Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Cross Entropy Optimization of Action Modification Policies for Continuous-Valued MDPs ⋆

Kamelia Mirkamali * **Lucian Buşoniu** **

* *Department of Computer Engineering, Khaje Nasir Toosi University of Technology, Tehran, Iran (e-mail: k.mirkamali@kntu.ac.ir).*
** *Automation Department, Technical University of Cluj-Napoca, Cluj-Napoca, Romania (e-mail: lucian@busoniu.net)*

Abstract: We propose an algorithm to search for parametrized policies in continuous state and action Markov Decision Processes (MDPs). The policies are represented via a number of basis functions, and the main novelty is that each basis function corresponds to a small, discrete modification of the continuous action. In each state, the policy chooses a discrete action modification associated with a basis function having the maximum value at the current state. Empirical returns from a representative set of initial states are estimated in simulations to evaluate the policies. Instead of using slow gradient-based algorithms, we apply cross entropy method for updating the parameters. The proposed algorithm is applied to a double integrator and an inverted pendulum problem, with encouraging results.

Keywords: Markov decision processes, policy search, cross-entropy optimization, continuous actions.

1. INTRODUCTION

In recent years, reinforcement learning (RL) has been increasingly studied in machine learning and control theory. RL is the problem faced by an agent that should learn optimal sequential behavior through trial and error interaction with a dynamic environment (Kaelbling et al., 1996; Van Hasselt et al., 2016; Sutton and Barto, 2018). RL problems are usually formalized as Markov decision processes (MDPs). Traditional RL approaches approximate the value functions (long-term expected return) by functional approximators with given architectures and a manageable number of adjustable parameters (Sutton and Barto, 2018). Instead of basing the policy updates on the approximate value function, policy approximation methods search in the policy space directly (Beitelspacher et al., 2006). In both value and policy approximation, there are both local and global optimization methods for adjusting the parameters. In value approximation dynamic programming provides a global method; while e.g. De Boer et al. (2005) introduce an algorithm that approximates value functions by the gradient-based method. Actor-critic algorithms, viewed as a hybrid of value function approximation and policy search, have been shown to be more effective than either of these options (Xu et al., 2014). The actor-critic update strategy relies on local gradient-based methods and requires an approximate value function (Melo and Lopes, 2008; Lillicrap et al., 2015). Although many policy search algorithms focus on gradient-based optimization methods, the key disadvantage associated with these methods is they risk getting stuck in local optima. Global optimization methods such as cross entropy (CE) optimization overcome this shortcoming, see e.g. Mannor et al. (2003).

Considering now the problem of solving MDPs with continuous actions, most existing approaches rely on various forms of continuous-action value function approximators, see e.g. Strösslin and Gerstner (2003); Sallans and Hinton (2004); Gaskett et al. (1999).

In particular, the Binary Action Search (BAS) method presented by Pazis and Lagoudakis (2009) searches the entire continuous action range for increment and decrement modifications to the values of the action variables, according to an internal binary policy defined over an augmented state space. This approach is coupled with Least-Squares Policy Iteration, which uses a weighted sum of basis functions (BFs) to approximate the value function.

In the different class of direct policy search methods, Busoniu et al. (2010) introduced an algorithm that applies global cross entropy optimization on the parameters of policy. It looks for the best policy represented by a number of tunable BFs, where discrete actions are assigned to the BFs. This algorithm is designed for MDPs with a discrete or discretized action space.

Here, we propose an algorithm that directly searches in the policy space with global cross entropy optimization, like the method of Busoniu et al. (2010), but differently from that method, it searches for the best discrete action modification (DAM) to the current continuous action. This change is the key contribution of the paper. It is inspired by the BAS method of Pazis and Lagoudakis (2009), and

⋆ This work was supported by a grant of the Romanian Ministry of Research and Innovation, CNCS - UEFISCDI, project number PN-III-P1-1.1-TE-2016-0670, within PNCDI III.

similar ideas have been used in model-predictive control (Mayne et al., 2000). The magnitude of the action modifications is a small fraction of the magnitude of the original actions. A prespecified number M of possible DAMs are chosen. Then, a number of parametrized BFs are defined, with tunable centers and shapes. A mapping between the BFs and the possible DAMs is also defined. For any state, the policy chooses the DAM associated to the BF that takes the largest value in that state. CE optimization is applied to search for both the best parameters of the BFs, and for their mapping to DAMs. In this way, we mix the advantages of continuous action policies in (Pazis and Lagoudakis, 2009) with those of global cross entropy optimization in (Busoniu et al., 2010). The resulting algorithm for CE optimization of parametrized policies is evaluated in a double integrator and an inverted pendulum problem.

The rest of the paper is organized as follows. Section 2 provides the necessary background. Section 3 introduces the CE policy optimization method for continuous actions. Section 4 reports our numerical study. Finally, Section 5 concludes and provides several directions for future research.

2. TECHNICAL BACKGROUND

2.1 Markov decision processes

In this section Markov decision processes are briefly described, focusing on the deterministic variant. For more details see Puterman (2014). A deterministic continuous-state and action MDP is defined by a 5-tuple $\{X, U, f, r, \lambda\}$, where X is the continuous state space of the system, U is the continuous action space, $f : X \times U \to X$ is the transition function ($x_{k+1} = f(x_k, u_k)$ denotes the transition to x_{k+1} when taking action u_k in state x_k), $r : X \times U \to R$ is a reward function ($r(x_k, u_k)$ is the reward for taking action u_k in state x_k) and $\lambda \in (0, 1]$ is the discount factor for future rewards. At each discrete time step k, in state x_k, the controller takes an action u_k according to a control policy π. A deterministic policy π for an MDP is a mapping from states to actions $\pi : X \to U$; $\pi(x)$ denotes the action chosen by policy π in state x.

The infinite-horizon discounted return for an initial state x_0 under a policy π is:

$$R^{\pi}(x_0) = \sum_{k=0}^{\infty} \lambda^k r(x_k, \pi(x_k)) \tag{1}$$

where $x_{k+1} = f(x_k, \pi(x_k))$.

The action-value, or Q-value $Q^{\pi}(x, u)$ of a state action pair (x, u) under a policy π is defined as the discounted return when the process begins in state x, action u is taken and after that all decisions are made according to policy π:

$$Q^{\pi}(x, u) = r(x, u) + \lambda R^{\pi}(f(x, u)) \tag{2}$$

The optimal Q-function $Q^*(x, u) = \max_{\pi} Q^{\pi}(x, u)$ is the maximal Q-function across all policies. The goal of the decision maker is to find an optimal policy π^* that maximizes the expected reward. If the optimal Q-function is known, the optimal policy can be found as follows:

$$\pi^*(x) = \arg\max_{u} Q^*(x, u) \tag{3}$$

2.2 The cross-entropy method

The cross entropy (CE) is an information theoretic measure that quantifies the difference between two probability distributions (Zhu, 2002). The CE *method* was originally developed for rare event simulation (Rubinstein, 1997). It was later extended to optimization in (Rubinstein, 1999); it can be used to find near-optimal solutions of combinatorial and continuous multi-extremal optimization problems. The main idea behind the CE method is to associate each optimization problem with a rare event estimation problem, and solve the latter by an adaptive CE procedure. In this manner a random sequence of solutions is generated, which in practice often converges to the optimal or near-optimal solution.

The CE method employs the following two phases iteratively (De Boer et al., 2005):

(1) Generate a sample of random solutions according to a parametrized probability distribution.
(2) Update the parameters of the distribution based on the best samples generated in the previous step, so as to produce a better sample in the next generation.

A simple form of CE optimization is shown in Algorithm 1, adapted from (De Boer et al., 2005). We aim to maximize a function $S(Y)$ across the values of $Y \in \mathcal{Y}$. We denote the maximum as follows:

$$\gamma^* = \max_{Y \in \mathcal{Y}} S(Y) \tag{4}$$

We start by converting the deterministic problem into stochastic one. Define a family of distributions $F(.; v)$ parametrized by v, with support equal to the set $\mathcal{Y}$. Problem (4) corresponds to an *associated stochastic problem* of estimating the probability that the function S will be larger than some level γ when applied to random samples from such a distribution $F(\cdot, v)$:

$$\ell(\gamma) = P_v(S(Y) \geq \gamma) = E_v I\{S(Y) \geq \gamma\} \tag{5}$$

where $I\{\cdot\}$ is the indicator function, equal to 1 when the argument is true. The event $S(Y) \geq \gamma$ is rare, so estimating ℓ is nontrivial.

The CE method solves the optimization problem by generating a sequence of pairs (γ_t, v_t), which should converge to a small neighborhood of an optimal pair (γ^*, v^*). Given the previous iterate v_{t-1} of the distribution parameter, a level γ_t is set by drawing random samples $Y_1, Y_2, ..., Y_n$ from $F(.; v_{t-1})$, sorting these samples by the ascending value of the objective function, and evaluating the $(1 - \rho)$-quantile of the sample performances:

$$\hat{\gamma_t} = S(Y_{\lceil (1-\rho)n \rceil}) \tag{6}$$

where $\rho \in (0, 1)$ is a chosen fraction of elite samples. Note that:

$$P_{v_{t-1}}(S(Y) \geq \gamma_t) \geq \rho \tag{7}$$

The next iterate v_t is derived as the solution of the program:

$$\max_{v} E_{v-1} I\{S(Y) \geq \gamma_t\} \ln F(Y; v) \tag{8}$$

which provides a good importance sampling distribution, l.e. one that increases the probability of the interesting event $S(Y) \geq \gamma_t$. The empirically sampled version of (8) is:

$$\max_v \frac{1}{n}\sum_{l=1}^{n} I\{S(Y_l) \geq \hat{\gamma}_t\} \ln F(Y_l; v) \quad (9)$$

Note that for many types of distributions used in practice, including those from the natural exponential family, (9) has a closed-form solution. We will use in our methods Gaussian densities parametrized by their mean and standard deviation, for which the solution is simply the mean and standard deviation of the elite samples; and Bernoulli distributions parametrized by their mean, for which the solution is the mean of the elite samples.

Instead of setting the next parameter directly to the solution of (9), the following smoothed version is often used:

$$v_t = \alpha \hat{v}_t + (1-\alpha) v_{t-1}, \quad (10)$$

where $\hat{v}_t$ is the solution of (9) and α is the smoothing parameter, with $0.7 < \alpha \leq 1$. It is clear that for $\alpha = 1$ the parameter is fully replaced. Smoothing reduces the chance that v degenerates in a way that assigns probability mass 0 to some parts of the solution space, thereby eliminating those parts from the search and possibly leading to suboptimal solutions.

Algorithm 1 Cross-Entropy Optimization

1: Choose some initial v_0
2: **repeat** at each iteration $t = 1, 2, \ldots$
3: Generate samples $Y_1, Y_2, ..., Y_n$ from $F(.; v_{t-1})$
4: Compute the $(1-p)$-quantile $\hat{\gamma}_t$ with (6)
5: Use the samples to solve the stochastic program (8)
6: Apply (10) to obtain v_t
7: **until** a stopping criterion is met

3. CROSS-ENTROPY SEARCH OF DISCRETE ACTION MODIFICATION POLICIES

This section describes the proposed algorithm, CE optimization of parametrized policies for finding continuous actions. First, the policy parametrization is discussed. Then, it is explained how CE optimization is applied to search for a good parametrized policy.

A notable class of representations in the policy search literature is based on Radial Basis Functions (RBFs). Here, we define N Gaussian RBFs over the state space, with parametrized centers and radii:

$$\phi_l(x; p) = \exp\left[-\sum_{d=1}^{D} \frac{(x_d - c_{l,d})^2}{b_{l,d}^2}\right], l = 1, ..., N \quad (11)$$

Here, D is the dimension of RBFs, equal to the number of state variables x, $c_{l,d}$ is d-th dimension center of l-th function, $b_{l,d}$ is d-th dimension radius of the l-th RBF, and x_d is value of state variable in d-th dimension. All the RBF centers and radii are collected in an RBF parameter vector p. The centers of the RBFs should stay inside the state space X. The radii must be strictly positive.

Our algorithm works for MDPs with continuous actions in which the successive near-optimal actions only have a local change across time steps. For instance, this holds when the dynamics and optimal solutions are continuous over the states (it does not, however, hold in all cases: consider for instance time-optimal bang-bang solutions,

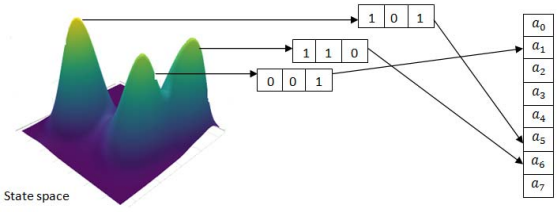

Fig. 1. Illustration of our DAM policy representation.

which are known to be discontinuous over time). Using this property, the policy can *modify* the current action by a small amount, instead of searching for an absolute action magnitude. We choose M discrete such possible modifications, where the best M is obtained from experiment. Let δ denote a DAM.

The algorithm looks for the best policy that can be represented using N RBFs that are associated with the M DAMs by a mapping (function) $j : \{1, \ldots, N\} \rightarrow \{1, \ldots, M\}$, as illustrated in Figure 1. For given RBFs and a given mapping, the policy chooses the DAM associated to RBF that has maximum value in x:

$$\pi_\delta(x) = \delta_{j(l^*)}, l^* = \max_l \phi_l(x; p) \quad (12)$$

Here, l is the RBF index, l^* is the RBF that has maximum value in x, $j(l^*)$ is the index of the DAM associated to this RBF by the current RBF-DAM mapping j, and $\delta_{j(l^*)}$ is the actual DAM. In the example of Figure 1, the number of RBFs is $N = 3$ and the number of DAMs is $M = 8$. Note that we use a binary representation of the mapping (indices) j, which requires $\lceil \log M \rceil = 3$ bits per RBF; these binary representations are also illustrated in the figure.

It is important to understand that applying DAMs with a policy that is only a function of the state x makes the problem non-Markov (since the next state is not directly predictable anymore). Nevertheless, policy search methods are known to exhibit some resilience to non-Markov problems. Adding the previous value of the action to the state signal is a version that we will explore in future work.

Our policy updating strategy searches for the optimal RBF parameters p and mapping j that maximize the following score function:

$$S(Y) = \sum_{x_0 \in X_0} R^Y(x_0) \quad (13)$$

where Y is the concatenation of p and the binary representation of j, and X_0 is a finite set of representative initial states. Moreover, R^Y is estimated return of one such state under the DAM policy π_δ parametrized by Y:

$$R^Y(x_0) = \sum_{k=0}^{K} \lambda^k r(x_k, u_k) \quad (14)$$

Crucially, $u_k = u_{k-1} + \pi_\delta(x_k)$, and u_{-1} may be initialized e.g. to 0. Here, K is the length of the episodes used to estimate the return.

For CE optimization the deterministic problem of maximizing (13) must be converted to a stochastic one. To

make the problem stochastic, Gaussian densities F_p are selected for the centers and radii p of all the RBFs, and Bernoulli mass functions F_j are selected for the binary representations of the DAM assignments j to RBFs. The Gaussian density for each center $c_{l,d}$ is parametrized by the mean $m^c_{l,d}$ and the standard deviation $s^c_{l,d}$; and similarly, the density for each radius $b_{l,d}$ is parametrized by the mean $m^b_{l,d}$ and the standard deviation $s^b_{l,d}$. Each bit with index $J = 1, \ldots, N \cdot \lceil \log M \rceil$ is extracted from its Bernoulli distribution parametrized by its mean β_J. Thus, the overall distribution parameter v contains all the means and standard deviations for all the RBF centers and radii ($4DN$ parameters); and the means of all the bits ($N \cdot \lceil \log M \rceil$ parameters). The overall parameter v is initialized in the beginning of the algorithm and updated using the elite samples of Y (p and j).

The initialization of means and standard deviations for centers and radii of all the RBFs is done as follows (for all BFs l in first CE iteration):

$$m^c_l = \frac{x_{\max} + x_{\min}}{2},\ s^c_l = \frac{x_{\max} - x_{\min}}{2},\\ m^b_l = s^b_l = \frac{x_{\max} - x_{\min}}{N} \tag{15}$$

where $x_{\max}$ and $x_{\min}$ are the bounds of the state space, and vector assignments are interpreted elementwise. The Bernoulli distributions are initialized uniform.

The proposed method is summarized in Algorithm 2. The inputs into the algorithm are the transition function f and reward function r; the discount factor λ; the percentile ρ of elite samples to use in the updates; the representative states X_0; the number of RBFs N; and the number of DAMs M.

Algorithm 2 CE search for DAM policies

```
Initialize density parameters v_0
for t = 1, ..., t_max do
    for l = 1, ..., n do
        Generate p_l from density F_p(·, v_{t-1})
        Generate j_l from distribution F_j(·, v_{t-1})
        Initialize sample score, s_l = 0
        for each x_0 ∈ X_0 do
            k = 0, u_{-1} = 0
            repeat
                δ_k = π_δ(x_k)
                u_k = u_{k-1} + δ_k
                s_l = s_l + λ^k r(x_x, u_k)
                x_{k+1} = f(x_k, u_k)
                k = k + 1
            until k > K
        end for
    end for
    Reorder such that s_1 < s_2 < ... < s_n
    Find parameters v̂_t from elite samples p_l and j_l,
     l = 1, 2, ..., ⌈(1 − ρ)n⌉
    v_t = α v̂_t + (1 − α) v_{t-1}
end for
```

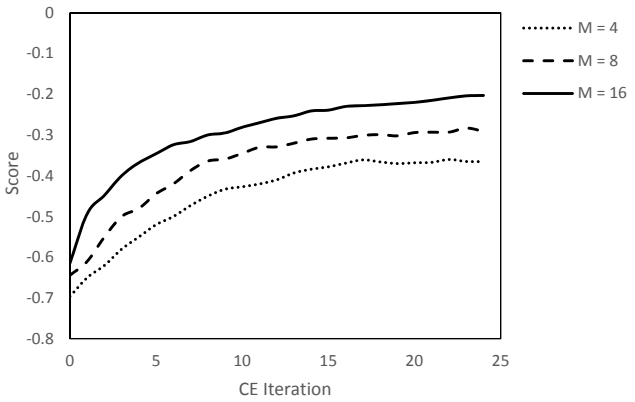

Fig. 2. Double integrator: Performance with varying number of DAMs

4. NUMERICAL STUDIES

We apply the algorithm proposed for two popular control problems: a double integrator (selected because it provides a simple linear baseline) and an inverted pendulum (selected to provide a more involved, nonlinear example).

4.1 Double integrator

The double integrator can model the control of an object along a line, such as a car moving on a road. The problem has two-dimensional continuous states $x = (p, v)$, consisting of the position p and the velocity v, and a one-dimensional continuous action $u = a$, the acceleration. [1] The continuous-time dynamics are:

$$\dot{p} = v, \quad \dot{v} = a \tag{16}$$

which we discretize in time with a sampling period of 0.1 s. States are bounded so that $|p| \leq 1$ and $|v| \leq 1$.

The objective is optimal control of the car acceleration to bring it to the zero position. This objective is expressed by a reward function that penalizes deviations from the goal position, as well as acceleration magnitude:

$$r(x, u) = p^2 + a^2 \tag{17}$$

To apply our algorithm, the following set X_0 of initial states (p_0, v_0) is considered:

$$\{(-0.8, 0), (-0.7, 0), ..., (0.8, 0)\}$$

All these states are zero-velocity, i.e. the car starts moving from the respective positions. The discount factor is $\gamma = 0.95$, and the parameter settings for the algorithm are: $\rho = 0.1$, $\alpha = 0.95$, $K = 500$, $n = 100$, $t_{\max} = 50$. These values were chosen with minimal or no tuning.

Since the car acceleration is restricted in the range $[-1, 1]$, any fraction of this interval can be selected to cover with DAMs; here, we select $[-0.1, 0.1]$, a small 10% fraction. We discretize this interval into M equidistant DAMs which are assigned to the RBFs.

Figure 2 shows the scores (13) during $t_{\max} = 25$ CE iterations, with $n = 100$ samples at each CE iteration,

[1] For a more natural notation in the examples, we allow the reuse of some symbols from the algorithmic development.

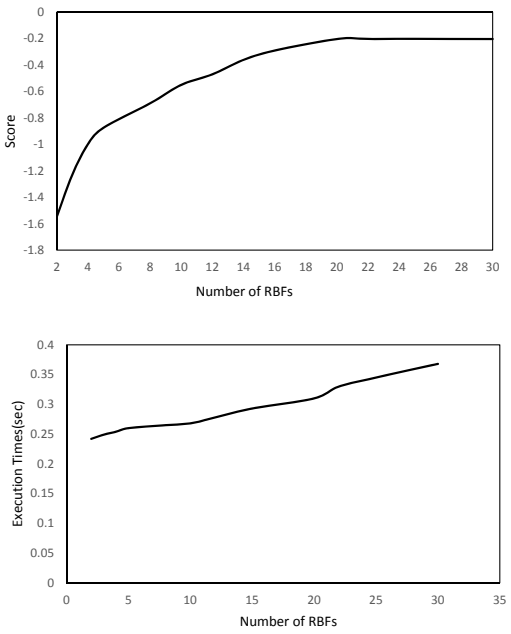

Fig. 3. Double integrator: Performance (top) and execution time (bottom) with varying number of RBFs

for $N = 20$ RBFs and $M = 8, 16, 32$ DAMs. Each curve is the mean of 25 runs of the experiment. The algorithm generally obtains better performance with a larger M, of course at larger computational cost per iteration. Since the problem is simple, after around 15 iterations the performance does not increase much.

Note that the number N of RBFs is taken 20 since larger values bring no benefit: performance plateaus while computational cost keeps increasing, see Figure 3.

4.2 Inverted pendulum

Consider the problem of balancing a pendulum in the upright position, by applying forces to a cart to which the pendulum is attached (Zheng et al., 2006). The problem has a two-dimensional continuous state space including the vertical angle θ and the angular velocity $\dot{\theta}$ of the pendulum; and a one-dimensional continuous action space – the force F. The dynamics are:

$$\ddot{\theta} = \frac{(M+m)(q\sin\theta - \dot{\theta}) - (lm\dot{\theta}^2\sin\theta + F)\cos\theta}{l(M + m(1-\cos^2\theta))} \quad (18)$$

where $g = 9.8$ m/s^2 is the gravitational acceleration, $m = 2$ kg is the mass of the pendulum, $M = 8$ kg is the mass of the cart, and $l = 0.5$ m is the length of pendulum (Pazis and Lagoudakis, 2009). The dynamics are discretized with a sampling period of 0.1 s.

The goal is to bring the pendulum to the upright position, $\theta = 0$, with zero angular velocity. This goal is expressed by the reward function:

$$r = \begin{cases} -\left[\left(\frac{2\theta}{\pi}\right)^2 + \dot{\theta}^2 + \left(\frac{F}{50}\right)^2)\right] & \text{if } |\theta| \le \frac{\pi}{2} \\ -1000 & \text{otherwise} \end{cases} \quad (19)$$

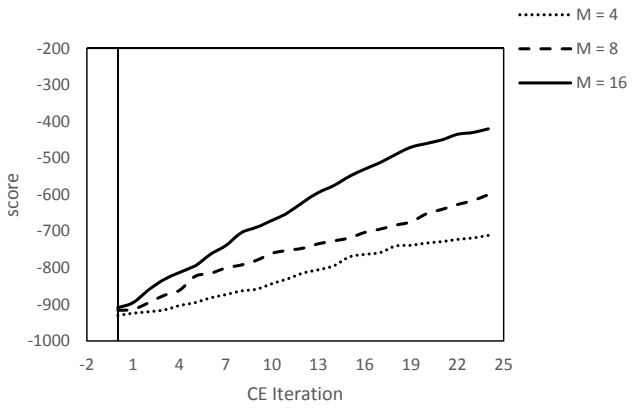

Fig. 4. Inverted pendulum: Performance with varying number of DAMs

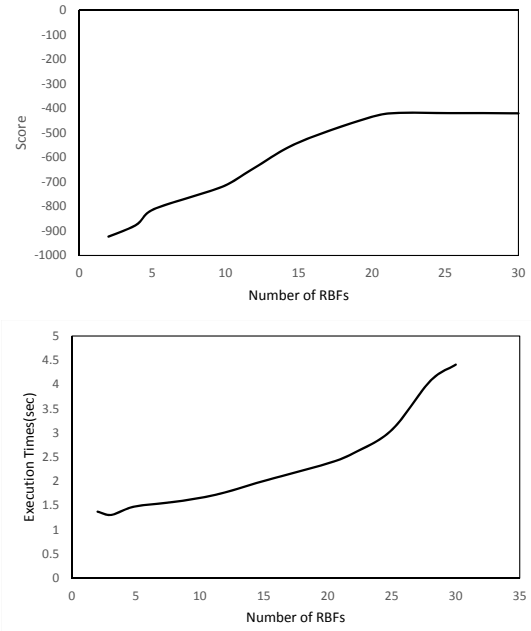

Fig. 5. Inverted pendulum: Performance (top) and execution time (bottom) with varying number of RBFs

To apply CE optimization, a set X_0 of representative initial states is taken for which all angular velocities are zero, and the angles vary in:

$$\{\pm(\frac{\pi}{2} - \epsilon), \pm\frac{2\pi}{5}, \pm\frac{3\pi}{10}, \pm\frac{\pi}{5}, \pm\frac{\pi}{10}\}$$

In all initial states the angular velocity is zero. Since $\theta = \frac{\pi}{2}$ and $\theta = -\frac{\pi}{2}$ are right at the edge of the allowed state space, they are shifted inside with a small value $\epsilon = 0.005$.

The parameters of the algorithm are the same as for the continuous double integrator, $\gamma = 0.95$, $\rho = 0.1$, $n = 100$ and $K = 500$. We run the algorithm for $N = 20$ and $M = 8, 16, 32$. Figure 4 shows that scores increase with M, and as before, Figure 5 shows that $N = 20$ is a good choice.

5. CONCLUSION AND FUTURE WORK

This paper proposed an algorithm for direct policy search in continuous state and action MDPs. The policy applies increments to the current action, where the increment chosen is the one associated to the largest basis function at the current state. Cross-entropy optimization is used to find the location and shape of the basis functions, as well as the mapping between these and the action increments. Encouraging simulation results for a linear and nonlinear problem were reported.

Besides augmenting the state by the current action in order to recover the Markov property, another idea that might improve the algorithm is to search for a simple linear combination of continuous-valued action modifications. CE optimization is able to search for such a parametrization. It will also be important to compare the algorithm to alternative methods for continuous-valued MDPs.

REFERENCES

Beitelspacher, J., Fager, J., Henriques, G., and McGovern, A. (2006). Policy gradient vs. value function approximation: A reinforcement learning shootout. *School of Computer Science, University of Oklahoma, Tech. Rep.*

Busoniu, L., Ernst, D., De Schutter, B., and Babuska, R. (2010). Cross-entropy optimization of control policies with adaptive basis functions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(1), 196–209.

De Boer, P.T., Kroese, D.P., Mannor, S., and Rubinstein, R.Y. (2005). A tutorial on the cross-entropy method. *Annals of operations research*, 134(1), 19–67.

Gaskett, C., Wettergreen, D., and Zelinsky, A. (1999). Q-learning in continuous state and action spaces. In *Australasian Joint Conference on Artificial Intelligence*, 417–428. Springer.

Kaelbling, L.P., Littman, M.L., and Moore, A.W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4, 237–285.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Mannor, S., Rubinstein, R.Y., and Gat, Y. (2003). The cross entropy method for fast policy search. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 512–519.

Mayne, D.Q., Rawlings, J.B., Rao, C.V., and Scokaert, P.O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6), 789–814.

Melo, F.S. and Lopes, M. (2008). Fitted natural actor-critic: A new algorithm for continuous state-action mdps. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 66–81. Springer.

Pazis, J. and Lagoudakis, M.G. (2009). Binary action search for learning continuous-action control policies. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 793–800. ACM.

Puterman, M.L. (2014). *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Rubinstein, R. (1999). The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2), 127–190.

Rubinstein, R.Y. (1997). Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1), 89–112.

Sallans, B. and Hinton, G.E. (2004). Reinforcement learning with factored states and actions. *Journal of Machine Learning Research*, 5(Aug), 1063–1088.

Strösslin, T. and Gerstner, W. (2003). Reinforcement learning in continuous state and action space. Technical report.

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement learning: An introduction*. MIT press.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double Q-learning. In *Thirtieth AAAI conference on Artificial Intelligence*.

Xu, X., Zuo, L., and Huang, Z. (2014). Reinforcement learning algorithms with function approximation: Recent advances and applications. *Information Sciences*, 261, 1–31.

Zheng, Y., Luo, S., and Lv, Z. (2006). Control double inverted pendulum by reinforcement learning with double CMAC network. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 4, 639–642. IEEE.

Zhu, Y.M. (2002). Volume image registration by cross-entropy optimization. *IEEE Transactions on Medical Imaging*, 21(2), 174–180.