Active Exploration by Searching for Experiments that Falsify the Computed Control Policy

Raphael Fonteneau*, Susan A. Murphy†, Louis Wehenkel* and Damien Ernst*
* Department of Electrical Engineering and Computer Science, University of Liège, BELGIUM
†Department of Statistics, University of Michigan, USA

***Abstract*—We propose a strategy for experiment selection - in the context of reinforcement learning - based on the idea that the most interesting experiments to carry out at some stage are those that are the most liable to falsify the current hypothesis about the optimal control policy. We cast this idea in a context where a policy learning algorithm and a model identification method are given a priori. Experiments are selected if, using the learnt environment model, they are predicted to yield a revision of the learnt control policy. Algorithms and simulation results are provided for a deterministic system with discrete action space. They show that the proposed approach is promising.**

I. Introduction

Many relevant decision problems in the field of engineering [19], finance [12], medicine ([15], [16]) or artificial intelligence [20] can be formalized as optimal control problems, which are problems where one seeks to compute a control policy so as to maximize a numerical performance criterion.

Often, for solving these problems, one has to deal with an incomplete knowledge of the two key elements of the optimal control problem, which are the system dynamics and the reward function. A vast literature has already proposed ways for computing approximate optimal solutions to these problems when the only information available on these elements is in the form of a set of system transitions, where every system transition is made of a state, the action taken while being in this state, and the values of the reward function and system dynamics observed in this state-action point. In particular, researchers in the field of reinforcement learning (RL) - where the goal was initially to design intelligent agents able to interact with an environment so as to maximize a numerical reward signal - have developed efficient algorithms to address this particular problem, commonly known as batch mode reinforcement learning (BMRL) algorithms.

In this paper, we consider the problem of choosing additional data gathering experiments on the real system in order to complete an already available sample of system trajectories, so as to improve the policy learned by a given BMRL algorithm as much as possible, i.e. by using a minimum number of additional data gathering experiments. Our strategy is based on using a predictive model (PM) of the system performance inferred from the already collected datasets. The PM allows us to predict the outcome of new putative experiments with the real system in terms of putative trajectories, and hence to predict the effect of including these putative trajectories into the sample used by the BMRL algorithm in terms of their impact on the policy inferred by this algorithm. In order to choose the next experiment, we suggest that a good strategy is to select an experiment which (putatively) would lead to a revision of the policy learned from the augmented dataset. In essence, this strategy consists in always trying to find experiments which are likely to falsify the current hypothesis about the optimal control policy.

This approach relies on two intuitions backed by many works/numerical experiments in the field of optimal control. The first intuition is that if when adding a new system transition to the set of existing ones, the BMRL algorithm run on this new set outputs a policy that falsifies the previously computed policy, then this new system transition may be particularly informative. The second intuition is related to the fact that for many problems, one may easily use the already collected information on the dynamics and reward function to build a PM of the system. Based on these two observations, our approach (i) iteratively screens a set of potential sampling locations, i.e. a set of state-action points candidate for sampling, (ii) computes for each one of these points a predicted system transition, and (iii) analyzes the influence that each such predicted transition would have on the policy computed by the BMRL algorithm when combined with the "true" system transitions previously collected. The output of this analysis is then used to (iv) select a sampling location which is "predicted" to generate a new system transition that falsifies the policy computed by the BMRL algorithm.

After detailing this approach and the context in which it is proposed in sections II, III and IV, we report in section V simulation results with the car-on-the-hill problem. Section VI discusses related work and Section VII concludes.

II. Problem Statement

We consider a deterministic time-invariant system whose discrete-time dynamics over T stages is described by

$$x_{t+1} = f(x_t, u_t) \quad t = 0, 1, \ldots, T-1,$$

where for all $t \in \{0, \ldots, T-1\}$, the state x_t is an element of the normed state space $(\mathcal{X}, \|.\|_{\mathcal{X}})$ and u_t is an element of a finite action space $\mathcal{U} = \{a^1, \ldots, a^m\}$ with $m \in \mathbb{N}_0$. $T \in \mathbb{N}_0$ denotes the finite optimization horizon. An instantaneous reward

$$r_t = \rho(x_t, u_t) \in \mathbb{R}$$

is associated with the action $u_t \in \mathcal{U}$ taken while being in state $x_t \in \mathcal{X}$. We assume that the initial state of the

978-1-4244-9886-4/11/$26.00 ©2011 IEEE

system $x_0 \in \mathcal{X}$ is known. For a given sequence of actions $\mathbf{u} = (u_0, \ldots, u_{T-1}) \in \mathcal{U}^T$, we denote by $J^{\mathbf{u}}(x_0)$ the $T-$stage return of the sequence of actions $\mathbf{u}$ when starting from x_0, defined as follows:

$$\forall \mathbf{u} \in \mathcal{U}^T, J^{\mathbf{u}}(x_0) = \sum_{t=0}^{T-1} \rho(x_t, u_t)$$

with $x_{t+1} = f(x_t, u_t), \forall t \in \{0, \ldots, T-1\}$. We denote by $J^*(x_0)$ the maximal value of $J^{\mathbf{u}}(x_0)$ over $\mathcal{U}^T$:

$$J^*(x_0) = \max_{\mathbf{u} \in \mathcal{U}^T} J^{\mathbf{u}}(x_0) \ .$$

An optimal sequence of actions $\mathbf{u}^*$ is a sequence for which

$$J^{\mathbf{u}^*}(x_0) = J^*(x_0) \ .$$

In the following, we call "system transition" a $4-$tuple $(x, u, \rho(x,u), f(x,u)) \in \mathcal{X} \times \mathcal{U} \times \mathbb{R} \times \mathcal{X}$ that gathers information on the functions f and ρ in a point (x, u) of the state-action space $\mathcal{X} \times \mathcal{U}$. Batch mode RL algorithms ([17], [8], [19]) have been introduced to infer near optimal control policies from the sole knowledge of a sample of system transitions

$$\mathcal{F}_n = \left\{\left(x^l, u^l, r^l, y^l\right)\right\}_{l=1}^{n}$$

where $r^l = \rho(x^l, u^l)$ and $y^l = f(x^l, u^l)$. In the rest of this paper, we denote by $BMRL$ a generic batch mode RL algorithm and by $BMRL(\mathcal{F}_n, x_0)$ the policy it computes.

The problem we address is to find a sampling strategy which allows to collect a set of system transitions $\mathcal{F}_n$ from which a high quality sequence of actions $\tilde{\mathbf{u}}^*_{\mathcal{F}_\mathbf{n}} \in \mathcal{U}^T$ can be inferred by $BMRL$, i.e. a sequence of actions $\tilde{\mathbf{u}}^*_{\mathcal{F}_\mathbf{n}} \in \mathcal{U}^T$ such that $J^{\tilde{\mathbf{u}}^*_{\mathcal{F}_\mathbf{n}}}(x_0)$ is as close as possible to $J^*(x_0)$. The sampling process is limited to $N_{\max} \in \mathbb{N}$ transitions, i.e. one can afford to collect at most $N_{\max}$ system transitions.

III. Iterative Sampling Strategy to Collect Informative System Transitions

In this section we describe one way to implement the general falsification strategy presented in Section I for addressing the problem stated in Section II.

Assuming that we are given a batch-mode RL algorithm, $BMRL$, a predictive model PM, and a sequence of numbers $L_n \in \mathbb{N}_0$, we proceed iteratively, by carrying out the following computations at any iteration $n < N_{\max}$:

- Using the sample $\mathcal{F}_n = \left\{\left(x^l, u^l, r^l, y^l\right)\right\}_{l=1}^{n}$ of already collected transitions, we first compute a sequence of actions $\tilde{\mathbf{u}}^*_{\mathcal{F}_\mathbf{n}} = BMRL(\mathcal{F}_n, x_0)$.
- Next, we draw a state-action point $(x, u) \in \mathcal{X} \times \mathcal{U}$ according to a uniform probability distribution $p_{\mathcal{X}\times\mathcal{U}}(\cdot)$ over the state-action space $\mathcal{X} \times \mathcal{U}$.
- Using the sample $\mathcal{F}_n$ and the predictive model PM, we then compute a "predicted" system transition by:

$$(x, u, \hat{r}_{\mathcal{F}_n}(x,u), \hat{y}_{\mathcal{F}_n}(x,u)) = PM(\mathcal{F}_n, x, u) \ .$$

- Using $(x, u, \hat{r}_{\mathcal{F}_n}(x,u), \hat{y}_{\mathcal{F}_n}(x,u))$, we build the "predicted" augmented sample by:

$$\hat{\mathcal{F}}_{n+1}(x,u) = \mathcal{F}_n \cup \{(x, u, \hat{r}_{\mathcal{F}_n}(x,u), \hat{y}_{\mathcal{F}_n}(x,u))\} \ ,$$

and use it to predict the revised policy by:

$$\hat{\mathbf{u}}^*_{\hat{\mathcal{F}}_{\mathbf{n+1}}(\mathbf{x},\mathbf{u})} = BMRL(\hat{\mathcal{F}}_{n+1}(x,u), x_0) \ .$$

  - If $\hat{\mathbf{u}}^*_{\hat{\mathcal{F}}_{\mathbf{n+1}}(\mathbf{x},\mathbf{u})} \neq \tilde{\mathbf{u}}^*_{\mathcal{F}_\mathbf{n}}$, we consider (x, u) as informative, because it is potentially falsifying our current hypothesis about the optimal control policy. We hence use it to make an experiment on the real-system so as to collect a new transition $\left(x^{n+1}, u^{n+1}, r^{n+1}, y^{n+1}\right)$ with $x_{n+1} = x$, $u_{n+1} = u$, $r^{n+1} = \rho(x,u)$ and $y^{n+1} = f(x,u)$, and we augment the sample with it:

$$\mathcal{F}_{n+1} = \mathcal{F}_n \cup \left\{\left(x^{n+1}, u^{n+1}, r^{n+1}, y^{n+1}\right)\right\} \ .$$

  - If $\hat{\mathbf{u}}^*_{\hat{\mathcal{F}}_{\mathbf{n+1}}(\mathbf{x},\mathbf{u})} = \tilde{\mathbf{u}}^*_{\mathcal{F}_\mathbf{n}}$, we draw another state-action point (x', u') according to $p_{\mathcal{X}\times\mathcal{U}}(\cdot)$ and repeat the process of prediction followed by policy revision.
  - If L_n state-action points have been tried without yielding a potential falsifier of the current policy, we give up and merely draw a state-action point $\left(x^{n+1}, u^{n+1}\right)$ "at random" according to $p_{\mathcal{X}\times\mathcal{U}}(\cdot)$, and augment $\mathcal{F}_n$ with the transition

$$\left(x^{n+1}, u^{n+1}, \rho(x^{n+1}, u^{n+1}), f(x^{n+1}, u^{n+1})\right) .$$

Influence of the $BMRL$ algorithm and the predictive model PM. For this iterative sampling strategy to behave well, the inference capabilities of the $BMRL$ algorithm it uses should obviously be as good as possible. Usually, BMRL algorithms rely on the training of function approximators [4] that either represent the system dynamics and the reward function of the underlying control problem, a (state-action) value function or a policy. Given the fact that here, at any iteration of the algorithm, the only knowledge on the problem is given in the form of a sample of system transitions, we advocate using BMRL algorithms with non-parametric function approximators such as, for example, nearest neighbor or tree-based methods.

The best predictive model PM would be an algorithm that would, given a state action pair (x, u), output a predicted transition equal to $(x, u, \rho(x,u), f(x,u))$. Since predicting with great accuracy $\rho(x,u)$ and $f(x,u)$ may be difficult, one could also imagine an algorithm that computes a set of predictions rather than a single "best guess". Indeed, with such a choice, it would be more likely that at least one of these predicted transitions would also lead to a predicted policy falsification if the exact one leads to a true policy falsification. However, working with a large predicted set may also increase the likelihood that a sampling location would be predicted as a policy falsifier while it is actually not the case. Notice also that if some prior knowledge on the problem is available, it may be possible to exploit it to define for a given sampling location, a set of transitions which is "compatible" with the previous samples collected (see, e.g., [10] where a compatible set is defined when assuming that

the problem is Lipschitz continuous with known Lipschitz constants). This could be used to increase the performance of a prediction algorithm by avoiding incompatible predictions.

Influence of the L_n sequence of parameters. L_n sets the maximal number of trials for searching a new experiment when n transitions have already been collected. Its value should be chosen large enough so as to ensure that, if there exist transitions that indeed lead to a policy falsification, one of those would be identified with high probability. It may however happen that, at some iteration n, there doesn't exist any (predicted) transition that would lead to a (predicted) policy falsification. In this case, our algorithm will conduct L_n trials, which may be problematic from the computational point of view if L_n is very large. Thus the choice of L_n is a trade-off between the desirability to have at any iteration a high probability to find a sample that leads to a policy falsification, and the need to avoid excessive computations when such a sampling location does not exist.

IV. $BMRL/PM$ Implementation Based on Nearest-neighbour Approximations

In this section, we present the batch-mode RL algorithm $BMRL$ and the predictive model PM to which our iterative sampling strategy will be applied in the context of simulations reported in Section V.

As $BMRL$ algorithm, we have chosen a model learning–type RL algorithm. It first approximates the functions f and ρ from the available sample of system transitions, and then solves "exactly" the optimal control problem defined by these approximations. This algorithm is fully detailed in Section IV-A. In Section IV-B, we present the PM used in our experiments. It computes its predictions based on the same approximations as those used by the $BMRL$ algorithm.

A. Choice of the Inference Algorithm $BMRL$

Model learning–type RL. Model learning–type RL aims at solving optimal control problems by approximating the unknown functions f and ρ and solving the so approximated optimal control problem instead of the unknown actual optimal control problem. The values y^l (resp. r^l) of the function f (resp. ρ) in the state-action points $(x^l, u^l) \quad l = 1 \dots n$ are used to learn a function $\tilde{f}_{\mathcal{F}_n}$ (resp. $\tilde{\rho}_{\mathcal{F}_n}$) over the whole space $\mathcal{X} \times \mathcal{U}$. The approximated optimal control problem defined by the functions $\tilde{f}_{\mathcal{F}_n}$ and $\tilde{\rho}_{\mathcal{F}_n}$ is solved and its solution is kept as an approximation of the solution of the optimal control problem defined by the actual functions f and ρ.

Given a sequence of actions $\mathbf{u} \in \mathcal{U}^T$ and a model learning–type RL algorithm, we denote by $\tilde{J}^{\mathbf{u}}_{\mathcal{F}_n}(x_0)$ the approximated $T-$stage return of the sequence of actions $\mathbf{u}$, i.e. the $T-$stage return when considering the approximations $\tilde{f}_{\mathcal{F}_n}$ and $\tilde{\rho}_{\mathcal{F}_n}$:

$$\forall \mathbf{u} \in \mathcal{U}^T, \tilde{J}^{\mathbf{u}}_{\mathcal{F}_n}(x_0) = \sum_{t=0}^{T-1} \tilde{\rho}_{\mathcal{F}_n}\left(\tilde{x}_t, u_t\right)$$

with $\tilde{x}_{t+1} = \tilde{f}_{\mathcal{F}_n}\left(\tilde{x}_t, u_t\right),\ \forall t \in \{0, \dots, T-1\}$ and $\tilde{x}_0 = x_0$. We denote by $\tilde{J}^*_{\mathcal{F}_n}(x_0)$ the maximal approximated $T-$stage return when starting from the initial state $x_0 \in \mathcal{X}$ according to the approximations $\tilde{f}_{\mathcal{F}_n}$ and $\tilde{\rho}_{\mathcal{F}_n}$:

$$\tilde{J}^*_{\mathcal{F}_n}(x_0) = \max_{\mathbf{u} \in \mathcal{U}^T} \tilde{J}^{\mathbf{u}}_{\mathcal{F}_n}(x_0)\ .$$

Using these notations, model learning–type RL algorithms aim at computing a sequence of actions $\tilde{\mathbf{u}}^*_{\mathcal{F}_\mathbf{n}} \in \mathcal{U}^T$ such that $\tilde{J}^{\tilde{\mathbf{u}}^*_{\mathcal{F}_\mathbf{n}}}_{\mathcal{F}_n}(x_0)$ is as close as possible (and ideally equal to) to $\tilde{J}^*_{\mathcal{F}_n}(x_0)$. These techniques implicitly assume that an optimal policy for the learned model leads also to high returns on the real problem.

Voronoi tessellation based RL algorithm. We describe here the model-learning type of RL algorithm that will be used later in our simulations. This algorithm approximates the reward function ρ and the system dynamics f using piecewise constant approximations on a Voronoi–like [2] partition of the state-action space (which is equivalent to a nearest-neighbour approximation) and will be referred to by the VRL algorithm. Given an initial state $x_0 \in \mathcal{X}$, the VRL algorithm computes an open-loop sequence of actions which corresponds to an "optimal navigation" among the Voronoi cells.

Before fully describing this algorithm, we first assume that all the state-action pairs $\left\{(x^l, u^l)\right\}_{l=1}^n$ given by the sample of transitions $\mathcal{F}_n$ are unique, i.e.

$$\forall l, l' \in \{1, \dots, n\}, (x^l, u^l) = (x^{l'}, u^{l'}) \implies l = l'\ .$$

We also assume that each action of the action space $\mathcal{U}$ has been tried at least once, i.e.,

$$\forall u \in \mathcal{U}, \exists l \in \{1, \dots, n\}, u^l = u\ .$$

The model is based on the creation of n Voronoi cells $\left\{V^l\right\}_{l=1}^n$ which define a partition of size n of the state-action space. The Voronoi cell V^l associated to the element (x^l, u^l) of $\mathcal{F}_n$ is defined as the set of state-action pairs $(x, u) \in \mathcal{X} \times \mathcal{U}$ that satisfy:

$$(i) \quad u = u^l \quad , \tag{1}$$

$$(ii) \quad l \in \arg\min_{l': u^{l'} = u} \left\{ \|x - x^{l'}\|_{\mathcal{X}} \right\} \ , \tag{2}$$

$$(iii) \quad l = \min_{l'} \left\{ l' \in \arg\min_{l': u^{l'} = u} \left\{ \|x - x^{l'}\|_{\mathcal{X}} \right\} \right\}. \tag{3}$$

One can verify that $\left\{V^l\right\}_{l=1}^n$ is indeed a partition of the state-action space $\mathcal{X} \times \mathcal{U}$ since every state-action $(x, u) \in \mathcal{X} \times \mathcal{U}$ belongs to one and only one Voronoi cell.

The function f (resp. ρ) is approximated by a piecewise constant function $\tilde{f}_{\mathcal{F}_n}$ (resp. $\tilde{\rho}_{\mathcal{F}_n}$) defined as follows:

$$\begin{aligned} \forall l \in \{1, \dots, n\}, \forall (x, u) \in V^l, \quad \tilde{f}_{\mathcal{F}_n}(x, u) &= y^l, \\ \tilde{\rho}_{\mathcal{F}_n}(x, u) &= r^l\ . \end{aligned}$$

Using the approximations $\tilde{f}_{\mathcal{F}_n}$ and $\tilde{\rho}_{\mathcal{F}_n}$, we define a sequence of approximated optimal state-action value functions

$\left(\tilde{Q}^*_{T-t}\right)_{t=0}^{T-1}$ as follows : $\forall t \in \{0,\ldots,T-1\}, \forall (x,u) \in \mathcal{X}\times\mathcal{U}$,

$$\begin{aligned}\tilde{Q}^*_{T-t}(x,u) &= \tilde{\rho}_{\mathcal{F}_n}(x,u) \\ &+ \arg\max_{u'\in\mathcal{U}} \tilde{Q}^*_{T-t-1}\left(\tilde{f}_{\mathcal{F}_n}(x,u),u'\right) ,\end{aligned}$$

with $Q^*_1(x,u) = \tilde{\rho}_{\mathcal{F}_n}(x,u), \quad \forall (x,u) \in \mathcal{X}\times\mathcal{U}$.

Using the sequence of approximated optimal state-action value functions $\left(\tilde{Q}^*_{T-t}\right)_{t=0}^{T-1}$, one can infer an open-loop sequence of actions $\tilde{\mathbf{u}}^*_{\mathcal{F}_\mathbf{n}} = (\tilde{u}^*_{\mathcal{F}_n,0},\ldots,\tilde{u}^*_{\mathcal{F}_n,T-1}) \in \mathcal{U}^T$ which is an exact solution of the approximated optimal control problem, i.e. which is such that $\tilde{J}^{\tilde{\mathbf{u}}^*_{\mathcal{F}_\mathbf{n}}}_{\mathcal{F}_n}(x_0) = \tilde{J}^*_{\mathcal{F}_n}(x_0)$ as follows:

$$\tilde{u}^*_{\mathcal{F}_n,0} \in \arg\max_{u'\in\mathcal{U}} \tilde{Q}^*_T(\tilde{x}^*_0,u') ,$$

and, $\forall t \in \{0,\ldots,T-2\}$,

$$\tilde{u}^*_{\mathcal{F}_n,t+1} \in \arg\max_{u'\in\mathcal{U}} \tilde{Q}^*_{T-(t+1)}\left(\tilde{f}_{\mathcal{F}_n}\left(\tilde{x}^*_t,\tilde{u}^*_{\mathcal{F}_n,t}\right),u'\right)$$

where $\tilde{x}^*_0 = x_0$ and $\tilde{x}^*_{t+1} = \tilde{f}_{\mathcal{F}_n}(\tilde{x}^*_t,\tilde{u}^*_t), \forall t \in \{0,\ldots,T-1\}$.

All the approximated optimal state-action value functions $\left(\tilde{Q}^*_{T-t}\right)_{t=0}^{T-1}$ are piecewise constant over each Voronoi cell, a property that can be exploited for computing them easily as it is shown in Figure 1. The VRL algorithm has linear complexity with respect to the cardinality n of the sample of system transitions $\mathcal{F}_n$, the optimization horizon T and the cardinality m of the action space $\mathcal{U}$. Furthermore, the VRL algorithm has consistency properties in Lipschitz continuous environments, for which the open-loop sequence of actions computed by the VRL algorithm converges towards an optimal sequence of actions when the sparsity of the sample of system transitions converges towards zero [9].

B. Choice of the Predictive Model PM

Model learning–type RL uses a predictive model of the environment. Our predictive model PM is thus given by the approximated system dynamics $\tilde{f}_{\mathcal{F}_n}$ and reward function $\tilde{\rho}_{\mathcal{F}_n}$ computed by the VRL algorithm. Given a sample of transitions $\mathcal{F}_n$ and a state-action point $(x,u) \in \mathcal{X}\times\mathcal{U}$, the PM algorithm computes a predicted system transition $(x,u,\hat{r}_{\mathcal{F}_n}(x,u),\hat{y}_{\mathcal{F}_n}(x,u)) = PM(\mathcal{F}_n,x,u)$ such that:

$$\begin{aligned}\forall (x,u) \in \mathcal{X}\times\mathcal{U}: \qquad & \hat{r}_{\mathcal{F}_n}(x,u) = \tilde{\rho}_{\mathcal{F}_n}(x,u) , \\ & \hat{y}_{\mathcal{F}_n}(x,u) = \tilde{f}_{\mathcal{F}_n}(x,u) .\end{aligned}$$

V. Experimental Simulation Results with the Car-on-the-hill Problem

We propose in this section to illustrate the sampling strategy proposed in the previous sections on the car-on-the-hill problem [7] which has been vastly used as benchmark for validating RL algorithms. First we describe the benchmark. Afterwards we detail the experimental protocol and finally, we present and discuss our simulation results.

Inputs: an initial state $x_0 \in \mathcal{X}$, a sample of transitions $\mathcal{F}_n = \left\{\left(x^l,u^l,r^l,y^l\right)\right\}_{l=1}^n$;
Output: a sequence of actions $\tilde{\mathbf{u}}^*_{\mathcal{F}_\mathbf{n}}$ and $\tilde{J}^*_{\mathcal{F}_n}(x_0)$;
Initialization:
Create a $n\times m$ matrix V such that $V(i,j)$ contains the index of the Voronoi cell (VC) where $\left(\tilde{f}_{\mathcal{F}_n}(x^i,u^i),a^j\right)$ lies ;
for $i = 1$ **to** n **do**
  $Q_{1,i} \leftarrow r^i$;
end for
Algorithm:
for $t = T-2$ **to** 0 **do**
  for $i = 1$ **to** n **do**
    $l \leftarrow \arg\max_{l'\in\{1,\ldots,m\}} \left\{Q_{T-t-1,V(i,l')}\right\}$;
    $Q_{T-t,i} \leftarrow r^i + Q_{T-t-1,V(i,l)}$;
  end for
end for
$l \leftarrow \arg\max_{l'\in\{1,\ldots,m\}} Q_{T,i'}$ where i' denotes the index of the VC where $(x_0,a^{l'})$ lies ;
$l^*_0 \leftarrow$ index of the VC where (x_0,a^l) lies ;
$\tilde{J}^*_{\mathcal{F}_n}(x_0) \leftarrow Q_{T,l^*_0}$;
$i \leftarrow l^*_0$;
$\tilde{u}^*_{\mathcal{F}_n,0} \leftarrow u^{l^*_0}$;
for $t = 0$ **to** $T-2$ **do**
  $l^*_{t+1} \leftarrow \arg\max_{l'\in\{1,\ldots,m\}} \left\{Q_{T-t-1,V(i,l')}\right\}$;
  $\tilde{u}^*_{\mathcal{F}_n,t+1} \leftarrow a^{l^*_{t+1}}$;
  $i \leftarrow V(i,l^*_{t+1})$;
end for
Return: $\tilde{\mathbf{u}}^*_{\mathcal{F}_\mathbf{n}} = (\tilde{u}^*_{\mathcal{F}_n,0},\ldots,\tilde{u}^*_{\mathcal{F}_n,T-1})$ and $\tilde{J}^*_{\mathcal{F}_n}(x_0)$.

Fig. 1. The Voronoi Reinforcement Learning (VRL) algorithm. $Q_{T-t,l}$ is the value taken by the function $\tilde{Q}^*_{T-t}$ in the Voronoi cell V^l.

A. The Car-on-the-hill Benchmark

In the car-on-the-hill benchmark, a point mass - which represents a car - has to be driven past the top of a hill by applying a horizontal force. For some initial states, the maximum available force is not sufficient to drive the car directly up the right hill. Instead, the car has to first be driven up the opposite (left) slope in order to gather energy prior to accelerating towards the goal. An illustration of the car-on-the-hill benchmark is given below in Figure 2.

The continuous-time dynamics of the car is given by

$$\ddot{z} = \frac{1}{1+\left(\frac{dH(z)}{dz}\right)^2}\left(\frac{u}{m_c} - g\frac{dH(z)}{dz} - \dot{z}^2\frac{dH(z)}{dz}\frac{d^2H(z)}{dz^2}\right)$$

where $z \in [-1,1]$ is the horizontal position of the car (expressed in m), $\dot{z} \in [-3,3]$ is the velocity of the car (given in m/s), $u \in \{-4,4\}$ is the horizontal force applied to the car (expressed in N), $g = 9.81 m/s^2$ is the gravitational acceleration and H denotes the slope of the hill:

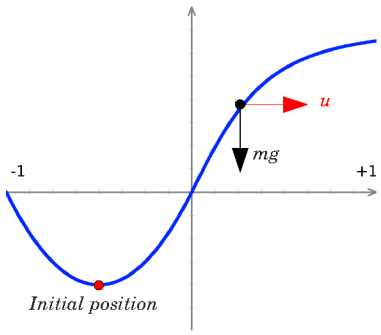

Fig. 2. Illustration of the car-on-the-hill benchmark.

$$H(z) = \begin{cases} z^2 + z & \text{if} \quad z < 0 \ , \\ \frac{z}{\sqrt{1+5z^2}} & \text{if} \quad z \geq 0 \ . \end{cases}$$

We assume that the car has a mass $m_c = 1kg$. The discrete time step is set to $Ts = 0.1s$ and the discrete time dynamics f is obtained by integrating the continuous-time dynamics between subsequent time steps. The action space $\mathcal{U}$ is made of the two elements: -4 and 4. Whenever the position z or velocity $\dot{z}$ exceeds the bounds, the car reaches an absorbing state in which it stays whatever the control actions taken. If $z_{t+1} < -1$ or if $|\dot{z}_{t+1}| > 3$, then the car reaches a "loosing" absorbing state s_{-1} and gets a -1 reward at each time-step till the end of the trial. If $z_{t+1} \geq 1$ and $|\dot{z}_{t+1}| \leq 3$, then the car reaches a "winning" absorbing state s_1, and gets a $+1$ reward at each time-step till the end of the trial. We have assumed in our simulation that we know that s_{-1} and s_{+1} are two absorbing states. The state space of the system is equal to $\mathcal{X} = [-1, 1] \times [-3, 3] \cup \{s_1, s_{-1}\}$.

The goal is to find a sequence of actions that leads to the highest sum of rewards over an optimization horizon $T = 20$ when the car starts in $x_0 = [-0.5, 0]$. Such a sequence of actions also drives the car in a minimum amount of time to the top of the hill.

The VRL algorithm was described in Section IV-A for problems with no absorbing states. It can easily be amended to handle the absorbing states of the car-on-the-hill problem. This can be done for example by modifying the set of system transitions used as input of the algorithm by adding $m \times n_{abs}$ "fake system transitions", where n_{abs} is the number of absorbing states of the problem. With respect to the car-on-the-hill problem, this results in the addition of the following four fake system transitions into any sample of transitions $\{(s_1, 4, 1, s_1), (s_1, -4, 1, s_1), (s_{-1}, 4, -1, s_{-1}), (s_{-1}, -4, -1, s_{-1})\}$ The definition of the Voronoi cells remains the same as in Equations (1), (2) and (3) if x^l is not an absorbing state. Otherwise, the norm $\|.\|_{\mathcal{X}}$ can be (abusively) "extended" to absorbing states as follows:

$$\|x - x^l\|_{\mathcal{X}} = \begin{cases} 0 & \text{if } x = x^l \ , \\ +\infty & \text{if } x \neq x^l \ . \end{cases}$$

B. Experimental Protocol

We propose to compare the performance of our sampling strategy described in Section III with the performance of a uniform sampling strategy. To this end, we run $q = 50$ times our sampling strategy, where each run $k = 1 \ldots q$ is initialized with a sample $\mathcal{F}_m^k$ that contains $m = 2$ system transitions (one transition for each action of the action space) as follows:

$$\forall k \in \{1, \ldots, q\}, \mathcal{F}_m^k = \{(x_0, -4, \rho(x_0, -4), f(x_0, -4)) \ , \\ (x_0, +4, \rho(x_0, +4), f(x_0, +4))\} \ .$$

We sequentially run our sampling strategy on each sample of transitions $\mathcal{F}_m^k \quad k = 1 \ldots q$ until it gathers $N_{\max} = 1000$ system transitions. These runs lead to q sequences of $(N_{\max} - m + 1)$ samples of system transitions: $\mathcal{F}_m^1, \mathcal{F}_{m+1}^1, \ldots, \mathcal{F}_{N_{\max}}^1, \ldots, \mathcal{F}_m^q, \mathcal{F}_{m+1}^q, \ldots, \mathcal{F}_{N_{\max}}^q$. We also generate q sequences of $(N_{\max} - m + 1)$ samples of system transitions $\mathcal{G}_m^1, \mathcal{G}_{m+1}^1, \ldots, \mathcal{G}_{N_{\max}}^1, \ldots, \mathcal{G}_m^q, \mathcal{G}_{m+1}^q, \ldots, \mathcal{G}_{N_{\max}}^q$ where, for all $k = 1 \ldots q$, for all $n = m \ldots N_{\max} - 1$, each sample $\mathcal{G}_{n+1}^k$ is obtained by adding one system transition $(x, u, \rho(x, u), f(x, u))$ to $\mathcal{G}_n^k$ for which (x, u) is drawn according to $p_{\mathcal{X}\times\mathcal{U}}(\cdot)$. The sequence of parameters L_n used for these experiments is defined as follows:

$$\forall n \in \{m, \ldots, N_{\max}\}, L_n = mn \ .$$

The probability distribution $p_{\mathcal{X}\times\mathcal{U}}(\cdot)$ is such that the probability of drawing a state-action point (x, u) with $x = s_1$ or $x = s_{-1}$ is zero, and uniform elsewhere.

C. Results and Discussions

Performances of the control policies inferred from the samples of $N_{\max}$ transitions. We first compute the returns of the $2q$ control policies respectively inferred by the VRL algorithm from the samples of $N_{\max}$ system transitions $\mathcal{F}_{N_{\max}}^k$ and with the randomly generated samples $\mathcal{G}_{N_{\max}}^k \quad k = 1 \ldots q$. The obtained results are reported on Figure 3 in terms of the distribution of returns of the inferred control policy over the 50 runs.

We observe that the VRL algorithm manages to compute for 28% of the runs a control policy for which the return is equal to 2, whereas not even a single control policy with a return greater than 0 was inferred from the q samples $\mathcal{G}_{N_{\max}}^k \quad k = 1 \ldots q$ generated using the uniform sampling strategy.

Notice that in order to obtain results of similar quality to those of our iterative sampling strategy, we found that one would need to use about $10,000$ randomly generated system transitions.

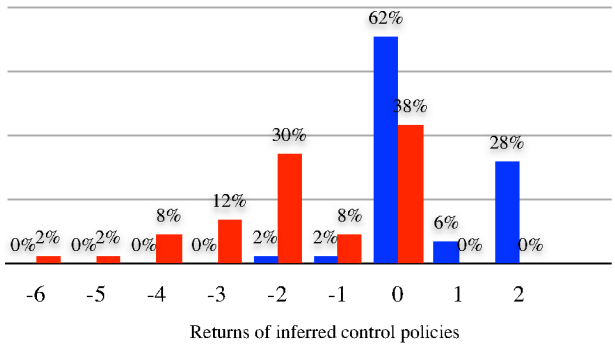

Fig. 3. Distribution of the returns of the control policies inferred from $\mathcal{F}^k_{N_{\max}} \quad k = 1 \ldots q$ (blue histogram, on the left) and $\mathcal{G}^k_{N_{\max}} \quad k = 1 \ldots q$ (red histogram, on the right).

Average performance and distribution of the returns of the inferred control policies. For a given cardinality n ($m \leq n \leq N_{\max}$), we compute the average actual performance $\mathcal{M}(n)$ of the q sequences of actions $\tilde{\mathbf{u}}^*_{\mathcal{F}^{\mathbf{k}}_{\mathbf{n}}} \quad k = 1 \ldots q$ computed by the VRL algorithm from the sample of system transitions $\mathcal{F}^k_n \quad k = 1 \ldots q$:

$$\mathcal{M}(n) = \frac{1}{q} \sum_{k=1}^{q} J^{\tilde{\mathbf{u}}^*_{\mathcal{F}^{\mathbf{k}}_{\mathbf{n}}}}(x_0) \ .$$

The average performance $\mathcal{M}(n) \quad n = m \ldots N_{\max}$ is compared with the average performance $\mathcal{M}_{unif}(n)$ of the q sequences of actions $\tilde{\mathbf{u}}^*_{\mathcal{G}^{\mathbf{k}}_{\mathbf{n}}} \quad k = 1 \ldots q$ inferred by the VRL algorithm from samples of system transitions $\mathcal{G}^k_n \quad k = 1 \ldots q$ gathered according to a uniform sampling strategy:

$$\mathcal{M}_{unif}(n) = \frac{1}{q} \sum_{k=1}^{q} J^{\tilde{\mathbf{u}}^*_{\mathcal{G}^{\mathbf{k}}_{\mathbf{n}}}}(x_0) \ .$$

The values of $\mathcal{M}(n)$ and $\mathcal{M}_{unif}(n)$ for $n = m \ldots N_{\max}$ are reported on Figure 4. We also report the distribution of the return of the policies $\tilde{\mathbf{u}}^*_{\mathcal{F}^{\mathbf{k}}_{\mathbf{n}}} \quad k = 1 \ldots q, n = m \ldots N_{\max}$ (resp. $\tilde{\mathbf{u}}^*_{\mathcal{G}^{\mathbf{k}}_{\mathbf{n}}} \quad k = 1 \ldots q, n = m \ldots N_{\max}$) in Figure 5 (resp. Figure 6). We observe that, with our sampling strategy, control policies leading to a return of 2 can be inferred from samples of less than 200 system transitions. We also notice that no policy leading to a return of 2 could be inferred from any of the uniformly sampled system transitions $\mathcal{G}^k_n \quad k = 1 \ldots q$.

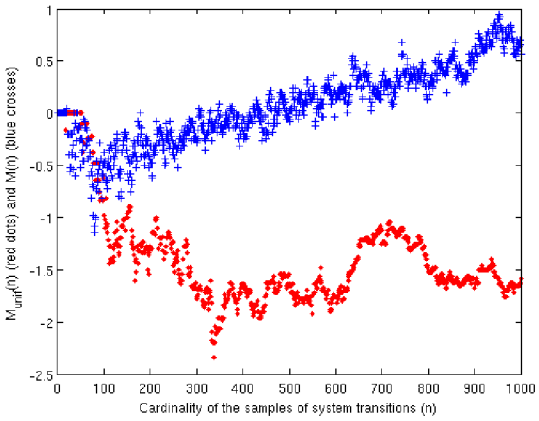

Fig. 4. Evolution of the average performance of our sampling strategy $\mathcal{M}(n)$ (blue crosses) compared with the average performance of the uniform sampling strategy $M_{unif}(n)$ (red dots).

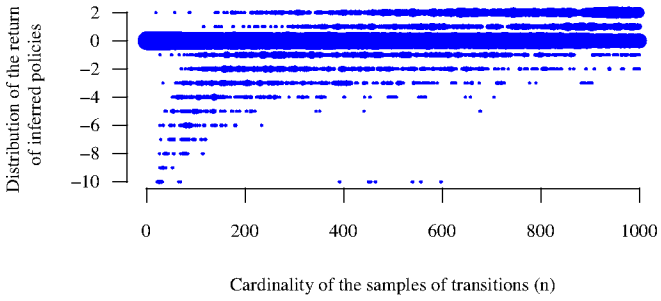

Fig. 5. Distribution of the return of the control policies $\tilde{\mathbf{u}}^*_{\mathcal{F}^{\mathbf{k}}_{\mathbf{n}}}$ k=1 ...q, $n = m \ldots N_{\max}$. For each value of n, the area covered by a bullet to which corresponds a return $r = -10 \ldots 2$ is proportional to the number of control policies from $\left\{\tilde{\mathbf{u}}^*_{\mathcal{F}^{\mathbf{k}}_{\mathbf{n}}}\right\}_{k=1}^{q}$ whose return is equal to r.

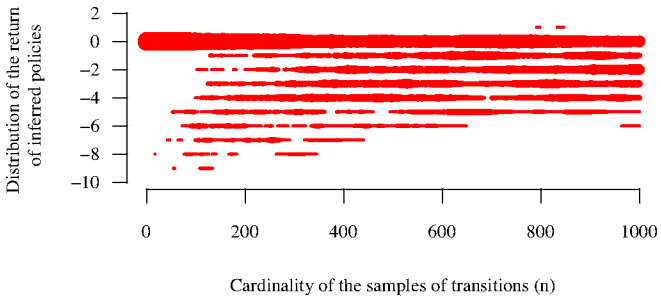

Fig. 6. Distribution of the return of the control policies $\tilde{\mathbf{u}}^*_{\mathcal{G}^{\mathbf{k}}_{\mathbf{n}}}$ k=1 ...q, $n = m \ldots N_{\max}$. For each value of n, the area covered by a bullet to which corresponds a return $r = -10 \ldots 2$ is proportional to the number of control policies from $\left\{\tilde{\mathbf{u}}^*_{\mathcal{G}^{\mathbf{k}}_{\mathbf{n}}}\right\}_{k=1}^{q}$ whose return is equal to r.

Representation of $\mathcal{F}^1_{N_{\max}}$ and $\mathcal{G}^1_{N_{\max}}$. We finally plot the system transitions gathered in the sample $\mathcal{F}^1_{N_{\max}}$ (resp. $\mathcal{G}^1_{N_{\max}}$) on Figure 7 (resp. on Figure 8). Each system transition (x^l, u^l, r^l, y^l) is represented by a colored symbol located at $x^l = [z, \dot{z}]$. A '+' sign indicates that $u^l = +4$, whereas a '•' sign indicates that $u^l = -4$. The symbol is colored in blue if $r^l = 0$. Larger symbols colored in black (green) are used if $r^l = -1$ ($r^l = 1$). The red curve represents the trajectory of the car when driven according to the inferred policy $\tilde{\mathbf{u}}^*_{\mathcal{F}^{\mathbf{1}}_{\mathbf{n}}}$ (resp. $\tilde{\mathbf{u}}^*_{\mathcal{G}^{\mathbf{1}}_{\mathbf{n}}}$). One can observe on Figure 7 that our sampling strategy tends to sample state-action points that are located in the neighbourhood of high-performance trajectories.

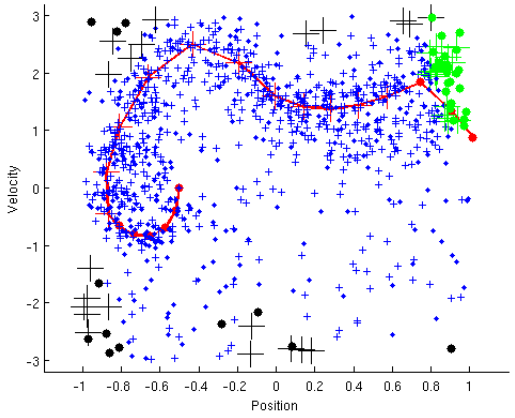

Fig. 7. Representation of the sample of system transitions $\mathcal{F}^1_{N_{\max}}$ (obtained through inferred policy variations-based sampling strategy).

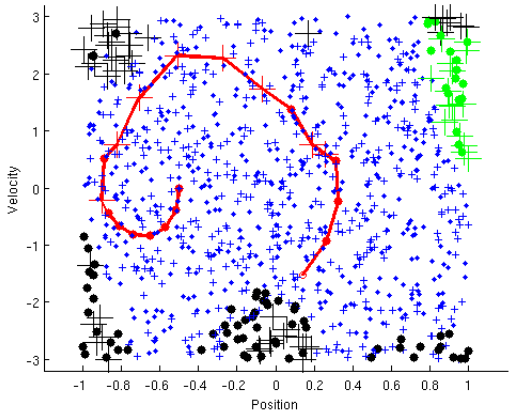

Fig. 8. Representation of the sample of system transitions $\mathcal{G}^1_{N_{\max}}$ (obtained through uniform sampling strategy).

VI. Related Work

The problem of sampling parsimoniously the state-action space of an optimal control problem for identifying good policies has already been addressed by several authors. The approach detailed in [6] is probably the closest to ours. In this paper, the authors propose a sequential sampling strategy which also favors sampling locations that are predicted to have a high-influence on the policy that will be inferred. While we focus in this paper on deterministic problems with continuous state spaces, their approach is particularized to stationary stochastic problems with finite state spaces.

In [11], another sequential sampling strategy is proposed. It works by computing bounds on the return of control policies and selects as sampling area the one which is expected to lead to the highest increase of the bounds' tightness. The approach requires the system dynamics and the reward function to be Lipschitz continuous, and relies at its heart on the resolution of a complex optimization problem.

Most of the works in the field of RL related to the generation of informative samples have focused on the problem of controlling a system so as to generate samples that can be used to increase the performance of the control policy while at the same time generating high rewards. One common approach for addressing this "exploration-exploitation" dilemma ([1], [5]) is to use a so-called ϵ-Greedy policy which is a policy that deviates with a certain probability from the estimate of the optimal one ([21], [13], [20]). The problem has been recently well-studied for stochastic Markov Decision Processes having one single state ([3]).

There is a considerable body of work in the field of adaptive discretization techniques in dynamic programming which is also related to our approach. In these works, the state-action space is iteratively sampled so as to lead rapidly to an optimal policy (see e.g., [14]). If at the inner loop of our approach, exact samples rather than predicted samples were used, it could certainly be assimilated to this body of work. The amount of computation required by our approach to identify at every iteration a new sample would however not make it necessarily a good adaptive discretization technique. Indeed, the efficiency of an adaptive discretization technique does not depend solely only on the number of samples it uses to identify a good policy, but well on its overall computational complexity.

Finally, it is worth mentioning that the problem of identifying a concise set of samples from which a good policy can be inferred has also been addressed in other contexts than the one considered in this paper. For example, [7] proposes a strategy for extracting from a given sample of system transitions, a much smaller subset that can still lead to a good policy. The strategy relies on the computation of errors in a Bellman equation and showed good results on problems having a smooth environment. In [18], the authors focus on the identification of a small sample of transitions that can lead to a good policy when combined with a BMRL algorithm without assuming any constraints on the number of samples that can be generated. The simulation results given in this paper show that for the car-on-the-hill benchmark, less than twenty well chosen samples can lead to an optimal policy. However, for identifying these samples, the state-action space had to be sampled a very large number of times (about hundreds of thousands of times).

VII. Conclusions

We have proposed a sequential strategy for sampling informative collections of system transitions for solving deterministic optimal control problems in continuous state spaces. This sampling strategy uses the ability of predicting system transitions, in order to identify experiments whose outcome would be likely to falsify the current hypothesis about the solution of the optimal control problem. Algorithms have been fully specified for the case of finite horizon deterministic optimal control problems with finite action spaces, by using nearest-neighbor approximations of the optimal control problem both in the RL algorithm and for predicting the outcome of experiments in terms of hypothetical system transitions.

The simulations were carried out on the car-on-the-hill problem and the results were promising. In particular, our

sampling strategy was found to be much more efficient than a uniform sampling one. These results motivate further study of the algorithms proposed in this paper. In particular, it would be interesting to establish under which conditions policy falsification caused by new samples also corresponds to actual policy improvements and what may be the influence of the prediction errors done when generating the "predicted system transitions" on the "predicted policy changes". This should be very helpful for analytically investigating the convergence speed of the proposed sampling strategy towards a sample of system transitions from which optimal or near-optimal policies could be inferred.

Finally, while an instance of this policy falsification concept for generating new experiments has been fully specified and validated for deterministic problems with discrete action spaces, we believe that it would also be interesting to investigate ways to exploit it successfully in other settings.

ACKNOWLEDGEMENTS

Raphael Fonteneau acknowledges the financial support of the FRIA. Damien Ernst is a research associate of the FRS-FNRS. This paper presents research results of the European excellence network PASCAL2 and of the Belgian Network BIOMAGNET, funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. We also acknowledge financial support from NIH grants P50 DA10075 and R01 MH080015. The authors also thank Bertrand Cornélusse for valuable discussions and help with the numerical experiments. The scientific responsibility rests with its authors.

REFERENCES

[1] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Reserch*, 3:397 – 422, 2003.

[2] F. Aurenhammer. Voronoi diagrams — a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[3] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. Online optimization in X-armed bandits. In *Advances in Neural Information Processing Systems 21*, pages 201–208. MIT Press, 2009.

[4] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming using Function Approximators*. Taylor & Francis CRC Press, 2010.

[5] J.D. Cohen, S.M. McClure, and A.J. Yu. Should I stay or should I go? How the human brain manages the trade-off between exploitation and exploration. *Philosophical Transactions of the Royal Society B 29*, 362(1481):933–942, 2007.

[6] A. Ephsteyn, A. Vogel, and G. DeJong. Active reinforcement learning. In *Proceedings of the 25th international conference on Machine learning (ICML 2008)*, volume 307, 2008.

[7] D. Ernst. Selecting concise sets of samples for a reinforcement learning agent. In *Proceedings of the Third International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2005)*, Singapore, 2005.

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

[9] R. Fonteneau and D. Ernst. Voronoi model learning for batch mode reinforcement learning. Technical report, University of Liège, 2010.

[10] R. Fonteneau, S. A. Murphy, L. Wehenkel, and D. Ernst. Towards min max generalization in reinforcement learning. *To be published as book chapter in the series Communications in Computer and Information Science (CCIS) by Springer-Verlag*, 2010.

[11] R. Fonteneau, S.A. Murphy, L. Wehenkel, and D. Ernst. Generating informative trajectories by using bounds on the return of control policies. In *Proceedings of the Workshop on Active Learning and Experimental Design 2010 (in conjunction with AISTATS 2010)*, 2010.

[12] J.E. Ingersoll. *Theory of Financial Decision Making*. Rowman and Littlefield Publishers, Inc., 1987.

[13] L.P. Kaelbling. *Learning in Embedded Systems*. MIT Press, 1993.

[14] R. Munos and A. Moore. Variable resolution discretization in optimal control. *Machine Learning*, 49:291–323, 2002.

[15] S.A. Murphy. Optimal dynamic treatment regimes. *Journal of the Royal Statistical Society, Series B*, 65(2):331–366, 2003.

[16] S.A. Murphy. An experimental design for the development of adaptive treatment strategies. *Statistics in Medicine*, 24:1455–1481, 2005.

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

[18] E. Rachelson, F. Schnitzler, L. Wehenkel, and D. Ernst. Optimal sample selection for batch-mode reinforcement learning. In *3rd International Conference on Agents and Artificial Intelligence (ICAART)*, 2011.

[19] M. Riedmiller. Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method. In *Proceedings of the Sixteenth European Conference on Machine Learning (ECML 2005)*, pages 317–328, Porto, Portugal, 2005.

[20] R.S. Sutton and A.G. Barto. *Reinforcement Learning*. MIT Press, 1998.

[21] S. Thrun. The role of exploration in learning control. In D. White and D. Sofge, editors, *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold, 1992.