Published in final edited form as:
Ann Oper Res. 2013 September 1; 208(1): 383–416. doi:10.1007/s10479-012-1248-5.

Batch Mode Reinforcement Learning based on the Synthesis of Artificial Trajectories

Raphael Fonteneau,
University of Liége, Belgium

Susan A. Murphy,
University of Michigan, USA

Louis Wehenkel, and
University of Liége, Belgium

Damien Ernst
University of Liége, Belgium

Raphael Fonteneau: raphael.fonteneau@ulg.ac.be; Susan A. Murphy: samurphy@umich.edu; Louis Wehenkel: l.wehenkel@ulg.ac.be; Damien Ernst: dernst@ulg.ac.be

Abstract

In this paper, we consider the batch mode reinforcement learning setting, where the central problem is to learn from a sample of trajectories a policy that satisfies or optimizes a performance criterion. We focus on the continuous state space case for which usual resolution schemes rely on function approximators either to represent the underlying control problem or to represent its value function. As an alternative to the use of function approximators, we rely on the synthesis of "artificial trajectories" from the given sample of trajectories, and show that this idea opens new avenues for designing and analyzing algorithms for batch mode reinforcement learning.

Keywords

Reinforcement Learning; Optimal Control; Artificial Trajectories; Function Approximators

1 Introduction

Optimal control problems arise in many real-life applications, such as engineering [40], medicine [41,35,34] or artificial intelligence [43]. Over the last decade, techniques developed by the Reinforcement Learning (RL) community [43] have become more and more popular for addressing those types of problems. RL was initially focusing on how to design intelligent agents able to interact with their environment so as to optimize a given performance criterion [43]. Since the end of the nineties, many researchers have focused on the resolution of a subproblem of RL: computing high performance policies when the only information available on the environment is contained in a batch collection of trajectories. This subproblem of RL is referred to as batch mode RL [18].

Most of the techniques proposed in the literature for solving batch mode RL problems over large or continuous spaces combine value or policy iteration schemes from the Dynamic Programming (DP) theory [2] with function approximators (e.g., radial basis functions, neural networks, etc) representing (state-action) value functions [7]. These approximators have two main roles: (i) to offer a concise representation of state-action value functions

defined over continuous spaces and (ii) to generalize the information contained in the finite sample of input data. Another family of algorithms that has been less studied in RL adopts a two stage process for solving these batch mode RL problems. First, they train function approximators to learn a model of the environment and, afterwards, they use various optimization schemes (e.g., direct policy search, dynamic programming) to compute a policy which is (near-)optimal with respect to this model.

While successful in many studies, the use of function approximators for solving batch mode RL problems has also drawbacks. In particular, the black box nature of this approach makes performance analysis very difficult, and hence severely hinders the design of new batch mode RL algorithms presenting some a priori desired performance guarantees. Also, the policies inferred by these algorithms may have counter-intuitive properties. For example, in a deterministic framework, for a fixed initial state, and when there is in the input sample a trajectory that has been generated by an optimal policy starting from this initial state, there is no guarantee that a function approximator-based policy will reproduce this optimal behavior. This is surprising, since a simple “imitative learning” approach would have such a desirable property.

The above observations have lead us to develop a new line of research based on the synthesis of “artificial trajectories” for addressing batch mode RL problems. In our approach, artificial trajectories are rebuilt from the tuples extracted from the given batch of trajectories with the aim of achieving an optimality property. In this paper, we revisit our work on this topic [19–23], with the objective of showing that these ideas open avenues for addressing many batch mode RL related problems. In particular, four algorithms that exploit artificial trajectories will be presented. The first one computes an estimate of the performance of a given control policy [22]. The second one provides a way for computing performance guarantees in deterministic settings [19]. The third one leads to the computation of policies having high performance guarantees [20, 23], and the fourth algorithm presents a sampling strategy for generating additional trajectories [21]. Finally, we highlight connections between the concept of artificial trajectory synthesis and other standard batch mode RL techniques.

The paper is organized as follows. First, Section 2 gives a brief review of the field of batch mode RL. Section 3 presents the batch mode RL setting adopted in this paper and several of the generic problems it raises. In Section 4, we present our new line of research articulated around the synthesis of artificial trajectories. Finally, Section 5 proposes to make the link between this paradigm and existing batch mode RL techniques, and Section 6 concludes.

2 Related Work

Batch mode RL techniques are probably rooted in the works of Bradtke and Barto [6] and Boyan [4] related to the use of least-squares techniques in the context of Temporal Difference learning methods (LSTD) for estimating the return of control policies. Those works have been extended to address optimal control problems by Lagoudakis and Parr [27] who have introduced the Least-Square Policy Iteration (LSPI) algorithm that mimics the policy iteration algorithm of the DP theory [2]. Several papers have proposed some theoretical works related to least-squares TD-based algorithms, such as for example Nedić and Bertsekas [36] and Lazaric et al. [29,30].

Another algorithm from the DP theory, the value iteration algorithm, has also served as inspiration for designing batch mode RL algorithms. For example, Ormoneit and Sen have developed a batch mode RL algorithm in 2002 [37] using kernel approximators, for which theoretical analyses are also provided. Reference [12] proposes an algorithm that combines value iteration with any type of regressors (e.g., regression trees, SVMs, neural networks).

Reference [13] has named this algorithm Fitted Q Iteration (FQI) and provides a careful empirical analysis of its performance when combined with ensembles of regression trees.

References [40, 28] and [44] study the performances of this FQI algorithm with (deep) neural networks and CMACs (Cerebella Model Articulator Controllers). The Regularized FQI algorithm proposes to use penalized least-squares regression as function approximator to limit the model-complexity of the original FQI algorithm [17]. Extensions of the FQI algorithm to continuous action spaces have also been proposed [1]. More theoretical works related with FQI have also been published [33, 10].

Applications of these batch mode RL techniques have already lead to promising results in robotics [38,3,45], power systems [14], image processing [15], water reservoir optimization [9,8], medicine [34,16,26] and driving assistance strategies [39].

3 Batch Mode RL: Formalization and Typical Problems

We consider a stochastic discrete-time system whose dynamics is given by

$$x_{t+1}=f(x_t+u_t+w_t) \qquad \forall t \in \{0,\ldots,T-1\}$$

where x_t belongs to a state space $\mathcal{X} \subset \mathbb{R}^d$, Where $\mathbb{R}^d$ is the $d-$ dimensional Euclidean space and $T \in \mathbb{N}\backslash\{0\}$ denotes the finite optimization horizon. At every time $t \in \{0,\ldots, T-1\}$, the system can be controlled by taking an action $u_t \in \mathcal{U}$, and is subject to a random disturbance $w_t \in \mathcal{W}$ drawn according to a probability distribution $p_{\mathcal{W}}(\cdot)$[1]. To each system transition from time t to $t+1$ is associated a reward signal:

$$r_t=\rho(x_t, u_t, w_t) \qquad \forall t \in \{0,\ldots,T-1\}.$$

Let $h: \{0,\ldots,T-1\} \times \mathcal{X} \rightarrow \mathcal{U}$ be a control policy. When starting from a given initial state x_0 and following the control policy h, an agent will get a random sum of rewards signal $R^h(x_0)$:

$$R^h(x_0, w_0, \ldots, w_{T-1})=\sum_{t=0}^{T-1}\rho(x_t, h(t,x_t), w_t)$$
$$\text{with} \quad x_{t+1}=f(x_t, h(t,x_t), w_t) \qquad \forall t \in \{0,\ldots,T-1\}$$
$$w_t \sim p_{\mathcal{W}}(\cdot).$$

In RL, the classical performance criterion for evaluating a policy h is its expected T–stage return:

Definition 1 (Expected *T*–stage Return)

$$J^h(x_0)=\mathop{\mathbb{E}}_{w_0,\ldots,w_{T-1}\sim p_{\mathcal{W}}(.)}[R^h(x_0, w_0, \ldots, w_{T-1})],$$

[1]Here the fundamental assumption is that w_t is independent of $w_{t-1}, w_{t-2}, \ldots, w_0$ given x_t and u_t; to simplify all notations and derivations, we furthermore impose that the process is time-invariant and does not depend on the states and actions x_t, u_t.

but, when searching for risk-aware policies, it is also of interest to consider the *value at risk* criterion:

Definition 2 (*T*–stage Value at Risk)

Let $b \in \mathbb{R}$ and $c \in [0, 1[$.

$$J_{VaR}^{h,(b,c)}(x_0) = \begin{cases} -\infty & \text{if } P(R^h(x_0, w_0, \ldots, w_{T-1}) < b) > c, \\ J^h(x_0) & \text{otherwise.} \end{cases}$$

The central problem of batch mode RL is to find a good approximation of a policy h^* that optimizes one such performance criterion, given the fact that *the functions f*, ρ *and* $p_W(\cdot)$ *are unknown*, and thus not accessible to simulation. Instead, they are "replaced" by a batch collection of $n \in \mathbb{N}\backslash\{0\}$ *elementary pieces of trajectories*, defined according to the following process.

Definition 3 (Sample of transitions)

Let

$$\mathcal{P}_n = \{(x^l, u^l)\}_{l=1}^n \in (\mathcal{X} \times \mathcal{U})^n$$

be a given set of state-action pairs. Consider the ensemble of samples of one-step transitions of size n that could be generated by complementing each pair (x^l, u^l) of $\mathcal{P}_n$ by drawing for each l a disturbance signal w^l at random from $p_W(.)$, and by recording the resulting values of $\rho(x^l, u^l, w^l)$ and $f(x^l, u^l, w^l)$. We denote by $\bar{\mathcal{F}}_n(\mathcal{P}_n, w^1, \ldots, w^n)$ one such "random" set of one-step transitions defined by a random draw of n disturbance signals w^l $l = 1 \ldots n$. We assume that we know one realization of the random set $\bar{\mathcal{F}}_n(\mathcal{P}_n, w^1, \ldots, w^n)$, that we denoted by $\mathcal{F}_n$:

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

where, for all $l \in \{1, \ldots, n\}$,

$$\forall l \in \{1, \ldots, n\}, \qquad \begin{aligned} r^l &= \rho(x^l, u^l, w^l), \\ y^l &= f(x^l, u^l, w^l), \end{aligned}$$

for some realizations of the disturbance process $w^l \sim p_W(\cdot)$.

Notice first that the resolution of the central problem of finding a good approximation of an optimal policy h^* is very much correlated to the problem of estimating the performance of a given policy. Indeed, when this latter problem is solved, the search for an optimal policy can in principle be reduced to an optimization problem over the set of candidate policies. We thus will start by addressing the problem of characterizing the performance of a given policy.

It is sometimes desirable to be able to compute policies having good performance guarantees. Indeed, for many applications, even if it is perhaps not paramount to have a policy h which is very close to the optimal one, it is however crucial to be able to guarantee that the considered policy h leads to high-enough cumulated rewards. The problem of computing such policies will also be addressed later in this paper.

In many applications, one has the possibility to move away from a pure batch setting by carrying out a limited number of experiments on the real system in order to enrich the available sample of trajectories. We thus also consider the problem of designing strategies for generating optimal experiments for batch mode RL.

4 Synthesizing Artificial Trajectories

We first formalize the concept of artificial trajectories in Section 4.1. In Section 4.2, we detail, analyze and illustrate on a benchmark how artificial trajectories can be exploited for estimating the performances of policies. We focus in Section 4.3 on the deterministic case, and we show how artificial trajectories can be used for computing bounds on the performances of policies. Afterwards, we exploit these bounds for addressing two different problems: the first problem (Section 4.4) is to compute policies having good performance guarantees. The second problem (Section 4.5) is to design sampling strategies for generating additional system transitions.

4.1 Artificial Trajectories

Artificial trajectories are made of elementary pieces of trajectories (one-step system transitions) taken from the sample $\mathcal{F}_n$. Formally, an artificial trajectory is defined as follows:

Definition 4 (Artificial Trajectory)—An *artificial trajectory* is an (ordered) sequence of T one-step system transitions:

$$[(x^{l_0}, u^{l_0}, r^{l_0}, y^{l_0}), \ldots, (x^{l_{T-1}}, u^{l_{T-1}}, r^{l_{T-1}}, y^{l_{T-1}})] \in \mathcal{F}_n^T$$

where

$$l_t \in \{1, \ldots, n\}, \qquad \forall t \in \{0, \ldots, T-1\}.$$

We give in Figure 1 an illustration of one such artificial trajectory.

Observe that one can synthesize n^T different artificial trajectories from the sample of transitions $\mathcal{F}_n$. In the rest of this paper, we present various techniques for extracting and exploiting “interesting” subsets of artificial trajectories.

4.2 Evaluating the Expected Return of a Policy

A major subproblem of batch mode RL is to evaluate the expected return $J^h(x_0)$ of a given policy h. Indeed, when such an oracle is available, the search for an optimal policy can be in some sense reduced to an optimization problem over the set of all candidate policies. When a model of the system dynamics, reward function and disturbances probability distribution is available, Monte Carlo estimation techniques can be run to estimate the performance of any control policy. But, this is indeed not possible in the batch mode setting. In this section, we detail an approach that estimates the performance of a policy by rebuilding artificial

trajectories so as to mimic the behavior of the Monte Carlo estimator. We assume in this section (and also in Section 4.3) that the action space $\mathcal{U}$ is continuous and normed.

4.2.1 Monte Carlo Estimation—The Monte Carlo (MC) estimator works in a model-based setting (i.e., in a setting where f, ρ and $p_{\mathcal{W}}(.)$ are known). It estimates $J^h(x_0)$ by averaging the returns of several (say $p \in \mathbb{N}_0$) trajectories which have been generated by simulating the system from x_0 using the policy h. More formally, the MC estimator of the expected return of the policy h when starting from the initial state x_0 writes:

Definition 5 (Monte Carlo Estimator)

$$\mathbb{M}_p^h(x_0) = \frac{1}{p}\sum_{i=1}^{p}\sum_{t=0}^{T-1}\rho(x_t^i, h(t, x_t^i), w_t^i)$$

With $\forall t \in \{0,\ldots,T-1\}$, $\forall i \in \{1,\ldots,p\}$,

$$\begin{aligned} w_t^i &\sim p_{\mathcal{W}}(.), \\ x_0^i &= x_0, \\ x_{t+1}^i &= f(x_t^i, h(t, x_t^i), w_t^i). \end{aligned}$$

The bias and variance of the MC estimator are:

Proposition 1 (Bias and Variance of the MC Estimator)

$$\begin{aligned} \mathop{\mathbb{E}}_{w_t^i \sim p_{\mathcal{W}}(.), i=1\ldots p, t=0\ldots T-1} [\mathbb{M}_p^h(x_0) - J^h(x_0)] &= 0, \\ \mathop{\mathbb{E}}_{w_t^i \sim p_{\mathcal{W}}(.), i=1\ldots p, t=0\ldots T-1} [(\mathbb{M}_p^h(x_0) - J^h(x_0))^2] &= \frac{\sigma_{R^h}^2(x_0)}{p}. \end{aligned}$$

where $\sigma_{R^h}^2(x_0)$ denotes the assumed finite variance of $R^h(x_0, w_0,\ldots,w_{T-1})$:

$$\sigma_{R^h}^2(x_0) = \mathop{Var}_{w_0,\ldots,w_{T-1} \sim p_{\mathcal{W}}(.)} [R^h(x_0, w_0, \ldots, w_{T-1})] < +\infty.$$

4.2.2 Model-free Monte Carlo Estimation—From a sample $\mathcal{F}_n$, our model-free MC (MFMC) estimator works by rebuilding $p \in \mathbb{N} \setminus \{0\}$ artificial trajectories. These artificial trajectories will then serve as proxies of p "actual" trajectories that could be obtained by simulating the policy h on the given control problem. Our estimator averages the cumulated returns over these artificial trajectories to compute its estimate of the expected return $J^h(x_0)$. The main idea behind our method amounts at selecting the artificial trajectories so as to minimize the discrepancy of these trajectories with a classical MC sample that could be obtained by simulating the system with policy h.

To rebuild a sample of p artificial trajectories of length T starting from x_0 and similar to trajectories that would be induced by a policy h, our algorithm uses each one-step transition in $\mathcal{F}_n$ at most once; we thus assume that $pT \leq n$. The p artificial trajectories of T one-step

transitions are created sequentially. Every artificial trajectory is grown in length by selecting, among the sample of not yet used one-step transitions, a transition whose first two elements minimize the distance – using a distance metric Δ in $\mathcal{X} \times \mathcal{U}$ – with the couple formed by the last element of the previously selected transition and the action induced by h at the end of this previous transition. Because we do not re-use one-step transitions, the disturbances associated with the selected transitions are i.i.d., which provides the MFMC estimator with interesting theoretical properties (see Section 4.2.3). Consequently, this also ensures that the p rebuilt artificial trajectories will be distinct.

Algorithm 1 MFMC algorithm to rebuilt a set of size p of T–length artificial trajectories from a sample of n one-step transitions.

Input: $\mathcal{F}_n$, $h(.,.), x_0, \Delta(.,.), T, p$

Let $\mathcal{G}$ denote the current set of not yet used one-step transitions in $\mathcal{F}_n$; Initially,

$\mathcal{G} \leftarrow \mathcal{F}_n$;

for $i = 1$ to p (extract an artificial trajectory) **do**

$t \leftarrow 0$;

$x_t^i \leftarrow x_0$;

while $t < T$ **do**

$u_t^i \leftarrow h\left(t, x_t^i\right)$;

$\mathcal{H} \leftarrow \underset{(x,u,r,y)\in\mathcal{G}}{\arg\min} \Delta((x,u),(x_t^i,u_t^i))$;

$l_t^i \leftarrow$ lowest index in $\mathcal{F}_n$ of the transitions that belong to $\mathcal{H}$;

$t \leftarrow t + 1$;

$x_t^i \leftarrow y^{l_t^i}$;

$\mathcal{G} \leftarrow \mathcal{G} \setminus \left\{\left(x^{l_t^i}, u^{l_t^i}, r^{l_t^i}, y^{l_t^i}\right)\right\}$; \\ *do not re-use transitions*

end while

end for

Return the set of indices $\{l_t^i\}_{i=1,t=0}^{i=p,t=T-1}$.

A tabular version of the algorithm for building the set of artificial trajectories is given in Table 1. It returns a set of indices of one-step transitions $\{l_t^i\}_{i=1,t=0}^{i=p,t=T-1}$ from $\mathcal{F}_n$ based on h, x_0, the distance metric Δ and the parameter p. Based on this set of indices, we define our MFMC estimate of the expected return of the policy h when starting from the initial state x_0:

Definition 6 (Model-free Monte Carlo Estimator)

$$\mathcal{M}_p^h(\mathcal{F}_n, x_0) = \frac{1}{p}\sum_{i=1}^{p}\sum_{t=0}^{T-1} r^{l_t^i}.$$

Figure 2 illustrates the MFMC estimator. Note that the computation of the MFMC estimator $\mathscr{M}_p^h(\mathscr{F}_n, x_0)$ has a linear complexity with respect to the cardinality n of $\mathcal{F}_n$, the number of artificial trajectories p and the optimization horizon T.

4.2.3 Analysis of the MFMC Estimator—In this section we characterize some main properties of our estimator. To this end, we study the distribution of our estimator $\mathscr{M}_p^h\left(\tilde{\mathscr{F}}_n(\mathscr{P}_n, w^1, \ldots, w^n), x_0\right)$, seen as a function of the random set $\tilde{\mathscr{F}}_n(\mathscr{P}_n, w^1, \ldots, w^n)$; in order to characterize this distribution, we express its bias and its variance as a function of a measure of the density of the sample $\mathcal{P}_n$, defined by its "k–dispersion"; this is the smallest radius such that all Δ-balls in $\mathcal{X} \times \mathcal{U}$ of this radius contain at least k elements from $\mathcal{P}_n$. The use of this notion implies that the space $\mathcal{X} \times \mathcal{U}$ is bounded (when measured using the distance metric Δ).

The bias and variance characterization will be done under some additional assumptions detailed below. After that, we state the main theorems formulating these characterizations. Proofs are given in [22].

Assumption: Lipschitz continuity of the functions f, ρ and h: We assume that the dynamics f, the reward function ρ and the policy h are Lipschitz continuous, i.e., we assume that there exist finite constants L_f, L_ρ and $L_h \in \mathbb{R}^+$ such that:

$$\begin{aligned}
&\forall (x, x', u, u', w) \in \mathcal{X}^2 \times \mathcal{U}^2 \times \mathcal{W},\\
&\| f(x,u,w) - f(x',u',w) \|_{\mathcal{X}} \leq L_f(\| x - x' \|_{\mathcal{X}} + \| u - u' \|_{\mathcal{U}}),\\
&|\rho(x,u,w) - \rho(x',u',w)| \leq L_\rho(\| x - x' \|_{\mathcal{X}} + \| u - u' \|_{\mathcal{U}}),\\
&\| h(t,x) - h(t,x') \|_{\mathcal{U}} \leq L_h \| x - x' \|_{\mathcal{X}}, \forall t \in \{0, \ldots, T-1\},
\end{aligned}$$

where $\| . \|_{\mathcal{X}}$ and $\| . \|_{\mathcal{U}}$ denote the chosen norms over the spaces $\mathcal{X}$ and $\mathcal{U}$, respectively.

Assumption: $\mathcal{X} \times \mathcal{U}$ is bounded: We suppose that $\mathcal{X} \times \mathcal{U}$ is bounded when measured using the distance metric Δ.

Definition 7 (Distance Metric Δ)

$$\forall (x, x', u, u') \in \mathcal{X}^2 \times \mathcal{U}^2, \qquad \Delta((x,u),(x',u')) = \| x - x' \|_{\mathcal{X}} + \| u - u' \|_{\mathcal{U}}.$$

Given $k \in \mathbb{N}\backslash\{0\}$ with $k \leq n$, we define the *k–dispersion*, $\alpha_k(\mathcal{P}_n)$ of the sample $\mathcal{P}_n$:

Definition 8 (k–Dispersion)

$$\alpha_k(\mathscr{P}_n) = \sup_{(x,u) \in \mathcal{X} \times \mathcal{U}} \Delta_k^{\mathscr{P}_n}(x,u),$$

where $\Delta_k^{\mathscr{P}_n}(x,u)$ denotes the distance of (x, u) to its k–th nearest neighbor (using the distance metric Δ) in the $\mathcal{P}_n$ sample.

Definition 9 (Expected Value of $\mathscr{M}_p^h\left(\tilde{\mathscr{F}}(\mathscr{P}_n, w^1, \ldots, w^n), x_0\right)$): We denote by $E^h_{p,\mathscr{P}_n}(x_0)$ the expected value:

$$E^h_{p,\mathcal{P}_n}(x_0) = \mathop{\mathbb{E}}_{w^1,\ldots,w^n \sim p\mathcal{W}(.)} \left[\mathcal{M}^h_p \left(\tilde{F}_n(\mathcal{P}_n, w^1, \ldots, w^n), x_0 \right) \right].$$

We have the following theorem:

Theorem 1 (Bias Bound for $\mathcal{M}^h_p\left(\tilde{\mathcal{F}}_n(\mathcal{P}_n, w^1, \ldots, w^n), x_0\right)$)

$$|J^h(x_0) - E^h_{p,\mathcal{P}_n}(x_0)| \leq C\alpha_{pT}(\mathcal{P}_n) \;\; with \; C = L_\rho \sum_{t=0}^{T-1} \sum_{i=0}^{T-t-1} (L_f(1+L_h))^i.$$

The proof of this result is given in [22]. This formula shows that the bias is bounded closer to the target estimate if the sample dispersion is small. Note that the sample dispersion itself actually only depends on the sample $\mathcal{P}_n$ and on the value of p (it will increase with the number of trajectories used by our algorithm).

Definition 10 (Variance of $\mathcal{M}^h_p\left(\tilde{\mathcal{F}}_n(\mathcal{P}_n, w^1, \ldots, w^n), x_0\right)$): We denote by $V^h_{p,\mathcal{P}_n}(x_0)$ the variance of the MFMC estimator defined by

$$V^h_{p,\mathcal{P}_n}(x_0) = \mathop{\mathbb{E}}_{w^1,\ldots,w^n \sim p\mathcal{W}(.)} \left[\left(\mathcal{M}^h_p \left(\tilde{\mathcal{F}}_n(\mathcal{P}_n, w^1, \ldots, w^n), x_0 \right) - E^h_{p,\mathcal{P}_n}(x_0) \right)^2 \right]$$

and we give the following theorem:

Theorem 2 (Variance Bound for $\mathcal{M}^h_p\left(\tilde{\mathcal{F}}_n(\mathcal{P}_n, w^1, \ldots, w^n), x_0\right)$)

$$V^h_{p,\mathcal{P}_n}(x_0) \leq \left(\frac{\sigma_{R^h}(x_0)}{\sqrt{p}} + 2C\alpha_{pT}(\mathcal{P}_n) \right)^2 \qquad with \;\; C = L_\rho \sum_{t=0}^{T-1} \sum_{i=0}^{T-t-1} (L_f(1+L_h))^i.$$

The proof of this theorem is given in [22]. We see that the variance of our MFMC estimator is guaranteed to be close to that of the classical MC estimator if the sample dispersion is small enough.

• Illustration: In this section, we illustrate the MFMC estimator on an academic problem. The system dynamics and the reward function are given by

$$x_{t+1} = \sin\left(\frac{\pi}{2}(x_t + u_t + w_t) \right)$$

and

$$\rho(x_t, u_t, w_t) = \frac{1}{2\pi} e^{-\frac{1}{2}(x_t^2 + u_t^2)} + w_t$$

with the state space $\mathcal{X}$ being equal to [−1,1] and the action space $\mathcal{U}$ to [−1,1]. The disturbance w_t is an element of the interval $\mathcal{W} = \left[-\frac{\epsilon}{2}, \frac{\epsilon}{2}\right]$ with $\epsilon = 0.1$ and $p_{\mathcal{W}}$ is a uniform probability distribution over this interval. The optimization horizon T is equal to 15. The policy h whose performances have to be evaluated is

$$h(t,x) = -\frac{x}{2}, \qquad \forall x \in \mathcal{X}, \forall t \in \{0,\ldots,T-1\}.$$

The initial state of the system is set $x_0 = -0.5$.

For our first set of experiments, we choose to work with a value of $p = 10$ i.e., the MFMC estimator rebuilds 10 artificial trajectories to estimate $J^h(-0.5)$. In these experiments, for different cardinalities $n_j = (10j)^2 = m_j^2 \quad j = 1 \ldots 10$, we build a sample $\mathcal{P}_{nj} = \{(x^l, u^l)\}$ that uniformly cover the space $\mathcal{X} \times \mathcal{U}$ as follows:

$$x^l = -1 + \frac{2j_1}{m_j} \text{ and } u^l = -1 + \frac{2j_2}{m_j} \qquad j_1, j_2 \in \{0,\ldots,m_j - 1\}.$$

Then, we generate 50 random sets $\mathcal{F}_{n_j}^1, \ldots, \mathcal{F}_{n_j}^{50}$ Over $\mathcal{P}_{n_j}$ and run our MFMC estimator on each of these sets. The results of this first set of experiments are gathered in Figure 3. For every value of n_j considered in our experiments, the 50 values computed by the MFMC estimator are concisely represented by a boxplot. The box has lines at the lower quartile, median, and upper quartile values. Whiskers extend from each end of the box to the adjacent values in the data within 1.5 times the interquartile range from the ends of the box. Outliers are data with values beyond the ends of the whiskers and are displayed with a red + sign. The squares represent an accurate estimate of $J^h(-0.5)$ computed by running thousands of Monte Carlo simulations. As we observe, when the samples increase in size (which corresponds to a decrease of the pT–dispersion $\alpha_{pT}(\mathcal{P}_n)$) the MFMC estimator is more likely to output accurate estimations of $J^h(-0.5)$. As explained throughout this paper, there exist many similarities between the model-free MFMC estimator and the model-based MC estimator. These can be empirically illustrated by putting Figure 3 in perspective with Figure 4. This latter figure reports the results obtained by 50 independent runs of the MC estimator, each one of these runs using also $p = 10$ trajectories. As expected, one can see that the MFMC estimator tends to behave similarly to the MC estimator when the cardinality of the sample increases.

In our second set of experiments, we choose to study the influence of the number of artificial trajectories p upon which the MFMC estimator bases its prediction. For each value $p_j = j^2$ $j = 1 \ldots 10$ we generate 50 samples $\mathcal{F}_{10,000}^1, \ldots, \mathcal{F}_{10,000}^{50}$ of one-step transitions of cardinality 10, 000 (using the sample $\mathcal{P}_{10000}$ defined in the first set of experiments) and use these samples to compute the MFMC estimator. The results are plotted in Figure 5. This figure shows that the bias of the MFMC estimator seems to be relatively small for small values of p and to increase with p. This is in accordance with Theorem 1 which bounds the bias with an expression that is increasing with p.

In Figure 6, we have plotted the evolution of the values computed by the model-based MC estimator when the number of trajectories it considers in its prediction increases. While, for small numbers of trajectories, it behaves similarly to the MFMC estimator, the quality of its

predictions steadily increases with p, while it is not the case for the MFMC estimator whose performances degrade once p crosses a threshold value. Notice that this threshold value could be made larger by increasing the size of the samples of one-step system transitions used as input of the MFMC algorithm.

4.2.4 MFMC Estimation of the value at risk—In order to take into consideration the riskiness of policies - and not only their good performances "on average" -, one may prefer to consider a Value-at-Risk–type performance criterion instead of expected return. Notice that this type of criterion has received more and more attention during the last few years inside the RL community [11,31,32].

If we consider the p artificial trajectories that are rebuilt by the MFMC estimator, the value at risk $J_{VaR}^{h,(b,c)}(x_0)$ can be efficiently approximated by the value $\tilde{J}_{VaR}^{h,(b,c)}(x_0)$ defined as follows:

Definition 11 (Estimate of the *T*–stage Value at Risk): Let $b \in \mathbb{R}$ and $c \in [0, 1[$.

$$\tilde{J}_{VaR}^{h,(b,c)}(x_0) = \begin{cases} -\infty & \text{if } \frac{1}{p}\sum_{i=1}^{p} \mathbb{I}_{\{\mathbf{r}^i < b\}} > c, \\ \mathscr{M}^h(\mathscr{F}_n, x_0) & \text{otherwise} \end{cases}$$

where $\mathbf{r}^i$ denotes the return of the i–th artificial trajectory:

$$\mathbf{r}^i = \sum_{t=0}^{T-1} r^{l_t^i}.$$

4.3 Artificial Trajectories in the Deterministic Case: Computing Bounds

From this subsection to the end of Section 4, we assume a deterministic environment. More formally, we assume that the disturbances space is reduced to a single element $\mathcal{W} = \{0\}$ which concentrates the whole probability mass $p_{\mathcal{W}}(0) = 1$. We abusively use the convention:

$$\forall (x, u) \in \mathcal{X} \times \mathcal{U}, \qquad f(x, u) = f(x, u, 0), \rho(x, u) = \rho(x, u, 0).$$

We still assume that the functions f, ρ and h are Lipschitz continuous. Observe that, in a deterministic context, only one trajectory is needed to compute $J^h(x_0)$ by Monte Carlo estimation. We have the following result:

Proposition 2 (Lower Bound from the MFMC)—Let $[(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1}$ be an artificial trajectory rebuilt by the MFMC algorithm when using the distance measure Δ. Then, we have

$$|\mathscr{M}_1^h(\mathscr{F}_n, x_0) - J^h(x_0)| \leq \sum_{t=0}^{T-1} L_{Q_{T-t}} \Delta((y^{l_{t-1}}, h(t, y^{l_{t-1}})), (x^{l_t}, u^{l_t}))$$

where

$$L_{Q_{T-t}} = L_\rho \sum_{i=0}^{T-t-1} (L_f(1+L_h))^i$$

and $y^{l-1} = x_0$.

The proof of this theorem can be found in [19]. Since the previous result is valid for any artificial trajectory, we have:

Corollary 1 (Lower Bound from any Artificial Trajectory)—Let $[(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1}$ be any artificial trajectory. Then,

$$J^h(x_0) \geq \sum_{t=0}^{T-1} r^{l_t} - \sum_{t=0}^{T-1} L_{Q_{T-t}} \Delta((y^{l_{t-1}}, h(t, y^{l_{t-1}})), (x^{l_t}, u^{l_t}))$$

This suggests to identify an artificial trajectory that leads to the maximization of the previous lower bound:

Definition 12 (Maximal Lower Bound)

$$L^h(\mathcal{F}_n, x_0) = \max_{[(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1} \in \mathcal{F}_n^T} \sum_{t=0}^{T-1} r^{l_t} - \sum_{t=0}^{T-1} L_{Q_{T-t}} \Delta((y^{l_{t-1}}, h(t, y^{l_{t-1}})), (x^{l_t}, u^{l_t})).$$

Note that in the same way, a minimal upper bound can be computed:

Definition 13 (Minimal Upper Bound)

$$U^h(\mathcal{F}_n, x_0) = \min_{[(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1} \in \mathcal{F}_n^T} \sum_{t=0}^{T-1} r^{l_t} + \sum_{t=0}^{T-1} L_{Q_{T-1}} \Delta((y^{l_{t-1}}, h(t, y^{l_{t-1}})), (x^{l_t}, u^{l_t})).$$

Additionaly, we can prove that both the lower and the upper bound are tight, in the sense that they both converge towards $J^h(x_0)$ when the dispersion of the sample of system transitions $\mathcal{F}_n$ decreases towards zero.

Proposition 3 (Tightness of the Bounds)

$$\exists C_b > 0: \quad J^h(x_0) - L^h(\mathcal{F}_n, x_0) \leq C_b \alpha_1(\mathcal{P}_n)$$
$$U^h(\mathcal{F}_n, x_0) - J^h(x_0) \leq C_b \alpha_1(\mathcal{P}_n)$$

where $\alpha_1(\mathcal{P}_n)$ denotes the 1 – dispersion of the sample of system transitions $\mathcal{F}_n$.

This result is proved in [19]. Note that the computation of both the maximal lower bound and minimal upper bound can be reformulated as a shortest path problem in a graph, for which the computational complexity is linear with respect to the optimization horizon T and quadratic with respect to the cardinality n of the sample of transitions $\mathcal{F}_n$.

4.3.1 Extension to Finite Action Spaces—The results given above can be extended to the case where the action space $\mathcal{U}$ is finite (and thus discrete) by considering policies that are fully defined by a sequence of actions. Such policies can be qualified as "open-loop". Let Π be the set of open-loop policies:

Definition 14 (Open-loop Policies)

$$\Pi=\{\pi:\{0,\ldots,T-1\}\rightarrow\mathcal{U}\}$$

Given an open-loop policy π, the (deterministic) T–stage return of π writes:

$$J^{\pi}(x_0)=\sum_{t=0}^{T-1}\rho(x_t,\pi(t))$$

With

$$x_{t+1}=f(x_t,\pi(t)),\qquad \forall t\in\{0,\ldots,T-1\}.$$

In the context of a finite action space, the Lipschitz continuity of f and ρ is: $\forall(x, x', u)\in\mathcal{X}^2\times\mathcal{U}$,

$$\|f(x,u)-f(x',u)\|_{\mathcal{X}}\leq L_f\|x-x'\|_{\mathcal{X}},\ \ |\rho(x,u)-\rho(x',u)|\leq L_\rho\|x-x'\|_{\mathcal{X}}.$$

Since the action space is not normed anymore, we also need to redefine the sample dispersion.

Definition 15 (Sample Dispersion): We assume that the state space is bounded, and we define the sample dispersion $\alpha^*(\mathcal{P}_n)$ as the smallest α such that

$$\exists\alpha>0:\qquad \sup_{x\in\mathcal{X}}\ \min_{l\in\{1,\ldots,n\}}\|x^l-x\|_{\mathcal{X}}\leq\alpha.$$

Let $\pi\in\Pi$ be an open-loop policy. We have the following result:

Proposition 4 (Lower Bound - Open-loop Policy π): Let $[(x^{l_t},u^{l_t},r^{l_t},y^{l_t})]_{t=0}^{T-1}$ be an artificial trajectory such that

$$u^{l_t}=\pi(t)\qquad\forall t\in\{0,\ldots,T-1\}.$$

Then,

$$J^{\pi}(x_0) \geq \sum_{t=0}^{T-1} r^{l_t} - \sum_{t=0}^{T-1} L'_{Q_{T-t}} \| y^{l_{t-1}} - x^{l_t} \|_{\mathcal{X}}.$$

where

$$L'_{Q_{T-t}} = L_\rho \sum_{i=0}^{T-t-1} (L_f)^i.$$

A maximal lower bound can then be computed by maximizing the previous bound over the set of all possible artificial trajectories that satisfy the condition $u^{l_t} = \pi(t)\ \forall t \in \{0,\ldots,T-1\}$. In the following, we denote by $\mathcal{F}^T_{n,\pi}$ the set of artificial trajectories that satisfy this condition:

$$\mathcal{F}^T_{n,\pi} = \left\{ [(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1} \in \mathcal{F}^T_n \,|\, u^{l_t} = \pi(t) \quad \forall t \in 0,\ldots,T-1 \right\}$$

Then, we have:

Definition 16 (Maximal Lower Bound - Open-loop Policy π)

$$L^{\pi}(\mathcal{F}_n, x_0) = \max_{[(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1} \in \mathcal{F}^T_{n,\pi}} \sum_{t=0}^{T-1} r^{l_t} - \sum_{t=0}^{T-1} L'_{Q_{T-t}} \| y^{l_{t-1}} - x^{l_t} \|_{\mathcal{X}}.$$

Similarly, a minimal upper bound $U^{\pi}(\mathcal{F}_n, x_0)$ can also be computed:

Definition 17 (Minimal Upper Bound - Open-loop Policy π)

$$U^{\pi}(\mathcal{F}_n, x_0) = \min_{[(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1} \in \mathcal{F}^T_{n,\pi}} \sum_{t=0}^{T-1} r^{l_t} + \sum_{t=0}^{T-1} L'_{Q_{T-t}} \| y^{l_{t-1}} - x^{l_t} \|_{\mathcal{X}}.$$

Both bounds are tight in the following sense:

Proposition 5 (Tightness of the Bounds - Open-loop Policy π)

$$\exists C'_b > 0: \quad J^{\pi}(x_0) - L^{\pi}(\mathcal{F}_n, x_0) \leq c'_b \alpha^*(\mathcal{P}_n),\ U^{\pi}(\mathcal{F}_n, x_0) - J^{\pi}(x_0) \leq C'_b \alpha^*(\mathcal{P}_n).$$

The proofs of the above stated results are given in [20].

4.4 Artificial Trajectories for Computing Safe Policies

Like in Section 4.3.1, we still assume that the action space $\mathcal{U}$ is finite, and we consider open-loop policies. To obtain a policy with good performance guarantees, we suggest to find an open-loop policy $\widehat{\pi}^*_{\mathcal{F}_n, x_0} \in \Pi$ such that:

$$\widehat{\pi}^*_{\mathcal{F}_n,x_0} \in \arg\max_{\pi\in\Pi} \; L^\pi(\mathcal{F}_n, x_0).$$

Recall that such an "open-loop" policy is optimized with respect to the initial state x_0. Solving the above optimization problem can be seen as identifying an optimal rebuilt artificial trajectory $\left[\left(x^{l^*_t}, u^{l^*_t}, r^{l^*_t}, y^{l^*_t}\right)\right]_{t=0}^{T-1}$ and outputting as open-loop policy the sequence of actions taken along this artificial trajectory:

$$\forall t \in \{0,\ldots,T-1\}, \qquad \widehat{\pi}^*_{\mathcal{F}_n,x_0}(t) = u^{l^*_t}.$$

Finding such a policy can again be done in an efficient way by reformulating the problem as a shortest path problem in a graph. We provide in [20] an algorithm called CGRL (which stands for "Cautious approach to Generalization in RL") of complexity $\mathcal{O}(n^2T)$ for finding such a policy. A tabular version of the CGRL is given in Table 2 and an illustration that shows how the CGRL solution can be seen as a shortest path in a graph is also given in Figure 7. We now give a theorem which shows the convergence of the policy $\widehat{\pi}^*_{\mathcal{F}_n,x_0}$ towards an optimal open-loop policy when the dispersion $\alpha^*(\mathcal{P}_n)$ of the sample of transitions converges towards zero.

Theorem 3 (Convergence of $\widehat{\pi}^*_{\mathcal{F}_n,x_0}$)—Let $\mathcal{J}^*(x_0)$ be the set of optimal open-loop policies:

$$\mathcal{J}^*(x_0) = \arg\max_{\pi\in\Pi} \; J^\pi(x_0),$$

Algorithm 2 CGRL algorithm.

Input: $\mathcal{F}_n = \{(x^l, u^l, r^l, y^l)\}_{l=1}^n, L_f, L_\rho, x_0, T$

Initialization:

$D \leftarrow n \times (T-1)$ matrix initialized to zero;

$A \leftarrow n$–dimensional vector initialized to zero;

$B \leftarrow n$–dimensional vector initialized to zero;

Computation of the Lipschitz constants $\left\{L'_{Q_N}\right\}_{N=1}^T$:

$L'_{Q_1} = L_\rho$;

for $k = 2 \ldots T$ **do**

$L'_{Q_k} \leftarrow L_\rho + L_f L'_{Q_{k-1}}$;

end for

$t \leftarrow T-2$;

while $t > -1$ **do**

for $i = 1 \ldots n$ **do**

$$j_0 \leftarrow \arg\max_{j\in\{1,\ldots,n\}} r^j - L'_{Q_{T-t-1}} \| y^i - x^j \|_{\mathcal{X}} + B(j);$$

$$m_0 \leftarrow \max_{j\in\{1,\ldots,n\}} r^j - L'_{Q_{T-t-1}} \| y^i - x^j \|_{\mathcal{X}} + B(j);$$

A(i) ← m_0;

$D(i, t+1) \leftarrow j_0$; \\ best tuple at $t+1$ if in tuple i at time t

end for

$B \leftarrow A$;

$t = t-1$;

end while

Conclusion:

$S \leftarrow (T+1)$–length vector of actions initialized to zero;

$$l \leftarrow \arg\max_{j\in\{1,\ldots,n\}} r^j - L'_{Q_T} \| x_0 - x^j \|_{\mathcal{X}} + B(j);$$

$$S(T+1) \leftarrow \max_{j\in\{1,\ldots,n\}} r^j - L'_{Q_T} \| x_0 - x^j \|_{\mathcal{X}} + B(j);$$ \\ *best lower bound*

$S(1) \leftarrow u^l$; \\ *CGRL* action for $t = 0$.

for t = 0 … T − 2 **do**

$l \leftarrow D(l, t+1)$;

$S(t+2,:) \leftarrow u^{l'}$; *other CGRL actions*

$l \leftarrow l'$;

end for

Return: S

and let us suppose that $\mathcal{J}^*(x_0) \neq \Pi$ (if $\mathcal{J}^*(x_0) = \Pi$, the search for an optimal policy is indeed trivial). We define

$$\epsilon(x_0) = \min_{\pi\in\Pi\setminus\mathcal{J}^*(x_0)} \left\{ \left(\max_{\pi'\in\Pi} J^{\pi'}(x_0) \right) - J^{\pi}(x_0) \right\}.$$

Then,

$$\left(C'_b \alpha^*(\mathcal{P}_n) < \epsilon(x_0) \right) \Rightarrow \widehat{\pi}^*_{\mathcal{F}_n,x_0} \in \mathcal{J}^*(x_0).$$

The proof of this result is also given in [20].

• Illustration: We now illustrate the performances of the policy $\widehat{\pi}^*_{\mathcal{F}_n,x_0}$ computed by the CGRL algorithm on a variant of the puddle world benchmark introduced in [42]. In this benchmark, a robot whose goal is to collect high cumulated rewards navigates on a plane. A puddle stands in between the initial position of the robot and the high reward area (see figure 8). If the robot is in the puddle, it gets highly negative rewards. An optimal navigation strategy drives the robot around the puddle to reach the high reward area. Two datasets of one-step transitions have been used in our example. The first set $\mathcal{F}$ contains elements that

uniformly cover the area of the state space that can be reached within T steps. The set $\mathcal{F}'$ has been obtained by removing from $\mathcal{F}$ the elements corresponding to the highly negative rewards. The full specification of the benchmark and the exact procedure for generating $\mathcal{F}$ and $\mathcal{F}'$ are given in [20]. On Figure 9, we have drawn the trajectory of the robot when following the policy $\widehat{\pi}^*_{\mathcal{F}_n,x_0}$. Every state encountered is represented by a white square. The plane upon which the robot navigates has been colored such that the darker the area, the smaller the corresponding rewards are. In particular, the puddle area is colored in dark grey/black. We see that the policy $\widehat{\pi}^*_{\mathcal{F}_n,x_0}$ drives the robot around the puddle to reach the high-reward area – which is represented by the light-grey circles.

Figure 10 represents the policy inferred from $\mathcal{F}$ by using the (finite-time version of the) Fitted Q Iteration algorithm (FQI) combined with extremely randomized trees as function approximators [13]. The trajectories computed by the policy $\widehat{\pi}^*_{\mathcal{F}_n,x_0}$ and FQI algorithms are very similar and so are the sums of rewards obtained by following these two trajectories. However, by using $\mathcal{F}'$ rather that $\mathcal{F}$, the policy $\widehat{\pi}^*_{\mathcal{F}_n,x_0}$ and FQI algorithms do not lead to similar trajectories, as it is shown on Figures 11 and 12. Indeed, while the policy $\widehat{\pi}^*_{\mathcal{F}_n,x_0}$ still drives the robot around the puddle to reach the high reward area, the FQI policy makes the robot cross the puddle. In terms of optimality, this latter navigation strategy is much worse. The difference between both navigation strategies can be explained as follows. The FQI algorithm behaves as if it were associating to areas of the state space that are not covered by the input sample, the properties of the elements of this sample that are located in the neighborhood of these areas. This in turn explains why it computes a policy that makes the robot cross the puddle. The same behavior could probably be observed by using other algorithms that combine dynamic programming strategies with kernel-based approximators or averagers [5,25,37]. The policy $\widehat{\pi}^*_{\mathcal{F}_n,x_0}$ generalizes the information contained in the dataset, by assuming, given the intial state, the most adverse behavior for the environment according to its weak prior knowledge about the environment. This results in the fact that it penalizes sequences of decisions that could drive the robot in areas not well covered by the sample, and this explains why the policy $\widehat{\pi}^*_{\mathcal{F}_n,x_0}$ drives the robot around the puddle when run with $\mathcal{F}'$.

4.4.1 Taking Advantage of Optimal Trajectories—In this section, we give another result which shows that, in the case where an optimal trajectory can be found in the sample of system transitions, then the policy $\widehat{\pi}^*_{\mathcal{F}_n,x_0}$ computed by the CGRL algorithm is also optimal.

Theorem 4 (Optimal Policies computed from Optimal Trajectories): Let $\pi^*_{x_0} \in \mathcal{J}^*(x_0)$ be an optimal open-loop policy. Let us assume that one can find in $\mathcal{F}_n$ a sequence of T one-step system transitions

$$[(x^{l_0}, u^{l_0}, r^{l_0}, x^{l_1}), (x^{l_1}, u^{l_1}, r^{l_1}, x^{l_2}), \ldots, (x^{l_{T-1}}, u^{l_{T-1}}, r^{l_{T-1}}, x^{l_T})] \in \mathcal{F}_n^T$$

such that

$$\begin{aligned} x^{l_0} &= x_0, \\ u^{l_t} &= \pi^*_{x_0}(t) \qquad \forall t \in \{0, \ldots, T-1\}. \end{aligned}$$

Let $\widehat{\pi}^*_{\mathscr{F}_n,x_0}$ be such that

$$\widehat{\pi}^*_{\mathscr{F}_n,x_0} \in \arg\max_{\pi\in\Pi} L^{\pi}(\mathscr{F}_n, x_0).$$

Then,

$$\widehat{\pi}^*_{\mathscr{F}_n,x_0} \in \mathscr{J}^*(x_0).$$

Proof Let us prove the result by contradiction. Assume that $\widehat{\pi}^*_{\mathscr{F}_n,x_0}$ is not optimal. Since $\pi^*_{x_0}$ is optimal, one has:

$$J^{\widehat{\pi}^*_{\mathscr{F}_n,x_0}}(x_0) < J^{\pi^*_{x_0}}(x_0). \quad (1)$$

Let us now consider the lower bound $\mathscr{B}^{\pi^*_{x_0}}(\mathscr{F}_n, x_0)$ on the return of the policy $\pi^*_{x_0}$ computed from the sequence of transitions

$$[(x^{l_0}, u^{l_0}, r^{l_0}, x^{l_1}), (x^{l_1}, u^{l_1}, r^{l_1}, x^{l_2}), \ldots, (x^{l_{T-1}}, u^{l_{T-1}}, r^{l_{T-1}}, x^{l_T})] \in \mathscr{F}_n^T.$$

By construction of this sequence of transitions, we have:

$$\begin{aligned}\mathscr{B}^{\pi^*_{x_0}}(\mathscr{F}_n, x_0) &= \sum_{t=0}^{T-1} r^{l_t} - \sum_{t=0}^{T-1} L'_{Q_{T-t}} \| x^{l_t} - x^{l_t} \|_{\mathscr{X}} \\ &= \sum_{t=0}^{T-1} r^{l_t} \\ &= J^{h^*_{x_0}}(x_0)\end{aligned}$$

By definition of the policy $\widehat{\pi}^*_{\mathscr{F}_n,x_0} \in \arg\max_{\pi\in\Pi} L^{\pi}(\mathscr{F}_n, x_0)$, we have:

$$L^{\widehat{\pi}^*_{\mathscr{F}_n,x_0}}(\mathscr{F}_n, x_0) \geq \mathscr{B}^{\pi^*_{x_0}}(\mathscr{F}_n, x_0) = J^{\pi^*_{x_0}}(x_0). \quad (2)$$

Since $L^{\widehat{\pi}^*_{\mathscr{F}_n,x_0}}(\mathscr{F}_n, x_0)$ is a lower bound on the return of $\widehat{\pi}^*_{\mathscr{F}_n,x_0}$, we have:

$$J^{\widehat{\pi}^*_{\mathscr{F}_n,x_0}}(x_0) \geq L^{\widehat{\pi}^*_{\mathscr{F}_n,x_0}}(\mathscr{F}_n, x_0). \quad (3)$$

Combining inequalities 2 and 3 yields a contradiction with inequality 1.

4.5 Rebuilding Artificial Trajectories for Designing Sampling Strategies

We suppose in this section that additional system transitions can be generated, and we detail hereafter a sampling strategy to select state-action pairs (x, u) for generating $f(x, u)$ and $\rho(x, u)$ so as to be able to discriminate rapidly – as new one-step transitions are generated –

between optimal and non-optimal policies from Π. This strategy is directly based on the previously described bounds.

Before describing our proposed sampling strategy, let us introduce a few definitions. First, note that a policy can only be optimal given a set of one-step transitions $\mathcal{F}$ if its upper bound is not lower than the lower bound of any element of Π. We qualify as "candidate optimal policies given $\mathcal{F}$" and we denote by $\Pi(\mathcal{F}, x_0)$ the set of policies which satisfy this property:

Definition 18 (Candidate Optimal Policies Given $\mathcal{F}$)

$$\Pi(\mathcal{F}, x_0) = \left\{\pi \in \Pi \;\middle|\; \forall \pi' \in \Pi, U^{\pi}(\mathcal{F}, x_0) \geq L^{\pi'}(\mathcal{F}, x_0)\right\}.$$

We also define the set of "compatible transitions given $\mathcal{F}$" as follows:

Definition 19 (Compatible Transitions Given $\mathcal{F}$)—A transition $(x, u, r, y) \in \mathcal{X} \times \mathcal{U} \times \mathbb{R} \times \mathcal{X}$ is said compatible with the set of transitions $\mathcal{F}$ if

$$\forall (x^l, u^l, r^l, y^l) \in \mathcal{F}, \quad (u^l = u) \Rightarrow \begin{cases} |r - r^l| \leq L_\rho \| x - x^l \|_{\mathcal{X}}, \\ \| y - y^l \|_{\mathcal{X}} \leq L_f \| x - x^l \|_{\mathcal{X}}. \end{cases}$$

We denote by $\mathcal{C}(\mathcal{F}) \subset \mathcal{X} \times \mathcal{U} \times \mathbb{R} \times \mathcal{U}$ the set that gathers all transitions that are compatible with the set of transitions $\mathcal{F}$.

Our sampling strategy generates new one-step transitions iteratively. Given an existing set $\mathcal{F}_m$ of $m \in \mathbb{N}\backslash\{0\}$ one-step transitions, which is made of the elements of the initial set $\mathcal{F}_n$ and the m-n one-step transitions generated during the first m-n iterations of this algorithm, it selects as next sampling point $(x^{m+1}, u^{m+1}) \in \mathcal{X} \times \mathcal{U}$, the point that minimizes in the worst conditions the largest bound width among the candidate optimal policies at the next iteration:

$$(x^{m+1}, u^{m+1}) \in \arg\min_{(x,u) \in \mathcal{X} \times \mathcal{U}} \left\{ \max_{(r,y) \in \mathbb{R} \times \mathcal{X} \, s.t. \, (x,u,r,y) \in \mathcal{C}(\mathcal{F}_m) \pi \in \Pi(\mathcal{F}_m \cup \{(x,u,r,y)\}, x_0)} \delta^{\pi}(\mathcal{F}_m \cup \{(x, u, r, y)\}, x_0) \right\}$$

where

$$\delta^{\pi}(\mathcal{F}, x_0) = U^{\pi}(\mathcal{F}, x_0) - L^{\pi}(\mathcal{F}, x_0).$$

Based on the convergence properties of the bounds, we conjecture that the sequence $(\Pi(\mathcal{F}_m, x_0))_{m \in \mathbb{N}}$ converges towards the set of all optimal policies in a finite number of iterations:

$$\exists m_0 \in \mathbb{N}\backslash\{0\} : \forall m \in \mathbb{N}, (m \geq m_0) \Rightarrow \Pi(\mathcal{F}_m, x_0) = \mathcal{J}^*(x_0).$$

The analysis of the theoretical properties of the sampling strategy and its empirical validation are left for future work.

• Illustration: In order to illustrate how the bound-based sampling strategy detailed above allows to discriminate among policies, we consider the following toy problem. The actual dynamics and reward functions are given by:

$$f(x,u)=x+u,$$
$$\rho(x,u)=x+u.$$

The state space is included in $\mathbb{R}$. The action space is set to

$$\mathcal{U}=\{-0.20,-0.10,0,+0.10,+0.20\}.$$

We consider a time horizon $T=3$, which induces $5^3=125$ different policies. The initial state is set to $x_0=-0.65$. Consequently, there is only one optimal policy, which consists in applying action +0.20 three times.

In the following experiments, the $\underset{(x,u)\in\mathcal{X}\times\mathcal{U}}{\arg\min}$ and $\max_{(r,y)\in\mathbb{R}\times\mathcal{X}\ s.t.(x,u,r,y)\in\mathcal{C}(\mathcal{F}_m)}$ operators, whose computation is of huge complexity, are approximated using purely random search algorithms (i.e. by randomly generating feasible points and taking the optimum over those points). We begin with a small sample of $n=5$ transitions (one for each action)

$$\{(0,-0.20,\rho(0,-0.20),f(0,-0.20)),(0,-0.10,\rho(0,-0.10),f(0,-0.10)),(0,0,\rho(0,0),f(0,0)),(0,0.10,\rho(0,0.10),f(0,0.10)),(0,0.20,\rho(0,0.20),f(0,0.20))\}$$

and iteratively augment it using our bound-based sampling strategy. We compare our strategy with a uniform sampling strategy (starting from the same initial sample of 5 transitions). We plot in Figure 13 the evolution of the empirical average number of candidate optimal policies (over 50 runs) with respect to the cardinality of the generated sample of transitions [2]. We empirically observe that the bound-based sampling strategy allows to discriminate policies faster than the uniform sampling strategy. In particular, we observe that, on average, bound-based strategy using 40 samples provides discriminating performances that are equivalent to those of the uniform sampling strategy using 80 samples, which represents a significant increase. Note that in this specific benchmark, one should sample 5 + 25 + 125 = 155 state-action pairs (by trying all possible policies) in order to be sure to discriminate all non-optimal policies.

[2]We have chosen to represent the average results obtained over 50 runs for both sampling methods rather the results obtained over one single run since (i) the variance of the results obtained by uniform sampling is high and (ii) the variance of the results obtained by the bound-based approach is also significant since the procedures for approximating the $\underset{(x,u)\in\mathcal{X}\times\mathcal{U}}{\arg\min}$ and $\max_{(r,y)\in\mathbb{R}\times\mathcal{X}\ s.t.(x,u,r,y)\in\mathcal{C}(\mathcal{F}_m)}$ operators rely on a random number generator.

4.6 Summary

We synthesize in Figure 14 the different settings and the corresponding results that have been presented in Section 4. Such results are classified in two main categories: stochastic setting and deterministic setting. Among each setting, we explicit the context (continuous/ finite action space) and the nature of each result (theoretical result, algorithmic contribution, empirical evaluation) using a color code.

5 Towards a New Paradigm for Batch Mode RL

In this concluding section, we highlight some connexions between the approaches based on synthesizing artificial trajectories and a more standard batch mode RL algorithm, the FQI algorithm [13] when it is used for policy evaluation. From a technical point of view, we consider again in this section the stochastic setting that was formalized in Section 3. The action space $\mathcal{U}$ is continuous and normed, and we consider a given closed-loop, time varying, Lipschitz continuous control policy h: $\{0, \ldots, T-1\} \times \mathcal{X} \to \mathcal{U}$.

5.1 Fitted Q Iteration for Policy Evaluation

The finite horizon FQI iteration algorithm for policy evaluation (FQI-PE) works by recursively computing a sequence of functions $\left(\widehat{Q}^h_{T-t}(.,.)\right)_{t=0}^{T-1}$ as follows:

Definition 20 (FQI-PE Algorithm)—• $\forall (x, u) \in \mathcal{X} \times \mathcal{U}$.

$$\widehat{Q}^h_0(x,u)=0 \qquad \forall (x,u) \in \mathcal{X} \times \mathcal{U},$$

• For $t = T-1 \ldots 0$, build the dataset $D=\left\{\left(i^l, o^l\right)\right\}_{l=1}^{n}$:

$$i^l=(x^l, u^l)$$
$$o^l=r^l+\widehat{Q}^h_{T-t-1}(y^l, h(t+1, y^l))$$

and use a regression algorithm $\mathcal{RA}$ to infer from D the function $\widehat{Q}^h_{T-t}$:

$$\widehat{Q}^h_{T-t}=\mathcal{RA}(D).$$

The FQI -PE estimator of the policy h is given by:

Definition 21 (FQI Estimator)

$$\widehat{J}^h_{FQI}(\mathcal{F}_n, x_0)=\widehat{Q}^h_T(x_0, h(0, x_0)).$$

5.2 FQI using *k*–Nearest Neighbor Regressors: an Artificial Trajectory Viewpoint

We propose in this section to use a k–Nearest Neighbor algorithm (k–NN) as regression algorithm $\mathcal{RA}$. In the following, for a given state action couple $(x, u) \in \mathcal{X} \times \mathcal{U}$, we denote by $l_i(x, u)$ the lowest index in $\mathcal{F}_n$ of the i-th nearest one step transition from the state-action

couple (x, u) using the distance measure Δ. Using this notation, the k–NN based FQI-PE algorithm for estimating the expected return of the policy h works as follows:

Definition 22 (k–*NN* FQI-PE Algorithm)—• $\forall (x, u) \in \mathcal{X} \times \mathcal{U}$,

$$\widehat{Q}_0^h(x,u)=0,$$

• For $t = T - 1 \ldots 0$, $\forall (x, u) \in \mathcal{X} \times \mathcal{U}$,

$$\widehat{Q}_{T-t}^h(x,u)=\frac{1}{k}\sum_{i=1}^{k}\left(r^{l_i(x,u)}+\widehat{Q}_{T-t-1}^h\left(y^{l_i(x,u)},h\left(t+1,y^{l_i(x,u)}\right)\right)\right).$$

The k–NN FQI-PE estimator of the policy h is given by:

$$\widehat{J}_{FQI}^h(\mathcal{F}_n,x_0)=\widehat{Q}_T^h(x_0,h(0,x_0)).$$

One can observe that, for a fixed initial state x_0, the computation of the k–NN FQI-PE estimator of h works by identifying $(k + k^2 + \ldots + k^T)$ non-unique one-step transitions. These transitions are non-unique in the sense that some transitions can be selected several times during the process. In order to concisely denote the indexes of the one-step system transitions that are selected during the k–NN FQI-PE algorithm, we introduce the notation $l^{i_0,i_1,\ldots i_t}$ for refering to the transition $l_{i_t}(y^{l^{i_0,\ldots i_{t-1}}}, h(t, y^{l^{i_0,\ldots i_{t-1}}}))$ for $i_0, \ldots i_t \in \{1, \ldots k\}$, $t \geq 1$ with $l^{i_0} = l_{i_0}(x_0, h(0, x_0))$. Using these notations, we illustrate the computation of the k–NN FQI-PE Estimator in Figure 15. Then, we have the following result:

Proposition 6 (k–NN FQI-PE using Artificial Trajectories)

$$\widehat{J}_{FQI}^h(\mathcal{F}_n,x_0)=\frac{1}{K^T}\sum_{i_0=1}^{k}\cdots\sum_{i_{T-1}=1}^{k}\left(r^{l^{i_0}}+r^{l^{i_0,i_1}}+\ldots+r^{l^{i_0,i_1,\ldots,i_{T-1}}}\right).$$

where the set of rebuilt artificial trajectories

$$\left\{\left[\left(x^{l^{i_0}},u^{l^{i_0}},r^{l^{i_0}},y^{l^{i_0}}\right),\ldots,\left(x^{l^{i_0,\ldots,i_{T-1}}},u^{l^{i_0,\ldots,i_{T-1}}},r^{l^{i_0,\ldots,i_{T-1}}},y^{l^{i_0,\ldots,i_{T-1}}}\right)\right]\right\}$$

is such that $\forall t \in \{0, \ldots, T - 1\}$, $\forall (i_0, \ldots, i_t) \in \{1, \ldots, k\}^{t+1}$,

$$\Delta\left(\left(y^{l^{i_0,\ldots,i_{t-1}}},h\left(t,y^{l^{i_0,\ldots,i_{t-1}}}\right)\right),\left(x^{l^{i_0,\ldots,i_t}},u^{l^{i_0,\ldots,i_t}}\right)\right)\leq\alpha_k(\mathcal{P}_n).$$

Proof We propose to prove by induction the property

$$\mathcal{H}_t: \widehat{J}^h_{FQI}(\mathcal{F}_n, x_0) = \frac{1}{k^t} \sum_{i_0=1}^{k} \dots \sum_{i_{t-1}=1}^{k} \left(r^{l^{i_0}} + \dots + r^{l^{i_0,\dots,i_{t-1}}} + \widehat{Q}^h_{T-t}\left(y^{l^{i_0,\dots,i_{t-1}}}, h\left(t, y^{l^{i_0,\dots,i_{t-1}}}\right)\right)\right).$$

Basis: According to the definition of the *k*–NN estimator, we have:

$$\widehat{J}^h_{FQI}(\mathcal{F}_n, x_0) = \frac{1}{k} \sum_{i_0=1}^{k} \left(r^{l^{i_0}} + \widehat{Q}^h_{T-1}\left(y^{l^{i_0}}, h\left(1, y^{l^{i_0}}\right)\right)\right),$$

which proves $\mathcal{H}_1$.

Induction step: Let us assume that $\mathcal{H}_t$ is true for $t \in \{1, \dots, T-1\}$. Then, we have:

$$\widehat{J}^h_{FQI}(\mathcal{F}_n, x_0) = \frac{1}{k^t} \sum_{i_0=1}^{k} \dots \sum_{i_{t-1}=1}^{k} \left(r^{l^{i_0}} + \dots + r^{l^{i_0,\dots,i_{t-1}}} + \widehat{Q}^h_{T-t}\left(y^{l^{i_0,\dots,i_{t-1}}}, h\left(t, y^{l^{i_0,\dots,i_{t-1}}}\right)\right)\right). \quad (4)$$

According to the definition of the *k*–NN FQI-PE algorithm, we have:

$$\widehat{Q}^h_{T-t}\left(y^{l^{i_0,\dots,i_{t-1}}}, h\left(t, y^{l^{i_0,\dots,i_{t-1}}}\right)\right) = \frac{1}{k} \sum_{i_t=1}^{k} \left(r^{l^{i_0,\dots,i_{t-1},i_t}} + \widehat{Q}^h_{T-t-1}\left(y^{l^{i_0,\dots,i_{t-1},i_t}}, h\left(t+1, y^{l^{i_0,\dots,i_{t-1},i_t}}\right)\right)\right) \quad (5)$$

Equations (4) and (5) give

$$
\begin{aligned}
&\widehat{J}^{h}_{FQI}(\mathscr{F}_n, x_0)\\
&\quad=\frac{1}{k^t}\sum_{i_0=1}^{k}\cdots\sum_{i_{t-1}=1}^{k}\Big(r^{l^{i_0}}\\
&\quad+\ldots+r^{l^{i_0,\ldots,i_{t-1}}}\\
&\quad+\frac{1}{k}\sum_{i_t=1}^{k}(r^{l^{i_0,\ldots,i_t}}\\
&\quad+\widehat{Q}^{h}_{T-t-1}(y^{l^{i_0,\ldots,i_t}}, h(t\\
&\qquad+1, y^{l^{i_0,\ldots,i_t}})))\Big)\\
&\quad=\frac{1}{k^t}\sum_{i_0=1}^{k}\cdots\sum_{i_{t-1}=1}^{k}\left(\frac{1}{k}\sum_{i_t=1}^{k}(r^{l^{i_0}}\right.\\
&\quad+\ldots+r^{l^{i_0,\ldots,i_{t-1}}})+\frac{1}{k}\sum_{i_t=1}^{k}(r^{l^{i_0,\ldots,i_t}}\\
&\quad+\widehat{Q}^{h}_{T-t-1}(y^{l^{i_0,\ldots,i_t}}, h(t\\
&\qquad\left.+1, y^{l^{i_0,\ldots,i_t}})))\right)\\
&\quad=\frac{1}{k^{t+1}}\sum_{i_0=1}^{k}\cdots\sum_{i_{t-1}=1}^{k}\sum_{i_t=1}^{k}\Big(r^{l^{i_0}}\\
&\quad+\ldots+r^{l^{i_0,\ldots,i_{t-1}}}\\
&\quad+r^{l^{i_0,\ldots,i_t}}\\
&\quad+\widehat{Q}^{h}_{T-t-1}(y^{l^{i_0,\ldots,i_t}}, h(t\\
&\qquad+1, y^{l^{i_0,\ldots,i_t}}))\Big),
\end{aligned}
$$

which proves $\mathcal{H}_{t+1}$. The proof is completed by observing that

$$\widehat{Q}^{h}_{0}(x,u)=0, \qquad \forall (x,u)\in\mathscr{X}\times\mathscr{U}$$

and by observing that the property

$$\Delta\left(\left(y^{l^{i_0,\ldots,i_{t-1}}}, h\left(t, y^{l^{i_0,\ldots,i_{t-1}}}\right)\right), \left(x^{l^{i_0,\ldots,i_t}}, u^{l^{i_0,\ldots,i_t}}\right)\right)\leq\alpha_k(\mathscr{P}_n).$$

directly comes from the use of k–NN function approximators.

The previous result shows that the estimate of the expected return of the policy h computed by the k–NN FQI-PE algorithm is the average of the return of k^T artificial trajectories. These artificial trajectories are built from $(k + k^2 + \ldots + k^T)$ *non-unique* one-step system transitions from $\mathcal{F}_n$ that are also chosen by minimizing the distance between two successive one-step transitions.

• **Illustration.** We empirically compare the MFMC estimator with the k–NN FQI-PE estimator on the toy problem presented in Section 4.2, but with a smaller time horizon $T = 5$. For a fixed cardinality $n = 100$, we consider all possible values of the parameter k ($k \in \{1, \ldots, 100\}$ since there are at most n nearest neighbours) and p ($p \in \{1,\ldots, 20\}$ since one can generate at most n/T different artificial trajectories without re-using transitions). For each value of p (resp. k), we generate 1000 samples of transitions using a uniform random distribution over the state action space. For each sample, we run the MFMC (resp. the k–NN FQI-PE estimator). As a baseline comparison, we also compute a 1000 runs of the MC estimator for every value of p. Figure 16 (resp. 17 and 18) reports the obtained empirical average (resp. variance and mean squared error).

We observe in Figure 16 that (i) the MFMC estimator with $p \in \{1,\ldots, 3\}$ is less biased than the k–NN FQI-PE estimator with any value of $k \in \{1,\ldots, 100\}$ and (ii) the bias of the MFMC estimator increases faster (with respect to p) than the bias of the k–NN FQI-PE estimator (with respect to k). The increase of the bias of the MFMC estimator with respect to p is suggested by Theorem 1, where an upper bound on the bias that increases with p is provided. This phenomenon seems to affect the k–NN FQI-PE estimator (with respect to k) to a lesser extent. In Figure 17, we observe that the k–NN FQI-PE estimator has a variance that is higher than that of the MFMC estimator for any $k = p$. This may be explained by the fact that for samples of $n = 100$ transitions, one-step transitions are often re-used by the k–NN FQI-PE estimator, which generates dependence between artificial trajectories. We finally plot in Figure 18 the observed empirical mean squared error (sum of the squared empirical bias and empirical variance) and observe that in our specific setting, the MFMC estimator offers for values of $p \in \{1,2,3,4\}$ a better bias versus variance compromise than the k–NN FQI-PE estimator with any value of k.

5.3 Kernel-based and Other Averaging type Regression Algorithms

The results exposed in Section 5.2 can be extended to the case where the FQI-PE algorithm is combined with kernel-based regressors and in particular tree-based regressors. In such a context, the sequence of functions $(\widehat{Q}^h_{T-t}(.))_{t=0}^{T}$ is computed as follows:

Definition 23 (KB FQI-PE Algorithm)—$\forall (x, u) \in \mathcal{X} \times \mathcal{U}$,

$$\widehat{Q}^h_0(x,u)=0,$$

For $t = T-1 \ldots 0$, $\forall (x, u) \in \mathcal{X} \times \mathcal{U}$,

$$\widehat{Q}^h_{T-t}(x,u)=\sum_{l=1}^{n} k_{\mathcal{F}_n}((x,u),(x^l,u^l))\left(r^l+\widehat{Q}^h_{T-t-1}(y^l,h(t+1,y^l))\right),$$

with

$$k_{\mathcal{F}_n}((x,u),(x^l,u^l))=\frac{\Phi\left(\frac{\Delta((x,u),(x^l,u^l))}{b_n}\right)}{\sum_{i=1}^{n}\Phi\left(\frac{\Delta((x,u),(x^i,u^i))}{b_n}\right)},$$

where $\Phi: [0,1] \rightarrow \mathbb{R}^+$ is a univariate non-negative "mother kernel" function, and $b_n > 0$ is the bandwidth parameter. We also suppose that $\int_0^1 \Phi(z)dz = 1$ and $\Phi(x) = 0 \ \forall x > 1$.

The KB estimator of the expected return of the policy h is given by:

$$\widehat{J}^h_{FQI}(\mathcal{F}_n, x_0) = \widehat{Q}^h_T(x_0, h(0, x_0)).$$

Given an initial state $x_0 \in \mathcal{X}$, the computation of the KB FQI-PE algorithm can also be interpreted as an identification of a set of one-step transitions from $\mathcal{F}_n$. At each time step t, all the one-step transitions (x^l, u^l, r^l, y^l) that are not farther than a distance b_n from $(x_t, h(t, x_t))$ are selected and weighted with a distance dependent factor. Other one-step transitions are weighted with a factor equal to zero. This process is iterated with the output of each selected one-step transitions. An illustration is given in Figure 19 . The value returned by the KB estimator can be expressed as follows:

Proposition 7

$$\widehat{J}^h_{FQI}(\mathcal{F}_n, x_0) = \sum_{i_0=1}^{n} \dots \sum_{i_{T-1}=1}^{n} \theta_0^{0,i_0} \theta_1^{i_0,i_1} \dots \theta_{T-1}^{i_{T-2},i_{T-1}} (r^{i_0} + \dots + r^{i_{T-1}})$$

with

$$\theta_0^{0,i_0} = k_{\mathcal{F}_n}\left((x_0, h(0, x_0)), (x^{i_0}, u^{i_0})\right),$$
$$\theta_{t+1}^{i_t,i_{t+1}} = k_{\mathcal{F}_n}\left((y^{i_t}, h(t+1, y^{i_t})), (x^{i_{t+1}}, u^{i_{t+1}})\right), \forall t \in \{0, \dots, T-2\}.$$

Proof We propose to prove by induction the property

$$\mathcal{H}_t: \widehat{J}^h_{FQI}(\mathcal{F}_n, x_0) = \left(\sum_{i_0=1}^{n} \dots \sum_{i_t=1}^{n} \theta^{0,i_0} \theta^{i_0,i_1} \dots \theta^{i_{t-1},i_t} \left(r^{i_0} + \dots + r^{i_{t-1}} + \widehat{Q}^h_{T-t}(y^{i_{t-1}}, h(t, y^{i_{t-1}}))\right)\right).$$

Basis: One has

$$\widehat{J}^h_{FQI}(\mathcal{F}_n, x_0) = \sum_{i_0=1}^{n} \theta^{0,i_0}(r^{i_0} + \widehat{Q}^h_{T-t}(y^{i_0}, h(1, y^{i_0}))),$$

which proves $\mathcal{H}_1$.

Induction step: Let us assume that $\mathcal{H}_t$ is true for $t \in \{1, \dots, T-1\}$. Then, one has

$$\widehat{J}^h_{FQI}(\mathscr{F}_n, x_0)=\left(\sum_{i_0=1}^{n}\ldots\sum_{i_t=1}^{n}\theta^{0,i_0}\theta^{i_0,i_1}\ldots\theta^{i_{t-2},i_{t-1}}\left(r^{i_0}+\ldots+r^{i_{t-1}}+\widehat{Q}^h_{T-t}(y^{i_{t-1}},h(t,y^{i_{t-1}}))\right)\right). \quad (6)$$

According to the KB value iteration algorithm, we have:

$$\widehat{Q}^h_{T-t}(y^{i_{t-1}},h(t,y^{i_{t-1}}))=\sum_{i_{t-1}=1}^{k}\theta^{i_{t-1},i_t}(r^{i_t}+\widehat{Q}^h_{T-t-1}(y^{i_t},h(t+1,y^{i_t}))). \quad (7)$$

Equations (6) and (7) give

$$\widehat{J}^h_{FQi}(\mathscr{F}_n, x_0)=\sum_{i_0=1}^{n}\ldots\sum_{i_{t-1}=1}^{n}\theta^{0,i_0}\ldots\theta^{i_{t-2},i_{t-1}}\times\left(r^{i_0}+\ldots+r^{i_{t-1}}+\sum_{i_t=1}^{n}\theta^{i_{t-1},i_t}(r^{i_t}+\widehat{Q}^h_{T-t-1}(y^{i_t},h(t+1,y^{i_t})))\right)$$

Since $\sum_{i_t=1}^{n}\theta^{i_{t-1},i_t}=1$, one has

$$\begin{aligned}
&\widehat{J}^h_{FQI}(\mathscr{F}_n, x_0)\\
&\quad=\sum_{i_0=1}^{n}\ldots\sum_{i_{t-1}=1}^{n}\theta^{0,i_0}\ldots\theta^{i_{t-2},i_{t-1}}\\
&\quad\times\left((\sum_{i_t=1}^{n}\theta^{i_{t-1},i_t})(r^{i_0}\right.\\
&\quad+\ldots+r^{i_{t-1}})+\sum_{i_t=1}^{n}\theta^{i_{t-1},i_t}(r^{i_t}\\
&\quad+\widehat{Q}^h_{T-t-1}(y^{i_t},h(t\\
&\qquad\left.+1,y^{i_t}))\right)\\
&\quad=\sum_{i_0=1}^{n}\ldots\sum_{i_{t-1}=1}^{n}\theta^{0,i_0}\ldots\theta^{i_{t-2},i_{t-1}}\sum_{i_t=1}^{n}\theta^{i_{t-1},i_t}\\
&\quad\times\left(r^{i_0}+\ldots+r^{i_{t-1}}+r^{i_t}+\widehat{Q}^h_{T-t-1}(y^{i_t},h(t\right.\\
&\qquad\left.+1,y^{i_t}))\right)
\end{aligned}$$

which proves $\mathcal{H}_{t+1}$. The proof is completed by observing that $\widehat{Q}^h_0(x,u)=0, \forall(x,u)\in\mathscr{X}\times\mathscr{U}$.

One can observe through Proposition 7 that the computation of the KB estimate of the expected return of the policy h can be expressed in the form of a weighted sum of the return of n^T artificial trajectories. Each artificial trajectory

$$[(x^{i_0},u^{i_0},r^{i_0},y^{i_0}),(x^{i_1},u^{i_1},r^{i_1},y^{i_1}),\ldots,(x^{i_{T-1}},u^{i_{T-1}},r^{i_{T-1}},y^{i_{T-1}})]$$

is weighted with a factor $\theta_0^{0,i_0}\theta_1^{i_0,i_1}\dots\theta_{T-1}^{i_{T-2},i_{T-1}}$. Note that some of these factors can eventually be equal to zero. Similarly to the k–NN estimator, these artificial trajectories are also built from the $T\times n^T$ *non-unique* one-step system transitions from $\mathcal{F}_n$.

More generally, we believe that the notion of artificial trajectory could also be used to characterize other batch mode RL algorithms that rely on other kinds of “averaging” schemes [24].

6 Conclusion

In this paper we have revisited recent works based on the idea of synthesizing artificial trajectories in the context of batch mode reinforcement learning problems. This paradigm shows to be of value in order to construct novel algorithms and performance analysis techniques. We think that it is of interest to revisit in this light the existing batch mode reinforcement algorithms based on function approximators in order to analyze their behavior and possibly create new variants presenting interesting performance guarantees.

Acknowledgments

Raphael Fonteneau is a Post-doctoral Fellow of the F.R.S.-FNRS. This paper presents research results of the European excellence network PASCAL2 and of the Belgian Network DYSCO, 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 scientific responsibility rests with its authors.

References

1. Antos A, Munos R, Szepesvári C. Fitted Q-iteration in continuous action space MDPs. Advances in Neural Information Processing Systems 20 (NIPS). 2007
2. Bellman, R. Dynamic Programming. Princeton University Press; 1957.
3. Bonarini A, Caccia C, Lazaric A, Restelli M. Batch reinforcement learning for controlling a mobile wheeled pendulum robot. Artificial Intelligence in Theory and Practice II. 2008:151–160.
4. Boyan J. Technical update: Least-squares temporal difference learning. Machine Learning. 2005; 49:233–246.
5. Boyan, J.; Moore, A. Advances in Neural Information Processing Systems 7 (NIPS). MIT Press; Denver, CO, USA: 1995. Generalization in reinforcement learning: Safely approximating the value function; p. 369-376.
6. Bradtke S, Barto A. Linear least-squares algorithms for temporal difference learning. Machine Learning. 1996; 22:33–57.
7. Busoniu, L.; Babuska, R.; De Schutter, B.; Ernst, D. Reinforcement Learning and Dynamic Programming using Function Approximators. Taylor & Francis CRC Press; 2010.
8. Castelletti A, Galelli S, Restelli M, Soncini-Sessa R. Tree-based reinforcement learning for optimal water reservoir operation. Water Resources Research. 2010; 46
9. Castelletti A, de Rigo D, Rizzoli A, Soncini-Sessa R, Weber E. Neuro-dynamic programming for designing water reservoir network management policies. Control Engineering Practice. 2007; 15(8): 1031–1038.
10. Chakraborty, B.; Strecher, V.; Murphy, S. Workshop on Model Uncertainty and Risk in Reinforcement Learning. NIPS; Whistler, Canada: 2008. Bias correction and confidence intervals for fitted Q-iteration.
11. Defourny, B.; Ernst, D.; Wehenkel, L. Workshop on Model Uncertainty and Risk in Reinforcement Learning. NIPS; Whistler, Canada: 2008. Risk-aware decision making and dynamic programming.
12. Ernst D, Geurts P, Wehenkel L. Iteratively extending time horizon reinforcement learning. European Conference on Machine Learning (ECML). 2003:96–107.
13. Ernst D, Geurts P, Wehenkel L. Tree-based batch mode reinforcement learning. Journal of Machine Learning Research. 2005; 6:503–556.

14. Ernst D, Glavic M, Capitanescu F, Wehenkel L. Reinforcement learning versus model predictive control: a comparison on a power system problem. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics. 2009; 39:517–529.
15. Ernst D, Marée R, Wehenkel L. Reinforcement learning with raw image pixels as state input. International Workshop on Intelligent Computing in Pattern Analysis/Synthesis (IWICPAS) Proceedings series: Lecture Notes in Computer Science. 2006; 4153:446–454.
16. Ernst D, Stan G, Goncalves J, Wehenkel L. Clinical data based optimal STI strategies for HIV: a reinforcement learning approach. Machine Learning Conference of Belgium and The Netherlands (BeNeLearn). 2006:65–72.
17. Farahmand, A.; Ghavamzadeh, M.; Szepesvri, C.; Mannor, S. Regularized fitted q-iteration: Application to planning. In: Girgin, S.; Loth, M.; Munos, R.; Preux, P.; Ryabko, D., editors. Recent Advances in Reinforcement Learning, *Lecture Notes in Computer Science*. Vol. 5323. Springer Berlin; Heidelberg: 2008. p. 55-68.
18. Fonteneau, R. Ph D thesis. University of Liége; 2011. Contributions to Batch Mode Reinforcement Learning.
19. Fonteneau, R.; Murphy, S.; Wehenkel, L.; Ernst, D. IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL). Nashville, TN, USA: 2009. Inferring bounds on the performance of a control policy from a sample of trajectories.
20. Fonteneau, R.; Murphy, S.; Wehenkel, L.; Ernst, D. A cautious approach to generalization in reinforcement learning. Second International Conference on Agents and Artificial Intelligence (ICAART); Valencia, Spain. 2010;
21. Fonteneau R, Murphy S, Wehenkel L, Ernst D. Generating informative trajectories by using bounds on the return of control policies. Workshop on Active Learning and Experimental Design 2010 (in conjunction with AISTATS 2010). 2010
22. Fonteneau, R.; Murphy, S.; Wehenkel, L.; Ernst, D. Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS), JMLR: W&CP 9. Chia Laguna, Sardinia, Italy: 2010. Model-free Monte Carlo–like policy evaluation; p. 217-224.
23. Fonteneau, R.; Murphy, SA.; Wehenkel, L.; Ernst, D. Towards min max generalization in reinforcement learning. Revised Selected Papers Series: Communications in Computer and Information Science (CCIS); Agents and Artificial Intelligence: International Conference, ICAART 2010; Valencia, Spain. January 2010; Heidelberg: Springer; 2011. p. 61-77.
24. Gordon G. Stable function approximation in dynamic programming. Twelfth International Conference on Machine Learning (ICML). 1995:261–268.
25. Gordon, G. Ph D thesis. Carnegie Mellon University; 1999. Approximate solutions to markov decision processes.
26. Guez A, Vincent R, Avoli M, Pineau J. Adaptive treatment of epilepsy via batch-mode reinforcement learning. Innovative Applications of Artificial Intelligence (IAAI). 2008
27. Lagoudakis M, Parr R. Least-squares policy iteration. Jounal of Machine Learning Research. 2003; 4:1107–1149.
28. Lange, S.; Riedmiller, M. European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN). Brugge, Belgium: 2010. Deep learning of visual control policies.
29. Lazaric, A.; Ghavamzadeh, M.; Munos, R. Tech rep, SEQUEL (INRIA). Lille - Nord Europe: 2010. Finite-sample analysis of least-squares policy iteration.
30. Lazaric A, Ghavamzadeh M, Munos R. Finite-sample analysis of LSTD. International Conference on Machine Learning (ICML). 2010:615–622.
31. Morimura, T.; Sugiyama, M.; Kashima, H.; Hachiya, H.; Tanaka, T. Nonparametric return density estimation for reinforcement learning; 27th International Conference on Machine Learning (ICML); Haifa, Israel. Jun. 21-25 (2010);
32. Morimura, T.; Sugiyama, M.; Kashima, H.; Hachiya, H.; Tanaka, T. Parametric return density estimation for reinforcement learning; 26th Conference on Uncertainty in Artificial Intelligence (UAI); Catalina Island, California, USA. Jul. 8-11; 2010. p. 368-375.
33. Munos R, Szepesvári C. Finite-time bounds for fitted value iteration. Journal of Machine Learning Research. 2008:815–857.

34. Murphy S. Optimal dynamic treatment regimes. 2003; 65(2):331–366.Journal of the Royal Statistical Society, Series B
35. Murphy S, Van Der Laan M, Robins J. Marginal mean models for dynamic regimes. Journal of the American Statistical Association. 2001; 96(456):1410–1423. [PubMed: 20019887]
36. Nedi A, Bertsekas DP. Least squares policy evaluation algorithms with linear function approximation. Discrete Event Dynamic Systems. 2003; 13:79–110.10.1023/A:1022192903948
37. Ormoneit D, Sen S. Kernel-based reinforcement learning. Machine Learning. 2002; 49(2-3):161–178.
38. Peters, J.; Vijayakumar, S.; Schaal, S. Third IEEE-RAS International Conference on Humanoid Robots. Citeseer; 2003. Reinforcement learning for humanoid robotics; p. 1-20.
39. Pietquin, O.; Tango, F.; Aras, R. Computational Intelligence in Vehicles and Transportation Systems (CIVTS), 2011 IEEE Symposium on. IEEE; 2011. Batch reinforcement learning for optimizing longitudinal driving assistance strategies; p. 73-79.
40. Riedmiller, M. Sixteenth European Conference on Machine Learning (ECML). Porto, Portugal: 2005. Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method; p. 317-328.
41. Robins J. A new approach to causal inference in mortality studies with a sustained exposure period–application to control of the healthy worker survivor effect. Mathematical Modelling. 1986; 7(9-12):1393–1512.
42. Sutton, R. Advances in Neural Information Processing Systems 8 (NIPS). MIT Press; Denver, CO, USA: 1996. Generalization in reinforcement learning: Successful examples using sparse coding; p. 1038-1044.
43. Sutton, R.; Barto, A. Reinforcement Learning. MIT Press; 1998.
44. Timmer, S.; Riedmiller, M. IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL). IEEE; 2007. Fitted Q iteration with cmacs; p. 1-8.
45. Tognetti, S.; Savaresi, S.; Spelta, C.; Restelli, M. Control Applications (CCA) & Intelligent Control (ISIC), 2009. IEEE; 2009. Batch reinforcement learning for semi-active suspension control; p. 582-587.

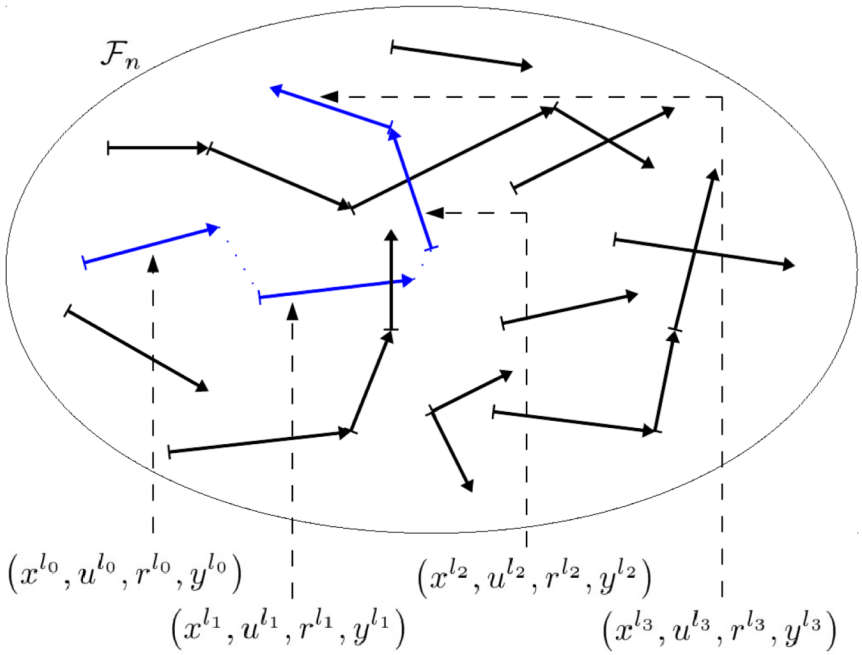


Fig. 1.
An example of an artificial trajectory rebuilt from 4 one-step system transitions from $\mathcal{F}_n$.

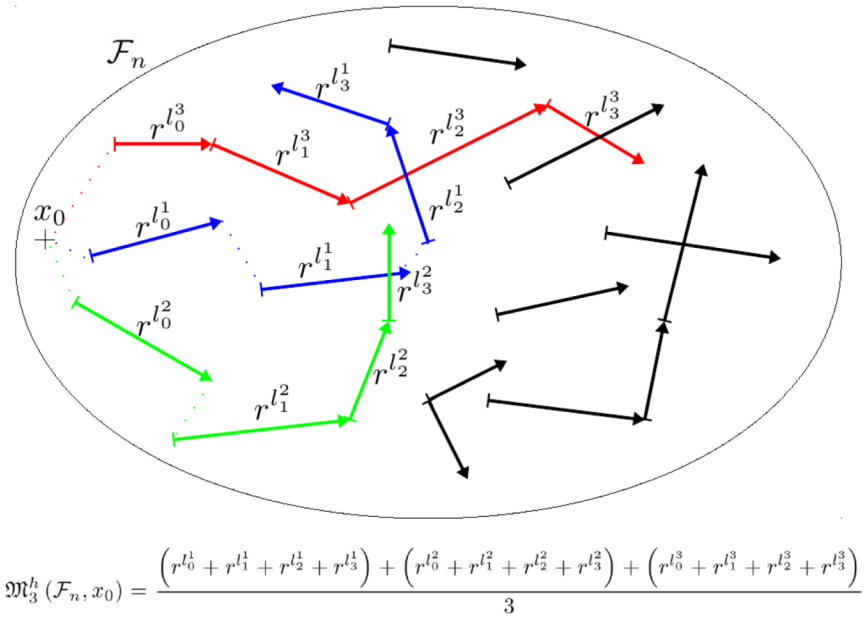


Fig. 2.
Rebuilding three 4-length trajectories for estimating the return of a policy.

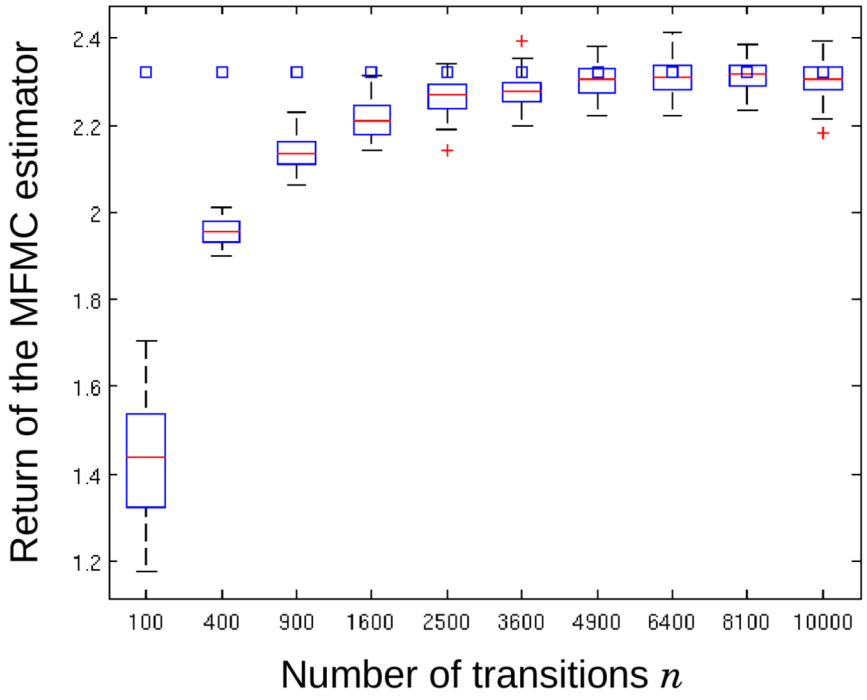


Fig. 3.

Computations of the MFMC estimator with $p = 10$, for different cardinalities n of the sample of one-step transitions. For each cardinality, 50 independent samples of transitions have generated. Squares represent $J^h(x_0)$.

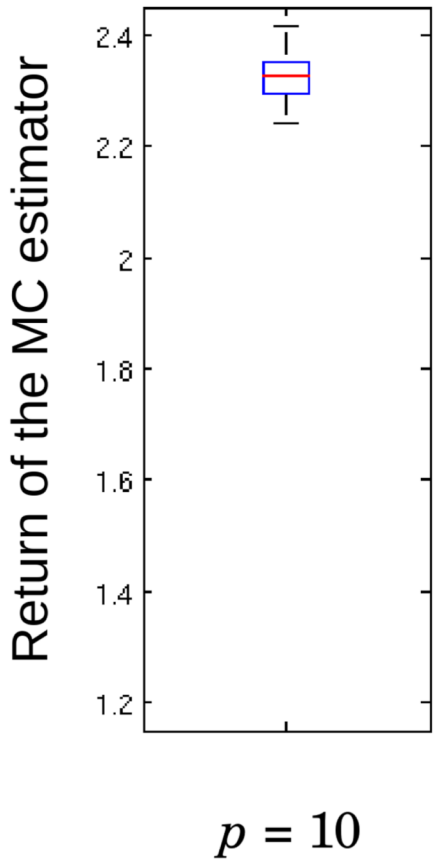


Fig. 4.
Computations of the MC estimator with $p = 10$. 50 independent runs have been computed.

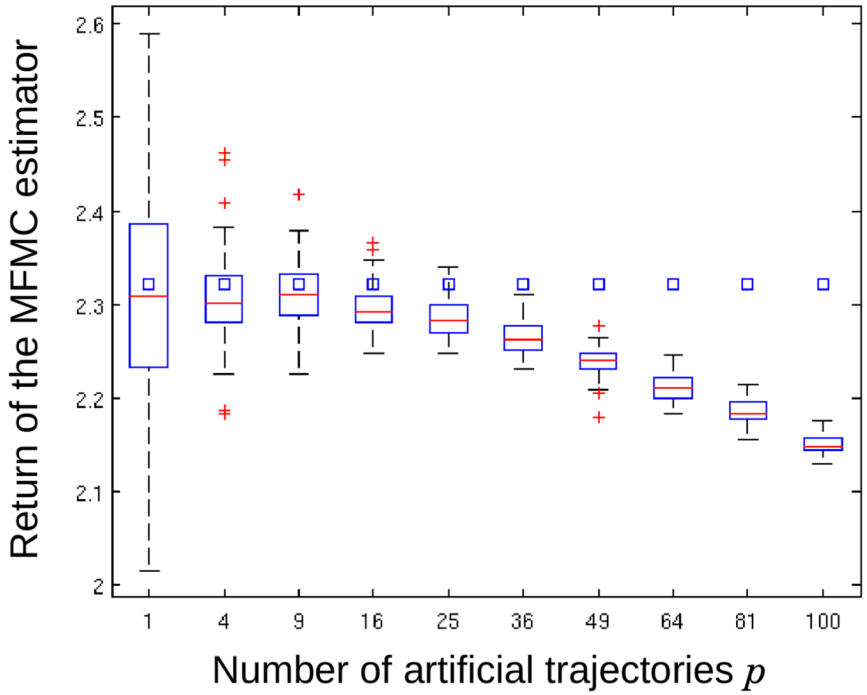


Fig. 5.

Computations of the MFMC estimator for different values of the number p of artificial trajectories extracted from a sample of $n = 10,000$ tuples. For each value of p, 50 independent samples of transitions have generated. Squares represent $J^h(x_0)$.

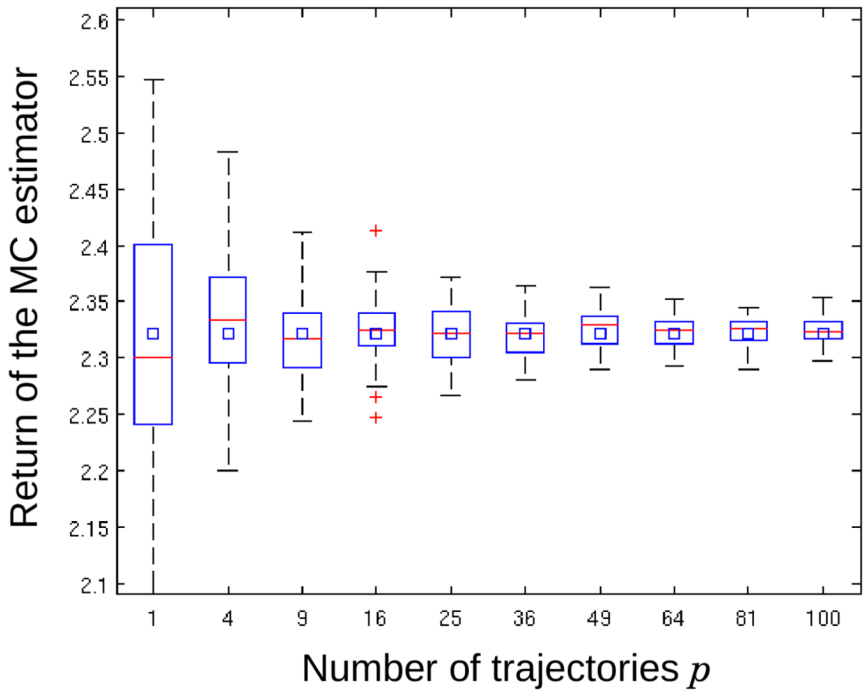
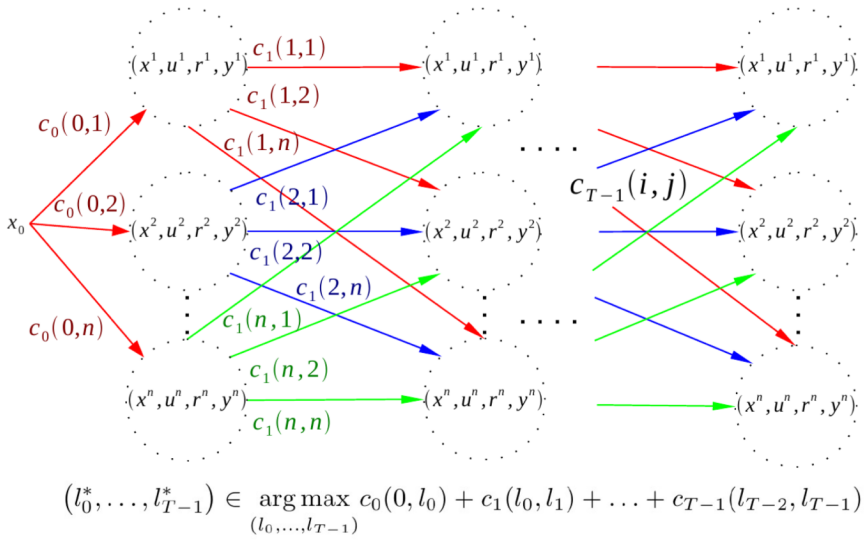


Fig. 6.

Computations of the MC estimator for different values of the number of trajectories p. For each value of p, 50 independent runs of the MC estimator have been computed. Squares represent $J^h(x_0)$.



$$(l_0^*,\ldots,l_{T-1}^*) \in \underset{(l_0,\ldots,l_{T-1})}{\arg\max}\; c_0(0,l_0) + c_1(l_0,l_1) + \ldots + c_{T-1}(l_{T-2},l_{T-1})$$

with

$$c_t(i,j) = -L'_{Q_{T-t}}\left\|y^i - x^j\right\|_{\mathcal{X}} + r^j, \qquad y^0 = x_0\ .$$

Fig. 7.

A graphical interpretation of the CGRL algorithm. The CGRL solution can be interpreted as a shortest path in a specific graph.

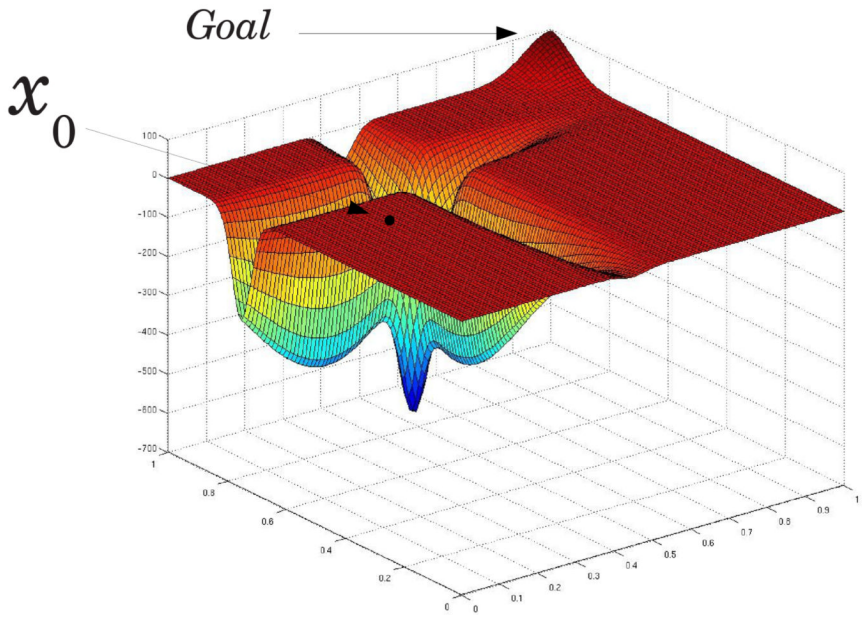


Fig. 8.
The Puddle World benchmark. Starting from x_0, an agent has to avoid the puddles and navigate towards the goal.

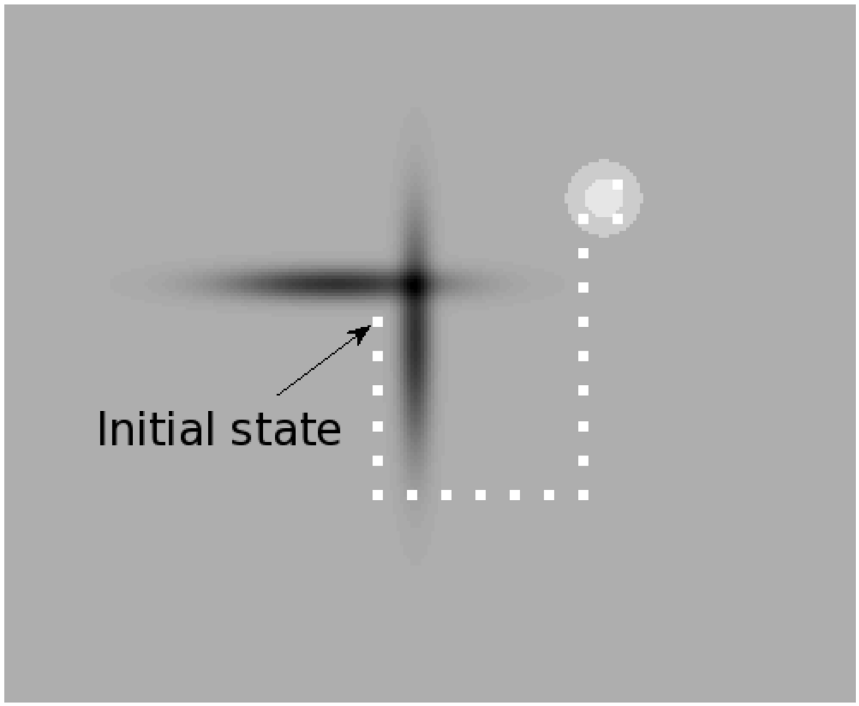


Fig. 9.
CGRL with $\mathcal{F}$.

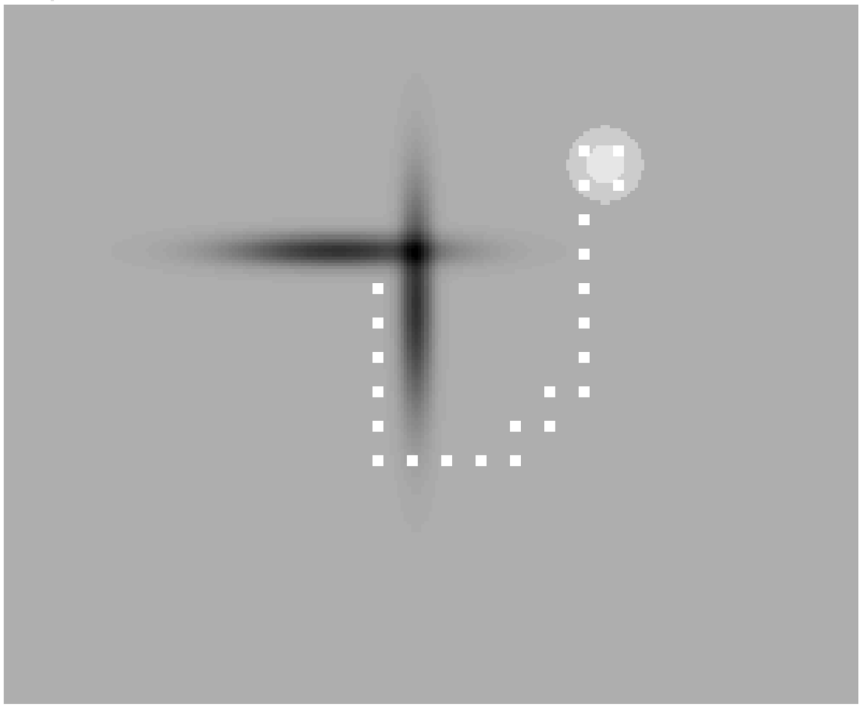


Fig. 10.
FQI with $\mathcal{F}$.

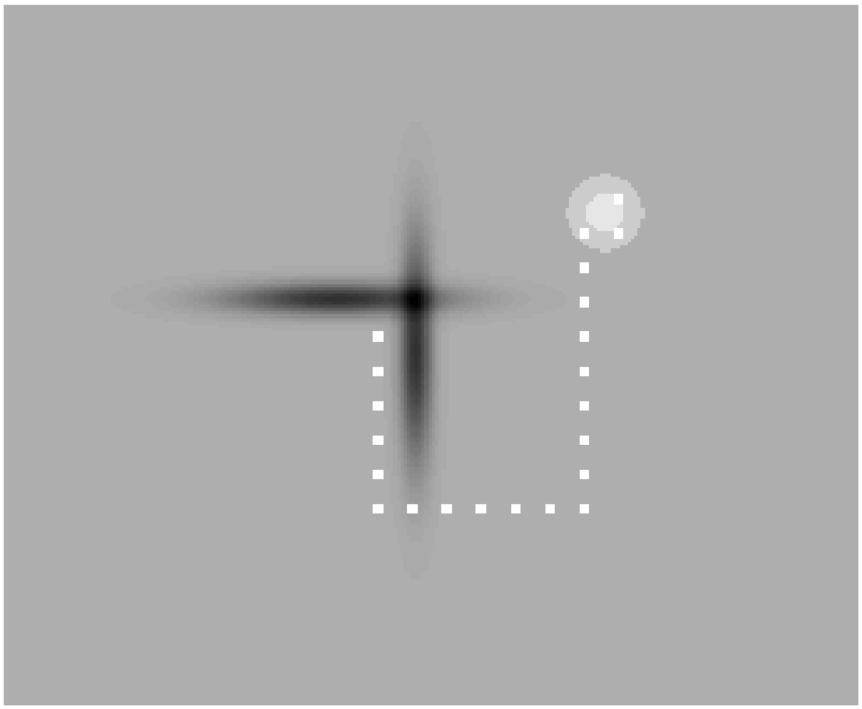


Fig. 11.
CGRL with $\mathcal{F}'$.

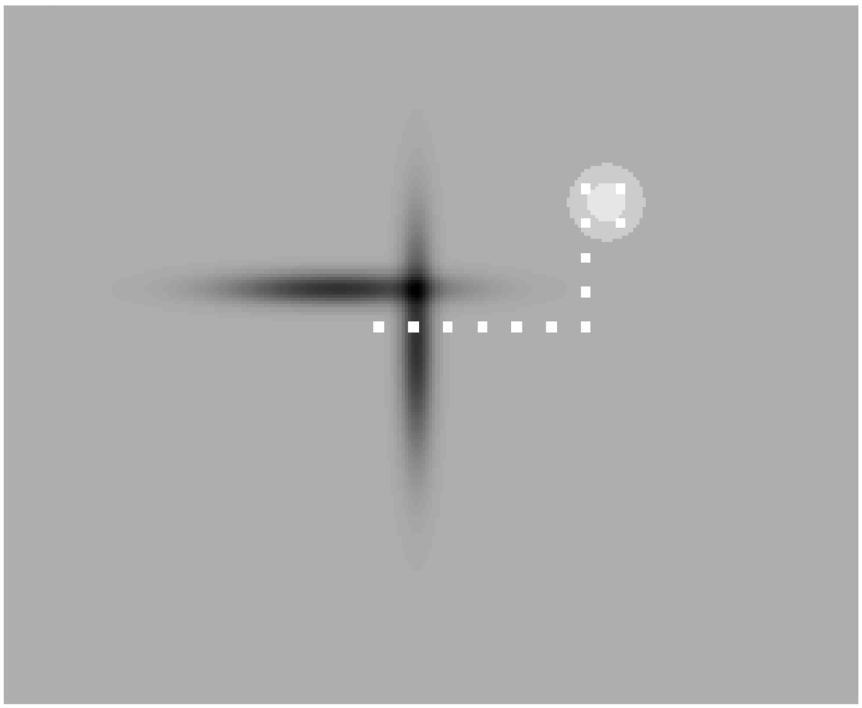


Fig. 12.
FQI with $\mathcal{F}'$.

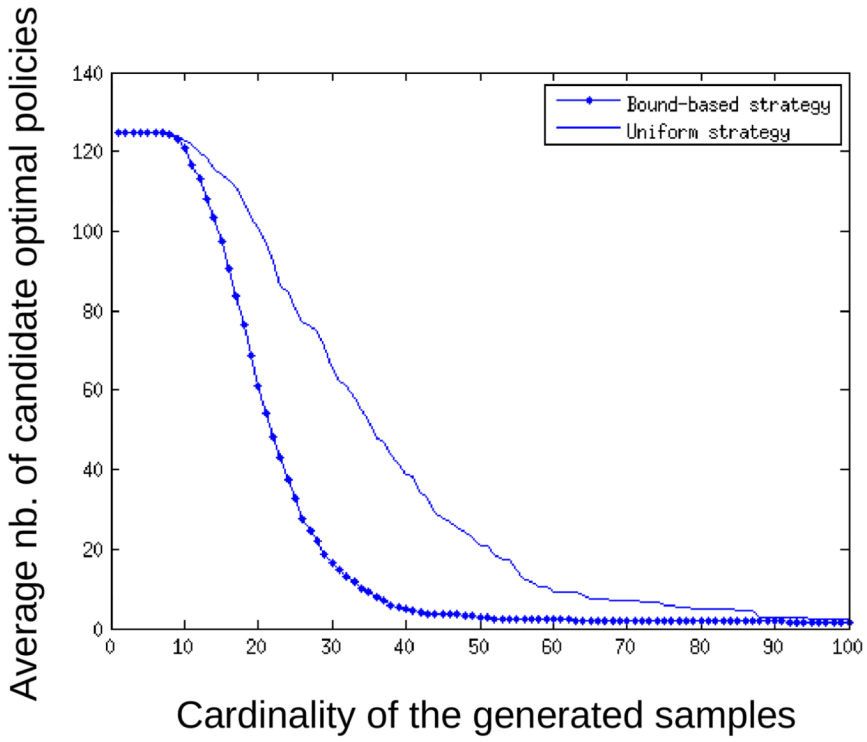


Fig. 13.
Evolution of the average number of candidate optimal policies with respect to the cardinality of the generated samples of transitions using our bound-based sampling strategy and a uniform sampling strategy (empirical average over 50 runs).

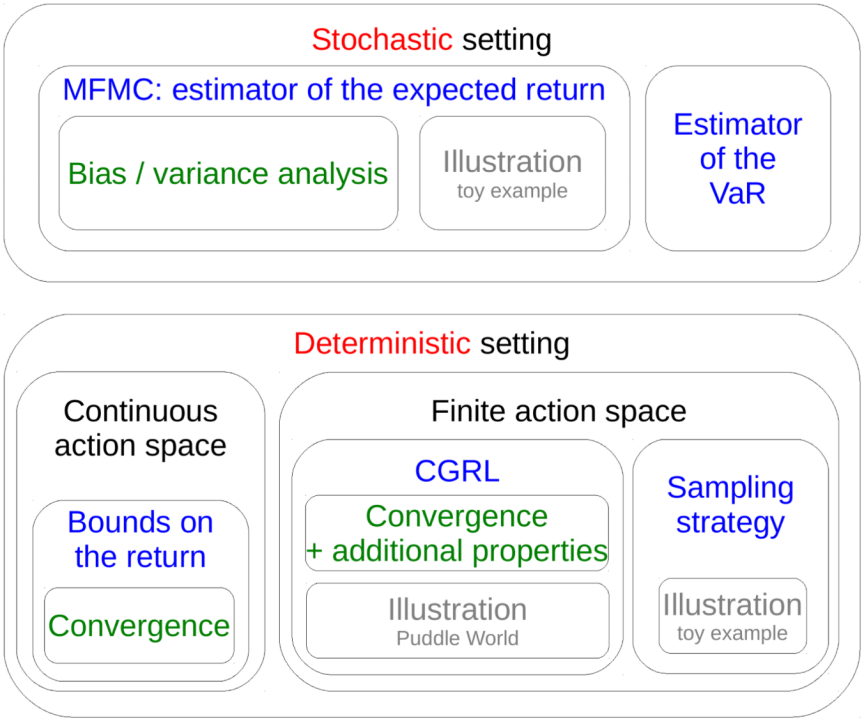


Fig. 14.
A schematic presentation of the results presented in Section 4. algorithm.

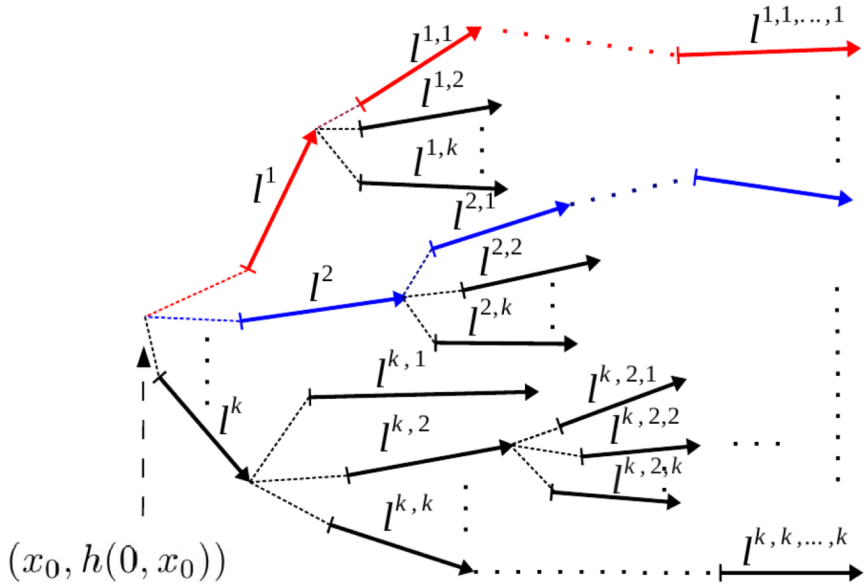


Fig. 15.
Illustration of the *k*–NN value iteration algorithm in terms of artificial trajectories.

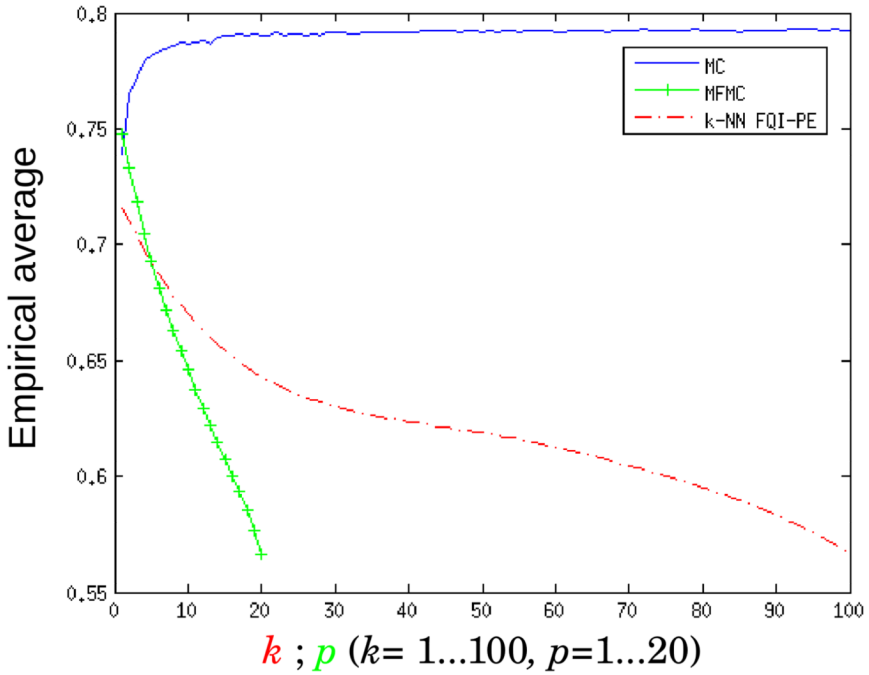


Fig. 16.

Empirical average observed for the MC estimator, the MFMC estimator and the *k*–NN FQI-PE estimator for different values of *k* and *p* (*k* ∈ {1,…, 100}, *p* ∈ {1,…, 20}, 1000 runs for each value of *k, p*).

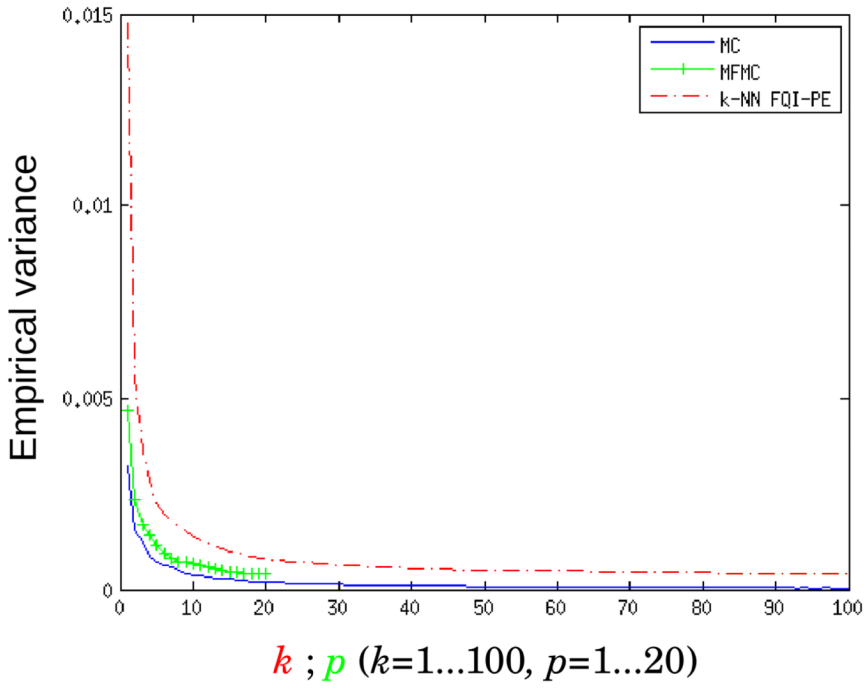


Fig. 17.

Empirical variance observed for the MC estimator, the MFMC estimator and the k–NN FQI-PE estimator for different values of k and p ($k \in \{1,\ldots, 100\}$, $p \in \{1,\ldots, 20\}$, 1000 runs for each value of k, p).

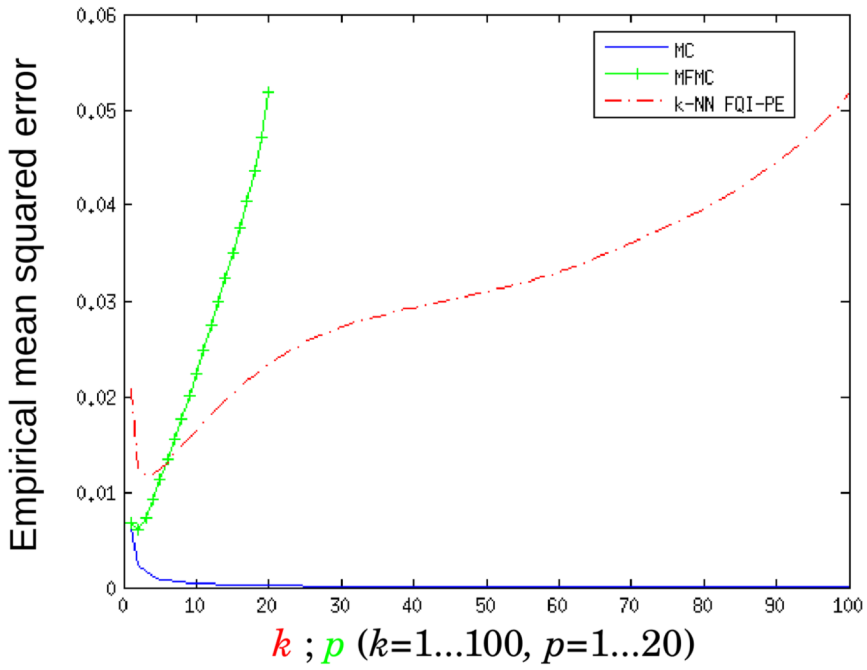
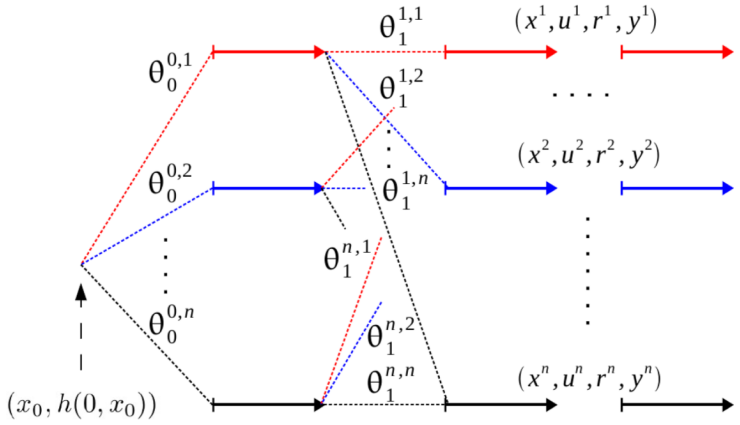


Fig. 18.

Empirical mean square error observed for the MC estimator, the MFMC estimator and the k–NN FQI-PE estimator for different values of k and p ($k \in \{1,\ldots, 100\}$, $p \in \{1,\ldots, 20\}$, 1000 runs for each value of k, p).



$$\theta_0^{0,i_0} = k_{\mathcal{F}_n}\Big((x_0, h(0, x_0)), (x^{i_0}, u^{i_0})\Big) \ ,$$

$$\theta_{t+1}^{i_t,i_{t+1}} = k_{\mathcal{F}_n}\Big((y^{i_t}, h(t+1, y^{i_t})), (x^{i_{t+1}}, u^{i_{t+1}})\Big), \forall t \in \{0, \ldots, T-2\} \ .$$

Fig. 19.
Illustration of the KB value iteration agorithm in terms of artificial trajectories.