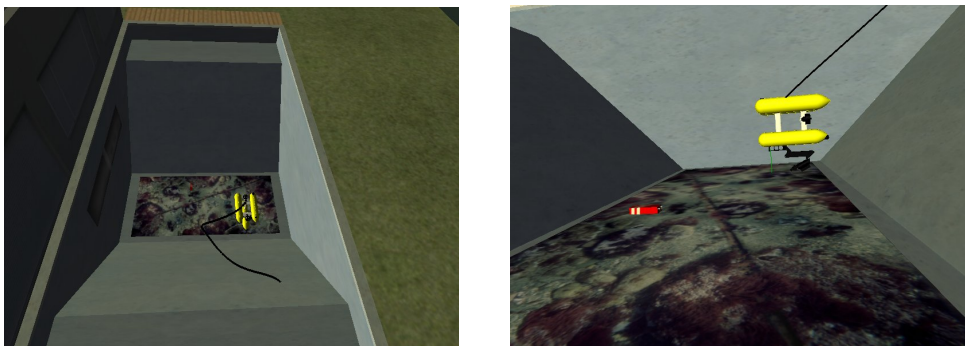

Fig. 3. The initial trajectory, produced by the PD controller alone, without learning.

Any robust controller can react to the environment but cannot counteract in advance unpredictable disturbances. Once the robot has entered the current there is little to do other then slow down the drifting. Therefore, the trajectory must be modified *before* entering the current, foreseeing its effect and counteracting them in advance. This is only possible by exploring the environment and learning about its characteristics. It requires to face the same task several times to make use of past experience to predict actions' effects. This is the focus of this paper: making a given controller adaptive and able to perform a task on which it would otherwise fail.

We assume an episodic scenario where the agent is able to return to the initial location (which is not inside the current) and start again. The battery power of Girona500 allows for 5 to 6 hours of operation, therefore we allocate for the on-line learning of behaviors no longer than 1 hour.

The reward function is given by:

$$J(\theta) = \begin{cases} -d & \text{if } d \geq D \\ t_D - D & \text{if } d < D \end{cases}$$

where D is a distance threshold, d is the minimum distance reached from the target, and t_D is the time spent at a distance from the target lower than D. In our experiments, $D = 1\text{m}$. Both d and t_D depend on θ through the policy, as expressed in Equation 4. Intuitively, the agent tries to maximize the time spent within 1m from the given point, using the distance as a heuristic when it is far from the target.

The policy computed and optimized is a correction to the given controller, whose output is velocity. Therefore, the actions are the continuous velocities summed to the controller's one. The policy has 18 parameters, and is a linear combination of position and velocity along the three axes for each component of the action.

5.1. Results

We first ran the global algorithm with an initial population of 100 samples. This phase is an off-line learning, and its aim is to provide a good starting point for the local search. Since, in this paper, we are going to run the second phase in simulation as well, we account for the discrepancy between the simulator and the actual environment by having a slightly different current in the two phases. The results are shown in Figure 4. After a few hundred trials, the algorithm finds a point above the threshold,

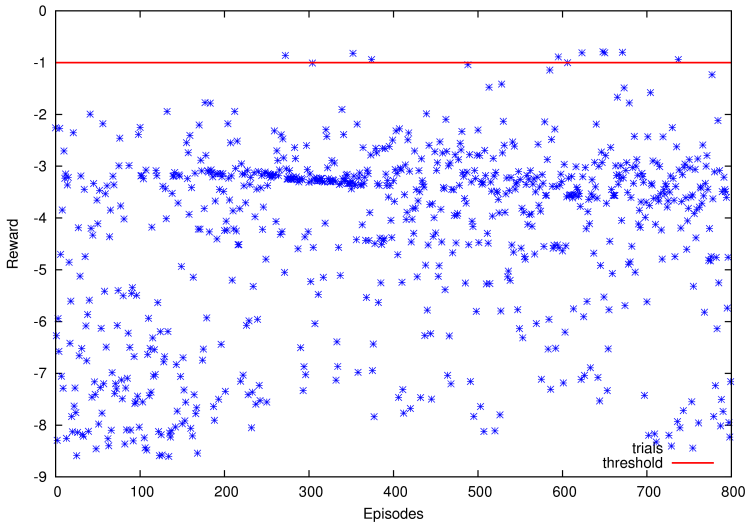

Fig. 4. The results of the trials during the initial random search

that is where the agent is able to approach the target point closer than 1m.

We took the best point found in this phase, and used it to initialize the policy for the local search. In this second phase, the current is the *real* one, which without learning produces the trajectory in Figure 3. We limited learning to 150 episodes, which take about one hour of real time. The policy learned during the first phase, when applied to the current of the second phase produces the trajectory shown in Figure 5. It manages to reach the point by staying higher than the current for as long as possible, and then entering it. Although learned under slightly different environment conditions, this behavior is already a good one. At this stage the agent is able to get within 1m from the target, therefore being able to perform the survey task. Local learning will attempt to maximize the time spent in this area, despite the real current.

The results of the local search are shown in Figure 6. The plot shows the learning curves from both the initial policy given from the first phase, and from the zero vector, which corresponds to no correction to the given controller. The local algorithm proved fast enough to reach the desired zone (-1 threshold in the reward) within the hour allocated for on-line learning. This shows that even with no prior information, this version of line search is well suited for Reinforcement Learning. Moreover, starting from the policy obtained through global learning, it performed better from

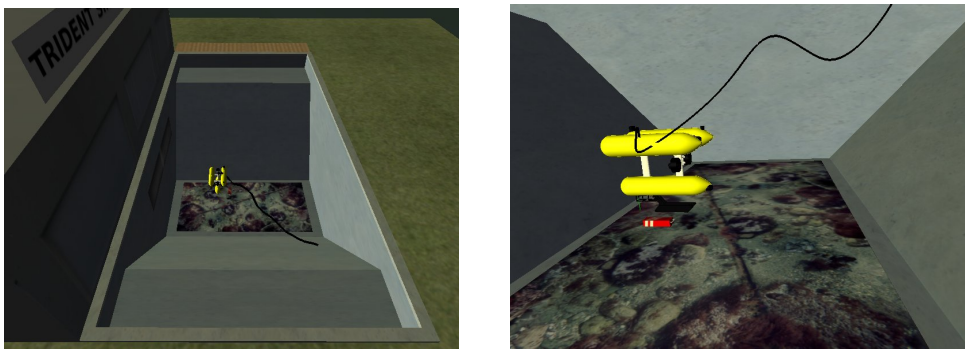

Fig. 5. The trajectory obtained by the first phase of learning, with the current of the real environment. The current in this picture is different from the one during learning.

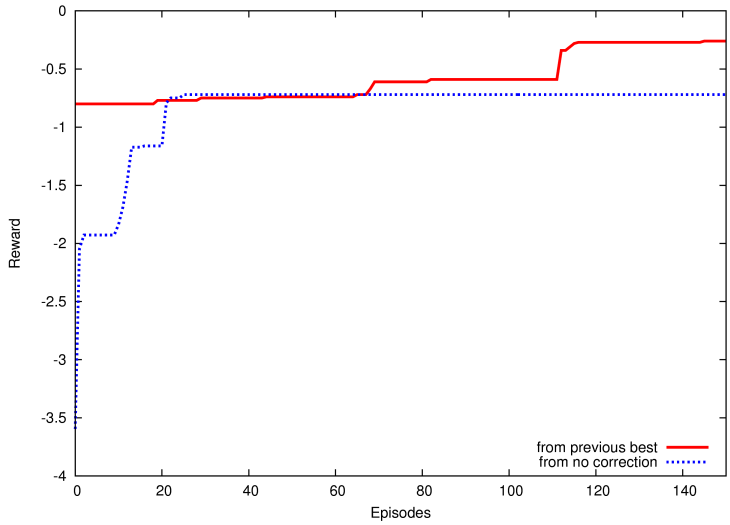

Fig. 6. Results of local learning from the initial point determined during the first phase and from the zero vector, which corresponds to no correction.

the first trials, reaching a even higher reward. The global random search, therefore, has been able to find a point with a better neighborhood than the natural initial point, that is when no correction is applied to the given controller.

6. Conclusion

The two algorithms we combined fit well with the paradigm of reinforcement learning in robotics. Simulators are fundamental tools, which are often used in place of the real robot for dangerous exploration. Off-line learning, however, cannot alone achieve adaptivity in unknown environments and go beyond what has been modeled

in the simulator. We showed how derivative-free line-search algorithms can be effective in reinforcement learning, and provide an alternative to gradient-based methods while retaining most of the features that made them so popular in robotics. With the two-phased learning strategy presented in this paper, we tailored an optimization algorithm for both off and on-line learning, obtaining the best from the global and the local method. Notably, this provides a very general methodology to wrap a learning framework around a given controller, increasing its robustness and ability to adapt the environments, especially under unmodeled circumstances.

7. Acknowledgments

This work was supported by the PANDORA European project under contract FP7-ICT-288273.

References

1. Barash, D .. A genetic search in policy space for solving Markov decision processes, – In: AAAI Spring Symposium on Search Techniques for Problem Solving under Uncertainty and Incomplete Information, 1999.
2. Baxter, J., P. Bartlett. Direct gradient-based reinforcement learning, – In: Proceedings of IEEE International Symposium on Circuits and Systems, Vol. 3, IEEE, 2000, 271–274.
3. Brachetti, P., M. De Felice Ciccoli, G. Di Pillo, S. Lucidi. A new version of the Price's algorithm for global optimization, Journal of Global Optimization, Vol. 10, No 2, 1997, 165–184.
4. Carpin, S., M. Lewis, J. Wang, S. Balakirsky, C. Scrapper. Bridging the gap between simulation and reality in urban search and rescue, Robocup 2006: Robot Soccer World Cup X, 1–12.
5. Conn, A., K. Scheinberg, L. Vicente. Introduction to derivative-free optimization, Vol. 8, Society for Industrial Mathematics, 2009.
6. Dorigo, M., M. Birattari, T. Stutzle. Ant colony optimization, IEEE Computational Intelligence Magazine, Vol. 1, No 4, 2006, 28–39.
7. Glover, F., M. Laguna. Tabu search, Vol. 1, Springer, 1998.
8. Goldberg, D .. Genetic algorithms in search, optimization, and machine learning, Addison-wesley, 1989.
9. Gomez, F., J. Schmidhuber, R. Miikkulainen. Efficient Non-Linear Control through Neuroevolution, – In: Proceedings of the European Conference on Machine Learning, Springer, Berlin, 2006, 654–662.
10. Horst, R., P. Pardalos, N. Thoai. Introduction to global optimization, Springer, 2000.
11. Jakobi, N., P. Husbands, I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics, Advances in artificial life, 704–720.
12. Kakade, S .. A natural policy gradient, Advances in neural information processing systems, Vol. 14, 2001, 1531–1538.
13. Kirkpatrick, S., C. Gelatt Jr, M. Vecchi. Optimization by simulated annealing, Science, Vol. 220, No 4598, 1983, 671–680.
14. Kober, J., J. Peters. Policy search for motor primitives in robotics, Machine learning, Vol. 84, No 1, 2011, 171–203.
15. Kormushev, P., D. G. Caldwell. Simultaneous Discovery of Multiple Alternative Optimal Policies by Reinforcement Learning, – In: IEEE International Conference on Intel-

ligent Systems (IS 2012), 2012.
16. Lucidi, S., M. Sciandrone. On the global convergence of derivative-free methods for unconstrained optimization, SIAM Journal of Optimization, Vol. 13, No 1, 2002, 97–116.
17. Peters, J., S. Schaal. Policy gradient methods for robotics, – In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2006, 2219–2225.
18. Peters, J., S. Schaal. Reinforcement learning by reward-weighted regression for operational space control, – In: Proceedings of the 24th international conference on Machine learning, ACM, 2007, 745–750.
19. Peters, J., S. Vijayakumar, S. Schaal. Natural Actor-Critic, – In: Proceedings of the 16th European Conference on Machine Learning (ECML), 2005, 280–291.
20. Price, W.. Global optimization by controlled random search, Journal of Optimization Theory and Applications, Vol. 40, No 3, 1983, 333–348.
21. Ribas, D., N. Palomeras, P. Ridao, M. Carreras, A. Mallios. Girona 500 AUV, from survey to intervention, IEEE/ASME Transactions on Mechatronics, Vol. 17, No 1, 2012, 46–53.
22. Sutton, R., D. McAllester, S. Singh, Y. Mansour. Policy gradient methods for reinforcement learning with function approximation, Advances in neural information processing systems, Vol. 12, No 22.
23. Theodorou, E., J. Buchli, S. Schaal. A generalized path integral control approach to reinforcement learning, The Journal of Machine Learning Research, Vol. 9999, 2010, 3137–3181.
24. Torczon, V., et al.. On the convergence of pattern search algorithms, SIAM Journal on optimization, Vol. 7, No 1, 1997, 1–25.
25. Torn, A., A. Zilinskas. Global Optimization, Springer, 1989.
26. Williams, R.. Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine learning, Vol. 8, No 3, 1992, 229–256.