Delft University of Technology

Delft Center for Systems and Control

Technical report 07-035

Fuzzy partition optimization for approximate fuzzy Q-iteration*

L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška

If you want to cite this report, please use the following reference instead:
L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška, "Fuzzy partition optimization for approximate fuzzy Q-iteration," *Proceedings of the 17th IFAC World Congress*, Seoul, Korea, pp. 5629–5634, July 2008.

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.51.19 (secretary)
fax: +31-15-278.66.79
URL: `http://www.dcsc.tudelft.nl`

*This report can also be downloaded via `http://pub.deschutter.info/abs/07_035.html`

Fuzzy Partition Optimization for Approximate Fuzzy Q-iteration ⋆

Lucian Buşoniu * **Damien Ernst** ** **Bart De Schutter** *
Robert Babuška *

* *Delft University of Technology, The Netherlands*
(e-mail: i.l.busoniu@tudelft.nl, b@deschutter.info, r.babuska@tudelft.nl)
** *Research Associate FNRS, University of Liège, Belgium*
(e-mail: ernst@montefiore.ulg.ac.be)

Abstract: Reinforcement Learning (RL) is a widely used learning paradigm for adaptive agents. Because exact RL can only be applied to very simple problems, approximate algorithms are usually necessary in practice. Many algorithms for approximate RL rely on basis-function representations of the value function (or of the Q-function). Designing a good set of basis functions without any prior knowledge of the value function (or of the Q-function) can be a difficult task. In this paper, we propose instead a technique to optimize the shape of a constant number of basis functions for the approximate, fuzzy Q-iteration algorithm. In contrast to other approaches to adapt basis functions for RL, our optimization criterion measures the actual performance of the computed policies in the task, using simulation from a representative set of initial states. A complete algorithm, using cross-entropy optimization of triangular fuzzy membership functions, is given and applied to the car-on-the-hill example.

Keywords: reinforcement learning, approximate Q-value iteration, fuzzy approximation, adaptive basis functions, cross-entropy optimization.

1. INTRODUCTION

Learning agents can tackle problems where pre-programmed solutions are difficult or impossible to design. Reinforcement learning (RL) is a popular learning paradigm for adaptive agents, thanks to the mildness of its assumptions on the environment (which can be a nonlinear, stochastic process), and to its ability to work without an explicit model of the environment (Sutton and Barto, 1998; Bertsekas, 2001). In RL , the agent receives feedback about its immediate performance in the form of a scalar reward signal, and its goal is to maximize the cumulative reward (the return) over the course of interaction with the environment. Well-understood algorithms with good convergence and consistency properties are available for solving RL problems (Bertsekas, 2001; Sutton and Barto, 1998). Most of these algorithms store a function that contains estimates of the return for every environment state (a value function) or state-action pair (a Q-function). Such algorithms are therefore not applicable to problems with large or continuous state spaces.

For these problems, approximate RL algorithms have to be used. Not all approximate algorithms guarantee convergence to a (suboptimal) solution; those that do usually employ basis-function approximations of the value or Q-function (Gordon, 1995; Ormoneit and Sen, 2002; Szepesvári and Smart, 2004; Ernst et al., 2005; Buşoniu et al., 2007). Many of these algorithms require the basis functions to be designed beforehand. In general, prior knowledge about the shape of the value function is needed in this design process. When prior knowledge is insufficient, many basis functions have to be defined to provide a good coverage and resolution over the entire state space, even in areas that might eventually be irrelevant to the optimal policy.

In this paper, we propose to optimize the parameters of a constant number of basis functions for the model-based fuzzy Q-iteration algorithm (Buşoniu et al., 2007). To evaluate each particular set of basis functions (membership functions, MFs), a policy is computed by running fuzzy Q-iteration to convergence. This is in contrast to most current techniques, which adapt the basis functions simultaneously with learning an RL solution. While our strategy incurs higher computational costs, it also ensures that a meaningful RL solution is available at any stage of the algorithm. The optimization criterion is the average return from a representative set of initial states, computed empirically with Monte Carlo simulations. Other criteria used in the literature typically estimate the accuracy in representing the value function. Our criterion has the advantage of being directly related to the agent's performance.

The proposed approach is useful when prior knowledge is not available and when the number of MF s is limited (e.g., due to limited memory resources). It tackles RL problems with large or continuous state spaces, and discrete action spaces. Using the cross-entropy method for optimization, we design an algorithm to optimize the locations of triangular MF s. Many other optimization algorithms could be

⋆ This research is financially supported by Senter, Dutch Ministry of Economic Affairs within the BSIK-ICIS project "Interactive Collaborative Information Systems" (grant no. BSIK03024), by the NWO Van Gogh grant VGP 79-99, and by the STW-VIDI project "Multi-Agent Control of Large-Scale Hybrid Systems" (DWV.6188).

applied to optimize the MF s, e.g., genetic algorithms, tabu search, pattern search, etc. Triangular MF s are chosen because they are the simplest type of MF s that ensure the convergence of fuzzy Q-iteration. The performance of the algorithm is evaluated on the classical car-on-the-hill RL benchmark (Munos and Moore, 2002; Ernst et al., 2005).

Other authors have also investigated approximate RL algorithms that change the number, position, and shape of the basis functions (Munos and Moore, 2002; Ratitch and Precup, 2004; Menache et al., 2005; Ernst et al., 2005; Mahadevan, 2005; Keller et al., 2006). Typically, basis functions are changed simultaneously with learning an RL solution. A pre-requirement for the convergence to an RL solution is the convergence of the basis functions to a final configuration. This is an open problem, usually circumvented by stopping the adaptation of the basis functions after a finite number of successive updates. Convergence proofs for fixed basis functions hold once the changes have stopped (Ernst et al., 2005).

The work that is most closely related to ours is that of Menache et al. (2005), who optimized the location and shape of a constant number of basis functions for policy evaluation. The optimization criterion used by Menache et al. (2005) was the Bellman error (residual) of the value function. We employ the empirical return of the computed policy instead.

The rest of this paper is structured as follows. Brief introductions to reinforcement learning, the fuzzy Q-iteration algorithm, and the cross-entropy method for optimization are given in Section 2. Our approach to optimizing the fuzzy MF s, together with a complete algorithm that optimizes triangular MF s using the cross-entropy method, are presented in Section 3. The performance of this algorithm is assessed in simulation in Section 4. Section 5 concludes the paper and presents some ideas for future work.

2. BACKGROUND

2.1 Reinforcement Learning

In this section, the RL task is briefly introduced and its optimal solution is characterized. The presentation is based on (Sutton and Barto, 1998; Bertsekas, 2001).

Consider a deterministic *Markov decision process*. Let X denote its state space, U its action space, $f : X \times U \to X$ its transition function, and $\rho : X \times U \to \mathbb{R}$ its reward function. As a result of the agent's action u_k in state x_k at the discrete time step k, the state changes to $x_{k+1} = f(x_k, u_k)$. At the same time, the agent receives the scalar reward signal $r_{k+1} = \rho(x_k, u_k)$, which evaluates the immediate effect of action u_k, but says nothing about its long-term effects. [1]

The agent chooses actions according to its policy $h : X \to U$. The goal of the agent is to learn a policy that maximizes, starting from the current moment in time ($k = 0$) and from any state x_0, the discounted return:

$$R_h(x_0) = \sum_{k=0}^{\infty} \gamma^k r_{k+1} = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, u_k) \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor, $u_k = h(x_k)$ (the policy), and $x_{k+1} = f(x_k, u_k)$ for $k \geq 0$. The discounted return compactly represents the rewards accumulated by the agent over the long run. The learning task is therefore to maximize the long-term performance, while only receiving feedback about the immediate, one-step performance.

This can be achieved by computing the optimal action-value function (Q-function), defined as:

$$Q^*(x, u) = \rho(x, u) + \gamma \max_h R_h(f(x, u)) \tag{2}$$

Any policy that selects for every state the action with the highest optimal Q-value: $h^*(x) = \arg\max_u Q^*(x, u)$ is optimal [it maximizes the return (1)]. RL techniques compute Q^* by using the Bellman optimality equation:

$$Q^*(x, u) = \rho(x, u) + \gamma \max_{u' \in U} Q^*(f(x, u), u') \tag{3}$$

Let the set of all Q-functions be denoted by $\mathbf{Q}$. Define the Q-iteration mapping $T : \mathbf{Q} \to \mathbf{Q}$, which computes the right-hand side of the Bellman equation for any Q-function:

$$[T(Q)](x, u) = \rho(x, u) + \gamma \max_{u' \in U} Q(f(x, u), u') \tag{4}$$

The model-based *Q-value iteration* (Q-iteration, for short) algorithm starts from an arbitrary Q-function Q_0 and at each iteration ℓ updates the Q-function using the formula $Q_{\ell+1} = T(Q_\ell)$. Q-iteration converges to Q^* as $\ell \to \infty$.

2.2 Approximate Fuzzy Q-iteration

When dealing with continuous or very large discrete state and/or actions spaces, the Q-function cannot be stored explicitly, and exact Q-iteration cannot be applied. Fuzzy Q-iteration is an approximate algorithm geared towards problems with continuous (or discrete but large) state spaces and discrete action spaces (Buşoniu et al., 2007). It employs a fuzzy partition of the state space into N fuzzy sets, each described by a membership function (MF) $\varphi_i : X \to [0, 1]$, $i = 1, \ldots, N$. A state x belongs to each set i with a degree of membership $\varphi_i(x)$. In its simplest form, the algorithm requires that:

(1) Each MF i has a singleton core x_i, i.e., there exists a unique x_i such that $\varphi_i(x_i) = 1$ (consequently, $\varphi_{\bar{i}}(x_i) = 0$ for all $\bar{i} \neq i$ by the previous assumption).
(2) The fuzzy partition is normalized, i.e., $\sum_{i=1}^{N} \varphi_i(x) = 1$, $\forall x \in X$.

Denote the discrete action space by $U = \{u_j | j = 1, \ldots, M\}$. Fuzzy Q-iteration stores an $N \times M$ matrix θ of parameters, with one component θ_{ij} corresponding to each fuzzy set-discrete action pair (i, u_j). Every iteration of the algorithm applies successively the following three mappings to the parameter matrix:

(i) The approximation mapping $F : \mathbb{R}^{N \times M} \to \mathbf{Q}$:

$$\widehat{Q}(x, u_j) = [F(\theta)](x, u_j) = \sum_{i=1}^{N} \varphi_i(x) \theta_{ij} \tag{5}$$

(ii) The Q-iteration mapping T, defined by (4).
(iii) The projection mapping $P : \mathbf{Q} \to \mathbb{R}^{N \times M}$:

$$\theta_{ij} = [P(Q)]_{ij} = Q(x_i, u_j) \tag{6}$$

[1] A stochastic formulation is possible. In that case, expected returns under the probabilistic transitions must be considered. For the sake of simplicity, only the deterministic case is considered in this paper. Our approach can be extended in an almost straightforward way to stochastic problems.

Each fuzzy Q-iteration can therefore be written as:

$$\theta_{\ell+1} = PTF(\theta_\ell) \tag{7}$$

where θ_0 is arbitrary (e.g., identically 0). To compute P in (7), only $N \cdot M$ values of $TF(\theta_\ell)$ have to be evaluated:

$$\theta_{\ell+1,ij} = \rho(x_i, u_j) + \gamma \max_{\bar{j}} [F(\theta_\ell)](f(x_i, u_j), u_{\bar{j}})$$

This requires only a finite number of evaluations of $F(\theta_\ell)$; the Q-function $F(\theta_\ell)$ does not have to be explicitly stored.

Fuzzy Q-iteration provably converges to a parameter matrix θ^* such that $F(\theta^*)$ is within a bound of the true optimum Q^* (Buşoniu et al., 2007). In practice, the algorithm terminates when $\max_{i,j} |\theta_{\ell+1,ij} - \theta_{\ell,ij}| \leq \varepsilon$ for some small ε. The approximation of the optimal policy is then given by:

$$h(x) = u_{j^*} \quad \text{with } j^* = \arg\max_j [F(\theta^*)](x, u_j) \tag{8}$$

2.3 Cross-Entropy Optimization

This section provides a brief introduction to the cross-entropy (CE) method for optimization. The presentation is based on (Rubinstein and Kroese, 2004).

Consider the following optimization problem:

$$\max_{\phi \in \Phi} s(\phi) \tag{9}$$

where $s : \Phi \to \mathbb{R}$ is the score function to maximize, and the variable ϕ takes values in the domain Φ. Denote the maximum by s^*. The CE method for optimization maintains a probability density with support Φ. At each iteration, a number of samples are drawn from this density and the score values for these samples are computed. A (smaller) number of samples that have the highest scores are kept, and the remaining samples are discarded. The probability density is then updated using the selected samples, such that at the next iteration the probability of drawing better samples is increased. The algorithm stops when the score of the worst selected sample no longer improves.

Formally, a family of probability densities $\{p(\cdot; v)\}$ has to be chosen. This family has support Φ and is parameterized by v. At each iteration τ of the CE algorithm, a number N_{CE} of samples is drawn from the density $p(\cdot; v_{\tau-1})$, their scores are computed, and the $(1 - \rho_{\text{CE}})$ quantile λ_τ of the sample scores is determined, with $\rho_{\text{CE}} \in (0, 1)$. Then, a so-called associated stochastic problem is defined, which involves estimating the probability that the score of a sample drawn from $p(\cdot; v_{\tau-1})$ is at least λ_τ:

$$\mathrm{P}_{v_{\tau-1}}(s(\phi) \geq \lambda_\tau) = \mathrm{E}_{v_{\tau-1}} I(s(\phi) \geq \lambda_\tau) \tag{10}$$

where $\mathrm{E}_{v_{\tau-1}}$ denotes expectation under $p(\cdot; v_{\tau-1})$, and I is the indicator function, equal to 1 whenever its argument is true, and 0 otherwise.

The probability (10) can be estimated by importance sampling. For this problem, an importance sampling density is one that increases the probability of the "interesting" event $s(\phi) \geq \lambda_\tau$. The best importance sampling density in the family $\{p(\cdot; v)\}$ (in the smallest cross-entropy sense) is given by the solution of:

$$\arg\max_v \mathrm{E}_{v_{\tau-1}} I(s(\phi) \geq \lambda_\tau) \ln p(\phi; v) \tag{11}$$

Algorithm 1 CE optimization

Input: density family $p(\cdot; v)$, parameter v_0, function s
Input: $\rho_{\text{CE}} \in (0, 1)$, $N_{\text{CE}} \geq 2$, $d_{\text{CE}} \geq 2$, $\tau_{\max} \geq 2$, $\varepsilon \geq 0$
1: $\tau = 1$
2: **repeat**
3: Draw samples $\phi_1, \ldots, \phi_{N_{\text{CE}}}$ from $p(\cdot; v_{\tau-1})$
4: Compute sample scores $s(\phi_l)$, $l = 1, \ldots, N_{\text{CE}}$
5: Order and index scores s.t. $s_1 \leq \cdots \leq s_{N_{\text{CE}}}$
6: $\lambda_\tau = s_{\lceil (1-\rho_{\text{CE}}) N_{\text{CE}} \rceil}$, the $(1 - \rho_{\text{CE}})$ quantile of the sample scores
7: $v_\tau = \arg\max_v \frac{1}{N_{\text{CE}}} \sum_{l=1}^{N_{\text{CE}}} I(s(\phi_l) \geq \lambda_\tau) \ln p(\phi_l; v)$
8: $\tau = \tau + 1$
9: **until** $(\tau > d_{\text{CE}}$ **and** $0 \leq \lambda_{\tau-i} - \lambda_{\tau-i-1} \leq \varepsilon$, for $i = 1, \ldots, d_{\text{CE}} - 1)$ **or** $\tau > \tau_{\max}$
Output: $\widehat{\phi}^*$, the best sample; and $\widehat{s}^* = s(\widehat{\phi}^*)$

An approximate solution of (11) is computed by:

$$v_\tau = \arg\max_v \frac{1}{N_{\text{CE}}} \sum_{l=1}^{N_{\text{CE}}} I(s(\phi_l) \geq \lambda_\tau) \ln p(\phi_l; v) \tag{12}$$

CE optimization proceeds then with the next iteration using the new density parameter v_τ (the probability (10) is never actually computed). The updated density aims at generating good samples with higher probability, thus bringing $\lambda_{\tau+1}$ closer to the optimum s^*. The goal is to eventually converge to a density which generates samples close to optimal value(s) of ϕ with very high probability. The highest score among the samples generated at all the iterations is taken as the approximate solution of the optimization problem, and the corresponding sample as an approximate optimum location.

Algorithm 1 summarizes the CE method for optimization. The algorithm is stopped when the improvement in the $(1 - \rho_{\text{CE}})$-quantile does not exceed ε for d_{CE} successive iterations, or when the maximum number of iterations $\tau_{\max}$ is reached. The number of samples N_{CE} is usually taken equal to a multiple of the dimension of v.

In certain cases, (12) has a closed-form solution. This is true e.g., when $\{p(\cdot; v)\}$ is the family of Gaussians parameterized by the mean μ and standard deviation σ (so, $v = [\mu, \sigma]^{\mathrm{T}}$). In this case, the solution of (12) is the mean and standard deviation of the best samples:

$$\mu_\tau = \frac{\sum_{l=1}^{N_{\text{CE}}} I(s(\phi_l) \geq \lambda_\tau) \phi_l}{\sum_{l=1}^{N_{\text{CE}}} I(s(\phi_l) \geq \lambda_\tau)} \tag{13}$$

$$\sigma_\tau = \sqrt{\frac{\sum_{l=1}^{N_{\text{CE}}} I(s(\phi_l) \geq \lambda_\tau) |\phi_l - \widehat{\mu}_\tau|^2}{\sum_{l=1}^{N_{\text{CE}}} I(s(\phi_l) \geq \lambda_\tau)}} \tag{14}$$

Another advantage of using the Gaussian family for CE optimization is that the algorithm can find a precise optimum location ϕ^*. This is because the Gaussian density can converge, for $\sigma \to 0$ and $\mu = \phi^*$, to a degenerate (Dirac) distribution that assigns all the probability mass to the value ϕ^*.

3. FUZZY PARTITION OPTIMIZATION

Fuzzy Q-iteration (see Section 2.2) requires the fuzzy partition to be designed beforehand. In general, prior

knowledge about the shape of the optimal Q-function is needed in this design process. When prior knowledge is not available, a large number of MF s have to be defined to provide a good coverage and resolution over the entire state space, even in areas that will eventually be irrelevant to the optimal policy.

In this section, we propose to optimize the shape of the MF s, while keeping their number constant. The proposed approach can be used when prior knowledge about the shape of the optimal Q-function is not available and the number of MF s is limited.

Let the MF s be parameterized by the vector $\phi \in \Phi$. Usually, ϕ includes information about the location and shape of the MF s. Denote the MF s by $\varphi_i(x;\phi) : X \to \mathbb{R}$, $i = 1, \ldots, N$, to highlight their dependence on ϕ. The goal is to find a parameter vector ϕ^* that maximizes the return from a set of representative initial states:

$$s(\phi) = \sum_{x_0 \in X_0} w(x_0) R_h(x_0) \tag{15}$$

where R_h is the return of the policy h (8) computed by running fuzzy Q-iteration to convergence with the fuzzy partition specified by the parameter value ϕ, and X_0 is a finite set of representative states, weighted by $w : X_0 \to (0, 1]$. Initial states that are deemed more important can be assigned larger weights, or identical weights equal to $\frac{1}{|X_0|}$ can be assigned to all the states ($|\cdot|$ denotes set cardinality).

The return (1) from each initial state x_0 is computed by simulating the system with the policy h. The discounted infinite-horizon return of a state can be approximated in finite time by simulating only the first K steps and assuming that all the subsequent rewards are 0. To guarantee that an error of at most ε is introduced, K must be:

$$K = \left\lceil \log_\gamma \frac{\varepsilon(1-\gamma)}{\|\rho\|_\infty} \right\rceil \tag{16}$$

where $\|\rho\|_\infty = \max_{x,u,\bar{x}} |\rho(x,u,\bar{x})|$ is assumed finite, and $\lceil \cdot \rceil$ produces the first integer larger than or equal to the argument.

Because the policy in the score function (15) is computed by running fuzzy Q-iteration to convergence for the parameter value ϕ, this approach does not suffer from the convergence problems associated with altering the basis functions simultaneously with running the RL algorithm. Also note that the score function (15) is directly related to the performance of the computed policy in the task.

The approach is not restricted to a particular optimization algorithm to search for ϕ^*. However, the score function (15) is a complicated function of ϕ, and is in general likely to have many local optima. This means a global optimization technique is preferable. The cross-entropy method described in Section 2.3 is chosen in this paper.

3.1 Cross-Entropy Fuzzy Q-iteration

In this section, a complete algorithm is given to optimize the fuzzy partition for fuzzy Q-iteration.

First, the shape of the MF s is specified. Let the state space dimension be n. In the sequel, it is assumed that the state space is a hyperbox centered on the origin, i.e., $X = [-x^1_{\max}, x^1_{\max}] \times \cdots \times [-x^n_{\max}, x^n_{\max}]$ where $x^m_{\max} \in (0, \infty)$, $m = 1, \ldots, n$. This assumption can be relaxed and is made here for simplicity.

Separately for each state variable x^m, a triangular fuzzy partition is defined. Such a partition is completely determined by an array of core values $c^m_1 < \cdots < c^m_{N_m}$, which generates N_m triangular MF s φ_{i_m}, $i_m = 1, \ldots, N_m$. The first and last core values are always equal to the limits of the domain, $c^m_1 = -x^m_{\max}$, $c^m_{N_m} = x^m_{\max}$. Figure 1 presents an example of a single-dimensional triangular partition.

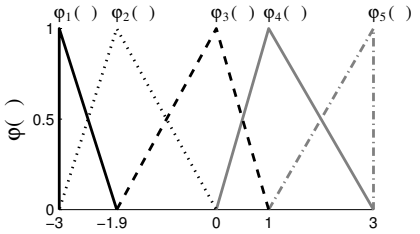

Fig. 1. Triangular fuzzy partition for a variable $x \in [-3, 3]$. Each fuzzy set is plotted in a different style. Core values are given on the horizontal axis.

The fuzzy partition of the n-dimensional state space is then defined as follows. The partition contains a fuzzy set for each combination $(i_1, \ldots, i_n)$ of single-dimensional sets. The MF of such a composite set is defined as the product of the individual MF s: $\prod_{m=1}^n \varphi_{i_m}(x^m)$. This composite MF has the core $[c^1_{i_1}, \ldots, c^n_{i_n}]^{\mathrm{T}}$. It is easy to verify that the fuzzy partition computed in this way still satisfies Assumptions 1, 2 from Section 2.2. The resulting approximator F is equivalent to multi-linear interpolation on the grid of cores.

The parameters to optimize are the scalar free cores on all the axes (the first and last core values on each axis are not free). The number of free cores is therefore $N_\phi = \sum_{m=1}^n (N_m - 2)$. The parameter vector ϕ can be obtained by concatenating the free cores, $\phi = [c^1_2, \ldots, c^1_{N_1-1}, \ldots\ldots, c^n_2, \ldots, c^n_{N_n-1}]^{\mathrm{T}}$. The domain of ϕ is $\Phi = (-x^1_{\max}, x^1_{\max})^{N_1-2} \times \cdots \times (-x^n_{\max}, x^n_{\max})^{N_n-2}$.

The goal is to find a parameter vector ϕ^* that maximizes the score function (15), using CE optimization. To this end, a family of probability density functions has to be chosen. We choose a density with independent Gaussian components for each of the N_ϕ parameters. [2] The Gaussian density is a usual choice for continuous CE optimization, mainly for the reasons stated at the end of Section 2.3. The density parameter v therefore consists of the mean μ and standard deviation σ, each of them a vector with N_ϕ elements. Because the support of this density is $\mathbb{R}^{N_\phi}$, which is larger than Φ, samples that do not belong to Φ are rejected and generated again. Equations (13) and (14) can be used to update μ and σ.

Finally, the initialization of μ and σ is specified as:

$$\mu_0 = \mathbf{0}$$
$$\sigma_0 = [x^1_{\max}, \ldots, x^1_{\max}, \ldots\ldots, x^n_{\max}, \ldots, x^n_{\max}]^{\mathrm{T}}$$

[2] If for a sample of the CE algorithm two or more core values on an axis m are identical, a single triangular MF with that core value is used in the fuzzy Q-iteration algorithm.

where each bound $x^m_{\max}$ is replicated $N_m - 2$ times, for $m = 1, \ldots, n$. These values ensure a good coverage of the state space with samples in the first iteration.

4. EXAMPLE: CAR ON THE HILL

In this section, we apply the algorithm developed in Section 3 to the car-on-the-hill problem. This is a classical benchmark for approximate RL . For instance, Munos and Moore (2002) used this problem as a primary benchmark for their variable-resolution value iteration technique. Ernst et al. (2005) used the car on the hill, among other examples, to validate an RL algorithm that approximates value functions with ensembles of regression trees.

In this problem, a point mass (the 'car') has to be driven past the top of a frictionless hill by applying a horizontal force (Figure 2). For some initial states, the maximum available force is not sufficient to drive the car straight up the hill. Instead, it has to be driven up the opposite slope (left) and gain momentum prior to accelerating towards the goal (right).

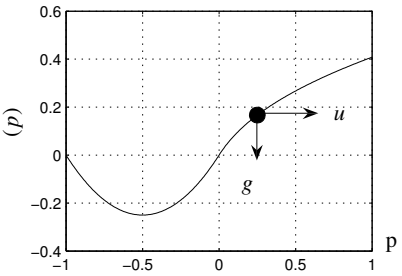

Fig. 2. The car on the hill. The 'car' is represented as a black bullet, its goal is to drive out of the figure to the right.

The horizontal position of the car (in meters) is denoted by p, and the shape of the hill is:

$$H(p) = \begin{cases} p^2 + p & \text{if } p < 0 \\ \dfrac{p}{\sqrt{1+5p^2}} & \text{if } p \geq 0 \end{cases}$$

The dynamics of this problem are (Ernst et al., 2005):

$$\ddot{p} = \frac{u}{1 + H'(p)^2} - \frac{g\, H'(p)}{1 + H'(p)^2} - \frac{\dot{p}^2\, H'(p)\, H''(p)}{1 + H'(p)^2} \qquad (17)$$

where a unity mass was assumed and $g = 9.81$ is the gravitational acceleration. The notations $\dot{p}$ and $\ddot{p}$ indicate the first and second time derivatives of p, while $H'(p)$ and $H''(p)$ denote the first and second derivatives of H with respect to p.

The state signal consists of the horizontal position and speed of the car, $x = [p, \dot{p}]^{\mathrm{T}}$, and the control signal u is the applied force. The state space is $X = [-1, 1] \times [-3, 3]$ plus a terminal state (see below), and the discrete action space is $U = \{-4, 4\}$. Whenever x_{k+1} is not in X (i.e., the position or speed exceed the bounds), it is considered that the terminal state has been reached. The goal is drive past the top of the hill to the right with a speed within the allowed limits. Reaching the terminal state in any other way is considered a failure. The reward function chosen to express this goal is (Ernst et al., 2005):

$$\rho(x_k, u_k) = \begin{cases} -1 & \text{if } p_{k+1} < -1 \text{ or } |\dot{p}_{k+1}| > 3 \\ 1 & \text{if } p_{k+1} > 1 \text{ and } |\dot{p}_{k+1}| \leq 3 \\ 0 & \text{otherwise} \end{cases} \qquad (18)$$

The discount factor is $\gamma = 0.95$. A discrete time step $T_{\mathrm{s}} = 0.1\,\mathrm{s}$ is chosen.

To apply CE fuzzy Q-iteration, the following grid of representative initial states is used:

$$X_0 = \{-1, -0.75, -0.5, \ldots, 1\} \times \{-3, -2, -1, \ldots, 3\}$$

where each point is uniformly weighted by $\frac{1}{|X_0|}$ in (15). The parameters of the algorithm are set as follows: $N_{\mathrm{CE}} = 5 \cdot 2 \cdot N_\phi$ (5 times the number of parameters needed to describe the probability density used inside the CE optimization algorithm), $\rho_{\mathrm{CE}} = 0.05$, $d_{\mathrm{CE}} = 5$, $\tau_{\max} = 50$, $\varepsilon = 10^{-3}$. This same value of ε is used as admissible error in the score evaluation [see (16)], the fuzzy Q-iteration convergence threshold, and the CE convergence threshold. These are typical default values, with little or no tuning performed.

The algorithm is run with an identical number of MF s for each state variable, $N_1 = N_2$, and this number is gradually increased from 3 to 20. [3] Each such experiment is run 10 times, and the average score, together with the maximum and the minimum score in any run, are reported for every value of $N_1 = N_2$. Figure 3-left compares the score obtained by CE optimization with the score of the policies computed with fuzzy Q-iteration with equidistant MF s, for $N_1 = N_2$ varying from 3 to 50. Figure 3-right gives a detailed view of the CE optimization scores. The graphs also show the optimal score, computed by an exhaustive search for optimal open-loop control sequences.

For the same number of MF s, CE fuzzy Q-iteration consistently and reliably provides better performance than fuzzy Q-iteration with equidistant MF s. Starting with 15 MF s on each axis, the worst performance of the CE algorithm is still better than the best performance of equidistant MF s for any $N_1 = N_2 \leq 50$. For $N_1 = N_2 \geq 12$, the performance of CE fuzzy Q-iteration is very close to the optimal one.

Figure 4 presents a typical convergence plot. Note that the convergence is not monotonous; this is because the algorithm uses random samples in every iteration.

5. CONCLUSION

In this paper, we have proposed an approach to optimize the shapes of MF s for approximate fuzzy Q-iteration. The optimization criterion measures the actual performance of the computed policy in the task, using simulation to compute the return from a representative set of initial states. Using the cross-entropy method for optimization, an algorithm was designed to optimize the locations of triangular MF s. For the car-on-the-hill benchmark, this algorithm showed better performance than fuzzy Q-iteration with equidistant triangular sets.

[3] The experiments stop at 20 MF s to limit computation times per experiment in the order of hours on our 3 GHz P-IV machine, using MATLAB 7.1.

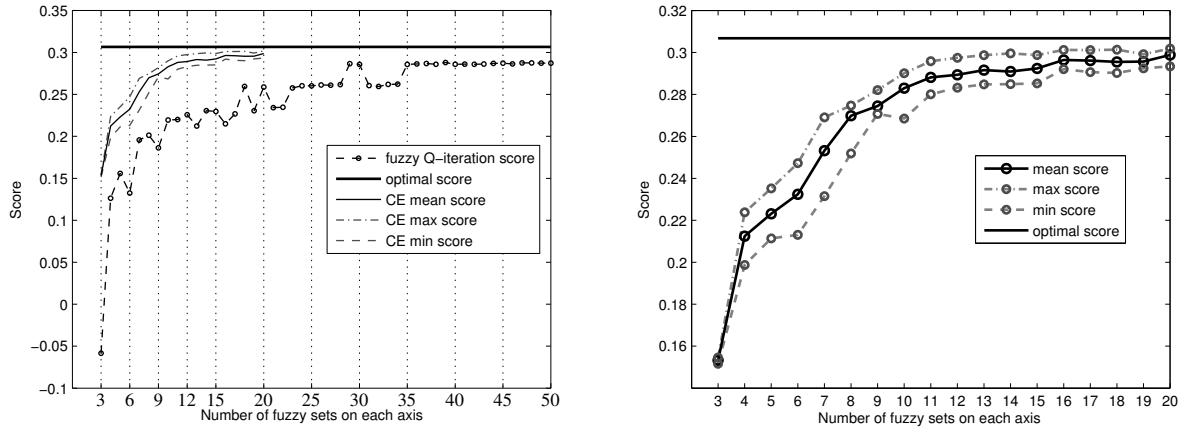

Fig. 3. *Left:* Comparison between CE fuzzy Q-iteration, and fuzzy Q-iteration with equidistant fuzzy sets. *Right:* Detailed view of the CE fuzzy Q-iteration scores.

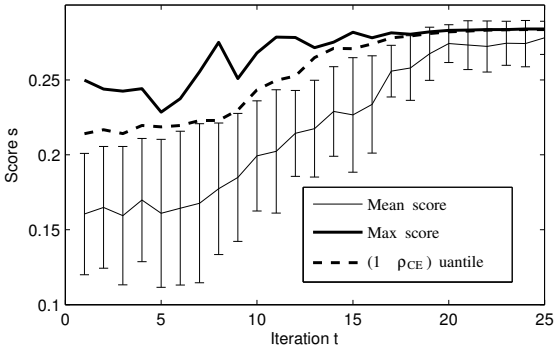

Fig. 4. Convergence for a typical run of CE fuzzy Q-iteration ($N_1 = N_2 = 12$). The algorithm converged in 25 iterations.

The triangular MF s used in this paper make fuzzy Q-iteration exponentially complex in the number of state dimensions. Since fuzzy Q-iteration is run for every set of MF s considered by the CE optimization, the resulting optimization procedure is very computationally intensive. Other types of MF s (e.g., Gaussian) could be used to obtain a fuzzy partition with less than exponential complexity. However, even for such MF s, a good solution might generally require exponentially many MF s.

While in this paper the CE method for optimization was employed, there is no obstacle in principle to applying any optimization technique to determine a good set of MF s. In particular, other metaheuristic optimization techniques like genetic algorithms, tabu search, pattern search, etc., could be used.

REFERENCES

Bertsekas, D.P. (2001). *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 2nd edition.

Buşoniu, L., Ernst, D., De Schutter, B., and Babuška, R. (2007). Fuzzy approximation for convergent model-based reinforcement learning. In *Proceedings 2007 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE-07)*, 968–973. London, UK.

Ernst, D., Geurts, P., and Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6, 503–556.

Gordon, G. (1995). Stable function approximation in dynamic programming. In *Proceedings Twelfth International Conference on Machine Learning (ICML-95)*, 261–268. Tahoe City, US.

Keller, P.W., Mannor, S., and Precup, D. (2006). Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proceedings Twenty-Third International Conference on Machine Learning (ICML-06)*, 449–456. Pittsburgh, US.

Mahadevan, S. (2005). Samuel meets Amarel: Automating value function approximation using global state space analysis. In *Proceedings 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference (AAAI-05)*, 1000–1005. Pittsburgh, US.

Menache, I., Mannor, S., and Shimkin, N. (2005). Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, 134, 215–238.

Munos, R. and Moore, A. (2002). Variable-resolution discretization in optimal control. *Machine Learning*, 49(2-3), 291–323.

Ormoneit, D. and Sen, S. (2002). Kernel-based reinforcement learning. *Machine Learning*, 49(2–3), 161–178.

Ratitch, B. and Precup, D. (2004). Sparse distributed memories for on-line value-based reinforcement learning. In *Proceedings 15th European Conference on Machine Learning (ECML-04)*, volume 3201 of *Lecture Notes in Computer Science*, 347–358. Pisa, Italy.

Rubinstein, R.Y. and Kroese, D.P. (2004). *The Cross Entropy Method. A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Information Science and Statistics. Springer.

Sutton, R.S. and Barto, A.G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Szepesvári, C. and Smart, W.D. (2004). Interpolation-based Q-learning. In *Proceedings Twenty-First International Conference on Machine Learning (ICML-04)*, 791–798. Bannf, Canada.