Finite-Element methods with local triangulation refinement for continuous Reinforcement Learning problems

Rémi Munos

DASSAULT-AVIATION, DGT-DTN-EL-Et.Avances,
78 quai Marcel Dassault, 92214 Saint-Cloud, FRANCE
and
CEMAGREF, LISC, Parc de Tourvoie,
BP 121, 92185 Antony Cedex, FRANCE
Tel : (0)1 40 96 61 79. Fax : (0)1 40 96 60 80
E-mail: Remi.Munos@cemagref.fr

Abstract. This paper presents a reinforcement learning algorithm designed for solving optimal control problems for which the state space and the time are continuous variables. Like Dynamic Programming methods, reinforcement learning techniques generate an optimal feed-back policy by the mean of the value function which estimates the best expectation of cumulative reward as a function of initial state. The algorithm proposed here uses finite-elements methods for approximating this function. It is composed of two dynamics : the *learning dynamics*, called *Finite-Element Reinforcement Learning*, which estimates the values at the vertices of a triangulation defined upon the state space, and the *structural dynamics*, which refines the triangulation inside regions where the value function is irregular. This mesh refinement algorithm intends to solve the problem of the combinatorial explosion of the number of values to be estimated. A formalism for reinforcement learning in the continuous case is proposed, the Hamilton-Jacobi-Bellman equation is stated, then the algorithm is presented and applied to a simple two-dimensional target problem.

1 Introduction

In this paper, we are concerned with adaptive non-linear control problems, like target or obstacle problems, viability problems or optimization problems, for which the state space and the time are continuous variables. In order to define an optimal control, reinforcement learning (RL) builds the *value function* which estimates the best expectation of future reinforcement for all possible controls as a function of initial state.

In the continuous case, the value function has to be represented by a general approximation system using a finite set of parameters. Several techniques have been proposed, for example by using neural networks (see [Bar90], [Gul92], [Lin93] and many others), fuzzy controllers (see [Now95]) or other approximation systems (see the sparse-coarse-coded CMACs of [Sut96]). However, as it has been pointed out in [Bai95] and [BM95], in general, the combination of reinforcement learning algorithms with such function approximators does not converge. The *Residual-gradient advantage updating* proposed in [HBK96] is a convergent algorithm in the sense of the convergence of gradient descent methods. But the problem with these methods is how to find a suitable architecture for the network, i.e. which permits to approximate the value function. Besides, gradient descent methods only insure local optimum.

Here, we present a direct reinforcement learning algorithm that uses finite-element methods with a local mesh refinement process for approximating the value function. The algorithm consists of the combination of these two dynamics :

- **The learning dynamics** : for a given triangulation of the state space, the *Finite-Element Reinforcement Learning* (FERL) modifies the values of the vertices according to the reinforcement obtained during the running of trajectories, so the value function is approximated with a piecewise linear function. The FERL algorithm used here is a convergent RL algorithm (see [Mun96]).
- **The structural dynamics** that locally refines the mesh of the triangulation in order to build an relevant triangulation whose precision depends on the regularities of the value function. Its starts with a rough triangulation composed of a small number of large simplexes (knowledge of a novice) and builds an accurate triangulation (knowledge of an expert) according to the learning dynamics. This structural dynamics can be seen as a categorization process via reinforcement from the environment.

Section 2 introduces a formalism for the study of reinforcement learning in the continuous case. *Section 3* describes the FERL algorithm for a given triangulation of the state space (i.e. the learning dynamics). *Section 4* presents the structural dynamics. *Section 5* illustrates this algorithm with an example of target problem in a two dimensional space.

2 Reinforcement Learning, the Continuous Case

In order to estimate the performances of a reinforcement learning algorithm, we need to compare the function computed by the algorithm to the value function of the continuous process.

In the following, we consider *deterministic systems* with *infinite time horizon* and *discounted reinforcement.* Let $x(t) \in \bar{O}$ be the state of the system with the state space O bounded, open subset of $\mathbb{R}^d$. The evolution of the system depends on the current state $x(t)$ and control $u(t)$; it is defined by a differential equation :

$$\frac{d}{dt}x(t) = f(x(t), u(t))$$

where the control $u(t)$ is a bounded, Lebesgue measurable function with values in a compact U. From any initial state x, the choice of the control $u(t)$ leads to a unique trajectory $x(t)$. Let τ be the exit time of $x(t)$ from $\bar{O}$ (with the convention that if $x(t)$ always stays in $\bar{O}$, then $\tau = \infty$). Then, we define the discounted reinforcement functional of state x, control $u(.)$:

$$J(x; u(.)) = \int_0^\tau \gamma^t r(x(t), u(t))dt + \gamma^\tau R(x(\tau))$$

where $r(x, u)$ is the *running reinforcement* and $R(x)$ the *terminal reinforcement.* γ is the *discount factor* $(0 \leq \gamma < 1)$.

The **objective of the control problem** is to find the optimal feed-back control $u^*(x)$ that optimizes the reinforcement functional for initial state x.

RL techniques belongs to the class of DP methods which compute the optimal control by the means of the **value function :**

$$V(x) = \sup_{u(.)} J(x; u(.)) \tag{1}$$

which is the maximum value of the functional as a function of initial state x.

In the RL approach, the system tries to approximate this function without knowing the state dynamics f nor the reinforcement functions r, R. Thus, RL appears as a constructive and iterative process, based on experience, that estimates the value function by successive approximations.

Following the dynamic programming principle, the value function satisfies a first-order nonlinear partial differential equation called the *Hamilton-Jacobi-Bellman* equation (see [FS93] for a survey).

Theorem 1 (Hamilton-Jacobi-Bellman). *If V is differentiable at $x \in O$, let $DV(x)$ be the gradient of V at x, then the following HJB equation holds at x.*

$$V(x) \ln \gamma + \sup_{u \in U} [DV(x).f(x, u) + r(x, u)] = 0$$

Besides, V satisfies the following boundary condition :

$$V(x) \geq R(x) \text{ for } x \in \partial O$$

The challenge of learning the value function is motivated by the fact that from V, we can deduce the following optimal feed-back control policy :

$$u^*(x) = \arg \sup_{u \in U} [DV(x).f(x, u) + r(x, u)]$$

In the following, we intend to approximate the value function with piecewise linear functions defined by their values at the vertices of a triangulation of the state space.

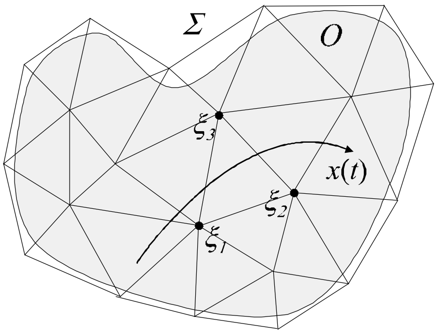


Fig. 1. Triangulation Σ of the state space O. A trajectory $x(t)$ crosses the simplex (ξ_1, ξ_2, ξ_3).

3 The Learning Dynamics

Let us consider a triangulation Σ such that the set of simplexes covers O (see figure 1). By using a finite-element convergent approximation scheme (derived from [Kus90]), the continuous control problem may be approximated by a Markovian Decision Process whose state space is the set of vertices $\{\xi\}$.

The value function V is approximated by a piecewise linear function V^Σ defined by its values at the vertices $\{\xi\}$. The value of V^Σ at any point x inside some simplex $(\xi_0, ..., \xi_d)$ is a linear combination of V^Σ at the vertices $\xi_0, ..., \xi_d$:

$$V^\Sigma(x) = \sum_{i=0}^{d} \lambda_{\xi_i}(x) V^\Sigma(\xi_i) \text{ for all } x \in \text{simplex } (\xi_0, ..., \xi_d)$$

with $\lambda_{\xi_i}(x)$ the *barycentric coordinates* of x inside the simplex $(\xi_0, ..., \xi_d) \ni x$. (We recall that the definition of the barycentric coordinates $\lambda_{\xi_i}(x)$ is such that: $\sum_{i=0}^{d} \lambda_{\xi_i}(x).(\xi_i - x) = 0$ and: $\sum_{i=0}^{d} \lambda_{\xi_i}(x) = 1$).

We approximate the Hamilton-Jacobi-Bellman equation with the Finite-Element scheme:

$$V^\Sigma(\xi) = \sup_{u \in U} \left[\gamma^{\tau(\xi,u)}.V^\Sigma(\eta(\xi,u)) + \tau(\xi,u) r(\xi,u) \right] \tag{2}$$

where $\eta(\xi,u)$ is the projection of ξ in a direction parallel to $f(\xi,u)$ onto the opposite side of the simplex (see figure 2) and $\tau(\xi,u)$ is such that :

$$\eta(\xi,u) = \xi + \tau(\xi,u) f(\xi,u)$$

From the linearity of V^Σ upon the simplexes, (2) is equivalent to:

$$V^\Sigma(\xi) = \sup_{u \in U} \left[\gamma^{\tau(\xi,u)}. \sum_{j=1}^{d} \lambda_{\xi_j}(\eta(\xi,u)).V^\Sigma(\xi_j) + \tau(\xi,u) r(\xi,u) \right]$$

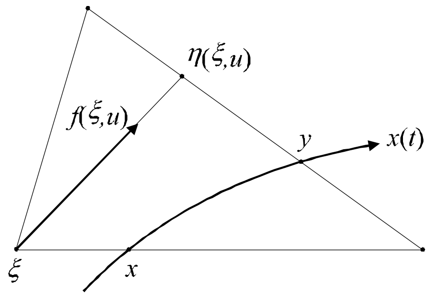


Fig. 2. A trajectory going through a simplex. $\eta(\xi, u)$ is the projection of ξ in a direction parallel to $f(\xi, u)$ onto the opposite side of the simplex. $\frac{y-x}{\lambda_\xi(x)}$ is a good approximation of $\eta(\xi, u) - \xi$.

which is a Dynamic Pogramming equation for a finite Markov Decision Process whose state space is the set of vertices $\{\xi\}$, and the probabilities of transition from state ξ to the adjacent states $\{\xi_j\}_{j=1..d}$ with control u are the barycentric coordinates $\lambda_{\xi_j}(\eta(\xi, u))$.

By introducing the Q-values $Q^\Sigma(\xi, u)$ such that $V^\Sigma(\xi) = \sup_{u\in U} Q^\Sigma(\xi, u)$ and thanks to a contraction property due to the discount factor γ, Dynamic Programming theory (see [Ber87]) insures that there is a unique solution, called V^Σ, that satisfies equations :

$$\begin{aligned} Q^\Sigma(\xi, u) &= \gamma^{\tau(\xi,u)}.V^\Sigma(\eta(\xi, u)) + \tau(\xi, u)r(\xi, u) \text{ for } \xi \in O \\ Q^\Sigma(\xi, u) &= R(\xi) \text{ for } \xi \notin O \end{aligned} \tag{3}$$

Here, it is not possible to use directly a Real Time Dynamic Programming algorithm (see [BBS91]) for solving iteratively equation (3) because the dynamics f (thus $\eta(\xi, u)$ and $\tau(\xi, u)$) is unknown. The model-based approach should be to build in a first time a model of the dynamics f and then to use DP methods with this model. In this paper, we are interested in the model-free approach which consists in an on-line and direct learning, that is which does not build any model of the dynamics. The Finite-Element Reinforcement Learning (introduced in [Mun96]) is a model-free RL algorithm that uses approximation of $\eta(\xi, u)$ and $\tau(\xi, u)$ thanks to the available on-line knowledge.

3.1 Presentation

Suppose that a trajectory $x(t)$ enters inside simplex T at point $x = x(t_1)$. At time t_1 suppose that a control u is chosen and kept until the trajectory leaves T at $y = x(t_2)$ (see figure 2). Let $\tau_x = t_2 - t_1$. Let $T_{in} \ni x$ be the $(d-1)$-input-simplex and $T_{out} \ni y$ the $(d-1)$-output-simplex.

The algorithm presented in the next section is the iterated version of equation (3) which uses:

$$\frac{V^{\Sigma}(y)-V^{\Sigma}(x)}{\lambda_{\xi}(x)}+V^{\Sigma}(\xi) \text{ as an approximation of } V^{\Sigma}(\eta(\xi,u))$$

$$\frac{\tau_x}{\lambda_{\xi}(x)} \text{ as an approximation of } \tau(\xi,u).$$

These approximations come from the linearity of V^{Σ} inside T and that from Thales' theorem, $\frac{y-x}{\lambda_{\xi}(x)}$ is an approximation of $\eta(\xi,u)-\xi$.

3.2 The Finite-Element Reinforcement Learning

In the following, we assume that the action space U is finite. Let $Q_n^{\Sigma}(\xi,u)$ and $V_n^{\Sigma}(\xi)$ be the iterated values of $Q^{\Sigma}(\xi,u)$ and $V^{\Sigma}(\xi)$. Let us choose a constant $\lambda \in (0,1]$ close to zero. Initial values $Q_0^{\Sigma}(\xi,u)$ are initialized to any value.

Consider a trajectory $x(.)$ going through a simplex T with a control u. When the trajectory leaves T at y, if the following conditions:

* $\lambda_{\xi}(x) \geq \lambda$ (this relation eliminates cases for which $\lambda_{\xi}(x)$ is too small)
* $\forall \xi_i \in T_{in} \cap T_{out}, \lambda_{\xi_i}(y) \geq \lambda_{\xi_i}(x)+\lambda$ (these relations imply that $y-x$ strictly belongs to the cone of vertex ξ and base T_{out} and insure that for small simplexes, $\eta(\xi,u) \in T_{out}$).

are satisfied, then update the Q-value of vertex ξ opposite to the exit side T_{out} for control u with :

$$\boxed{Q_{n+1}^{\Sigma}(\xi,u)=\gamma^{\frac{\tau_x}{\lambda_{\xi}(x)}}\left(\frac{V_n^{\Sigma}(y)-V_n^{\Sigma}(x)}{\lambda_{\xi}(x)}+V_n^{\Sigma}(\xi)\right)+\frac{\tau_x}{\lambda_{\xi}(x)}r(x,u)} \tag{4}$$

When the trajectory reaches the border of the state space, at time τ, the closest vertex $\xi_j \notin O$ from the exit point $x(\tau)$ is updated with :

$$V_{n+1}^{\Sigma}(\xi_j)=R(x(\tau))$$

Remark. With some additional hypotheses on the dynamics f and the regularities of r, R and ∂O, this algorithm converges to the value function of the continuous problem as n tends to infinity and the size of the simplexes tends to zero (see [Mun96]).

Meanwhile, for a given triangulation, the V_n^{Σ}-values do not converge as n tends to infinity. In order to insure the convergence, we need to combine the learning dynamics with a triangulation refinement process, called structural dynamics.

4 The Structural Dynamics

The structural dynamics intends to build a triangulation such that the simplexes enclose states whose value function is almost linear. Here, we choose the "general towards specialized" approach : the initial triangulation is composed of a small number of large simplexes ; then the structural dynamics refines the triangulation at places where the value function is irregular (the simplexes of low reinforcement discriminant skill).

4.1 The Delaunay Triangulation

In this paper, we chose the Delaunay triangulation, which is a very used triangulation technique for finite-elements methods when the state space is of low dimensionality. Delaunay triangulation is built from the basis of a set of vertices and is close related to the Voronoi diagram (there is a dual relationship). The study of Delaunay triangulations and their properties are beyond the scope of this paper (see for example some recent work: [Mid93], [Rup94]). We will only give a definition in the 2-dimensional case :

Given a finite set of points, 3 points contribute a triangle to the Delaunay triangulation if the circumscribing circle through those points contains no other points in its interior.

4.2 The Delaunay Refinement Process

The refinement process consist of adding new vertices and compute the Delaunay triangulation associated to the new set of vertices (we use the incremental Watson' algorithm [Wat81], see figure 3).

The choice of adding a new vertex inside a given simplex depends on a criterion based on the expected reinforcement average variation : the *Q-deviation.* For each vertex ξ and control u, the Q-deviation $E_n^{\Sigma}(\xi, u)$ is incrementally updated at the same time as the Q-value is by :

$$E_{n+1}^{\Sigma}(\xi, u) = \frac{1}{n+1}\left(n.E_n^{\Sigma}(\xi, u) + \left[Q_{n+1}^{\Sigma}(\xi, u) - Q_n^{\Sigma}(\xi, u)\right]^2\right)$$

and the Q-deviation $E_n^{\Sigma}(T)$ of a simplex T is the sum of the Q-deviations of its vertices ξ for controls u such that the dynamics $f(\xi, u)$ "goes inside" T.

The refinement rule is the following:

If the Q-deviation of a simplex T is superior to some value, add to the set of vertices the barycenter of T (for example, in figure 3, the barycenter of the gray simplex is added to the list of vertices).

Remark. The refinement of boundary simplexes (those whose all vertices except one are outside O) consists in adding the barycenter of the vertices that are outside O (see an illustration in section *5).*

Remark. During the triangulation refinement process, some initial averaged Q-values (for the next learning dynamics) are attributed to the points added to the list of vertices. This insures a continuous learning through successive structural dynamics processes.

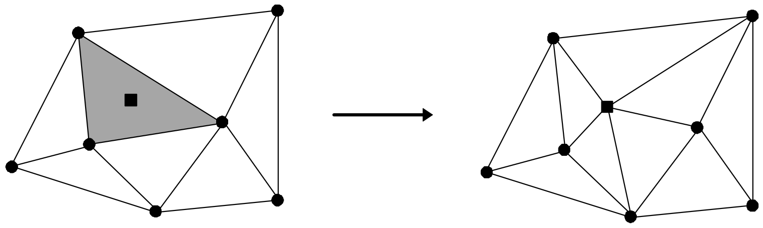


Fig. 3. Addition of a new point (the square dot) to the set of vertices and the resulting Delaunay triangulation

For a given triangulation, we have to distinguish two phases during the learning dynamics :

- The *transitional phase* during which reinforcements are propagated upon the whole state space according to the dynamics.
- The *almost stationary phase* during which the Q-values oscillate around an average value.

Either a Q-value converges, so its Q-deviation tends to 0, or it does not and the Q-deviation does not tend to 0. So the Q-deviations are a measure of the irregularity of the expected reinforcement (thus indirectly of the value function).

The structural dynamics consists in refining the triangulation at places where the Q-deviation of the simplexes are high. Then the new triangulation becomes more precise at the irregularities of the value function.

A succession of learning and structural dynamics is executed until the Q-deviations of all simplexes are small enough.

Remark. The refinement criterion used here, based on the Q-deviations, is very simple and may be improved by taking into account other factors like the existence of a change in the optimal control inside a simplex, or the coherence of the values at the vertices of a simplex depending on the local optimal control. The Q-deviation criterion is only an illustration of a possible local measure for triangulation refinement process, and it is used in the following simulation.

5 Illustration with a Simple Target Problem

5.1 Description

Let us consider a mass moving on a bar (segment $[-1,1]$) which has to stop at one of its extremities. The control consists in pushing the mass with a constant strength either on the left or on the right (the control is $u = \pm 1$). The state of the system is: $x = (y, v)$ with y ($\in [-1,1]$) the position and v ($\in [-2,2]$) the velocity of the mass. Thus, the dynamics of the system is :

$$\begin{cases} y'(t) = v(t) \\ v'(t) = u(t) \end{cases}$$

Let the running reinforcement $r = 0$ and the terminal reinforcement R depends on the side from which the mass leaves the bar (see figure 4): if the bar reaches the left extremity with a null velocity (left target) then it receives $R = +1$, if it reaches the right extremity with a null velocity (right target), it receives $R = +2$. If it reaches an extremity with a positive velocity, the terminal reinforcement will decrease with the velocity (until $R = -1$ for the maximal velocity).

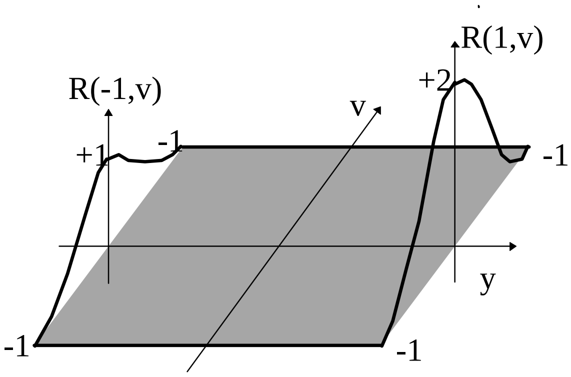


Fig. 4. The terminal reinforcement $R(y, v)$. $R = +1$ (resp. $R = +2$) at the left (resp. right) extremity of the bar for a null velocity. R decreases until -1 for the maximal velocity.

Thus, the objective of the learning is twice: first, the system has to learn to reach, as fast as possible, each extremity of the bar with a low velocity, as much as possible. Then, it has to learn to choose which extremity (the right one with a possible reinforcement of $+2$ or the left one with $+1$) is the best depending on its position and velocity (thus on the time required for reaching the target).

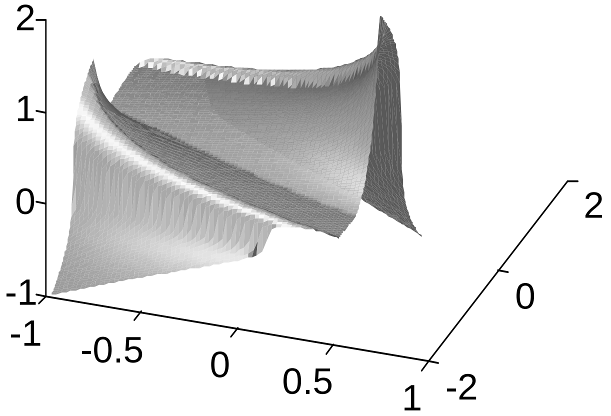


Fig. 5. The (exact) value function $V(x)$.

5.2 The Optimal Solution

In this example, it is easy to compute exactly the value function (see figure 5) and the optimal control. Thus we can estimate the difference between the computed V_n-values and the optimal value function. A measure of this approximation error is the error sup :

$$\text{Error sup} = \sup_{x \in O} \left| V_n^{\Sigma}(x) - V(x) \right|$$

5.3 Numerical Results

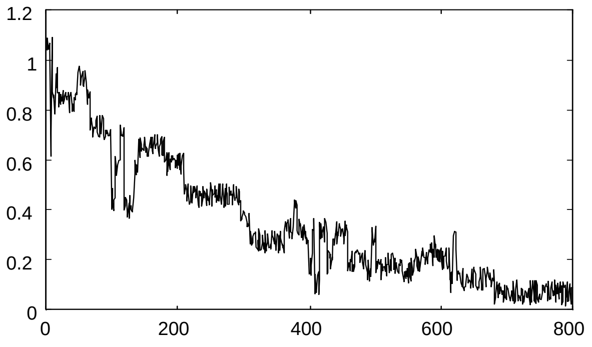
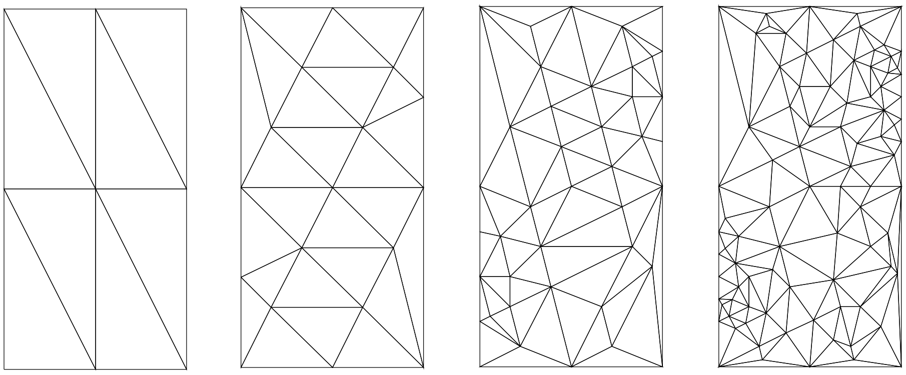


Fig. 6. The error sup as a function of the number of running trajectories.

The results of the simulation are depicted in figure 6. This approximation error is given as a function of the number n of running trajectories for successive triangulations Σ_1, Σ_2, Σ_3, Σ_4 (see figure 7).



Triangulation Σ_1 Σ_2 Σ_3 Σ_4

Figure 7: Successive triangulations during the simulation.

Initial triangulation (Σ_1) corresponds to $1 \leq n \leq 100$. For $n = 100$, the simplexes whose Q-deviation is higher than 0.001 are refined, which leads to triangulation Σ_2. In a same way, triangulation Σ_3 occurs at $n = 200$, and triangulation Σ_4 occurs at $n = 400$ and is kept until the end of the simulation.

5.4 Analyze of the Results

Globally, the error sup tends to 0 as successive learning and structural dynamics are executed. At the end of the simulation, the error sup is lower than 0.1 and continues to oscillate around 0.05.

By comparison to the (exact) value function of figure 5, we observe that the refinement process occurs at places where V is the most irregular.

Comparison with a constant and uniform triangulation: We have run a simulation with a constant triangulation composed of 160 (number lightly superior to the 153 simplexes of triangulation Σ_4) uniformly distributed simplexes and obtained with 800 trajectories an error sup of 0.22. This result indicates the benefit of using the local refinement structural dynamics.

6 Conclusion

The combination of the learning and structural dynamics provides an interesting reinforcement learning algorithm for the continuous and deterministic case. A first improvement should be to study the stochastic case (when the evolution of the system is governed by a stochastic differential equation) for which we could use the Q-learning (see [Wat89]) version of FERL rule:

$$\Delta Q_n^{\Sigma}(\xi, u) = \alpha_n \left[\gamma^{\frac{\tau_x}{\lambda_\xi(x)}} \left(\frac{V_n^{\Sigma}(y) - V_n^{\Sigma}(x)}{\lambda_\xi(x)} + V_n^{\Sigma}(\xi) \right) - Q_n^{\Sigma}(\xi, u) + \frac{\tau_x}{\lambda_\xi(x)} r(x, u) \right]$$

with some decreasing learning rate α.

The local refinement criterion based on the Q-deviations generates a triangulation that adapts to the regularities of the value function. Meanwhile, the refinement process used here is very simple and sometimes generates more simplexes than necessary. A possible improvement should be to consider a structural dynamics including both "bottom-up" and "top-down" processes, for example by suppressing some points at places where locally, the computed Q-values are regular, or by moving some vertices according to the dynamics f. Another approach should consist on a triangulation initialized around terminal reinforcements and progressively increasing inside the state space during the running of trajectories.

References

[Bai95] Leemon C. Baird. Residual algorithms : Reinforcement learning with function approximation. *Machine Learning : proceedings of the Twelfth International Conference*, 1995.

[Bar90] Andrew G. Barto. *Neural networks for control.* W.T. Miller, R.S.Sutton, P.J. Werbos editors. MIT press, Cambridge, Massachussetts, 1990.

[BBS91] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Real-time learning and control using asynchronous dynamic programming. Technical Report 91-57, Computer Science Department, University of Massachusetts, 1991.

[Ber87] Dimitri P. Bertsekas. *Dynamic Programming : Deterministic and Stochastic Models.* Prentice Hall, 1987.

[BM95] J.A. Boyan and A.W. Moore. Generalization in reinforcement learning : Safely approximating the value function. *Advances in Neural Information Processing Systems*, 7, 1995.

[FS93] Wendell H. Fleming and H. Mete Soner. *Controlled Markov Processes and Viscosity Solutions.* Applications of Mathematics. Springer-Verlag, 1993.

[Gul92] Vijay Gullapalli. *Reinforcement Learning and its application to control.* PhD thesis, University of Massachussetts, Amherst., 1992.

[HBK96] Mance E. Harmon, Leemon C. Baird, and A. Harry Klopf. Reinforcement learning applied to a differential game. *Adaptive Behavior*, 4:3–28, 1996.

[Kus90] Harold J. Kushner. Numerical methods for stochastic control problems in continuous time. *SIAM J. Control and Optimization*, 28:999–1048, 1990.

[Lin93] Long-Ji Lin. *Reinforcement Learning for Robots using Neural Networks.* PhD thesis, Carnegie Mellon University, Pittsburg, Pennsylvania, 1993.

[Mid93] Terje Midtbo. *Spatial Modelling by Delaunay Networks of Two and Three Dimensions.* PhD thesis, Norwegian Institute of Technology, 1993.

[Mun96] Rémi Munos. A convergent reinforcement learning algorithm in the continuous case : the finite-element reinforcement learning. *International Conference on Machine Learning*, 1996.

[Now95] Ann Nowé. Fuzzy reinforcement learning. an overview. *Advances in Fuzzy Theory and Technology*, 1995.

[Rup94] Jim Ruppert. A delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 1994.

[Sut96] Richard S. Sutton. Generalization in reinforcement learning : Successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems*, 8, 1996.

[Wat81] D.F. Watson. Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *The Computer Journal*, 24:167–172, 1981.

[Wat89] Christopher J.C.H. Watkins. *Learning from delayed reward.* PhD thesis, Cambridge University, 1989.

This article was processed using the LaTeX macro package with LLNCS style