Exploitation vs Caution: Risk-sensitive Policies for Offline Learning

Giorgio Angelotti, Nicolas Drougard, Caroline P. C. Chanel
1ISAE-SUPAERO, University of Toulouse, France
2ANITI, University of Toulouse, France
{name.surname}@isae-supraero.fr

Abstract

Offline model learning for planning is a branch of machine learning that trains agents to perform actions in an unknown environment using a fixed batch of previously collected experiences. The limited size of the data set hinders the estimate of the Value function of the relative Markov Decision Process (MDP), bounding the performance of the obtained policy in the real world. In this context, recent works showed that planning with a discount factor lower than the one used during the evaluation phase yields more performing policies. However, the optimal discount factor is finally chosen by cross-validation. Our aim is to show that looking for a sub-optimal solution of a Bayesian MDP might lead to better performances with respect to the current baselines that work in the offline setting. Hence, we propose Exploitation vs Caution (EvC), an algorithm which automatically selects the policy that solves a Risk-sensitive Bayesian MDP in a set of policies obtained by solving several MDPs characterized by different discount factors and transition dynamics. On one hand, the Bayesian formalism elegantly includes model uncertainty and on another hand the introduction of a risk-sensitive utility function guarantees robustness. We evaluated the proposed approach in different discrete simple environments offering a fair variety of MDP classes. We also compared the obtained results with state-of-the-art offline learning for planning baselines such as MOPO and MOReL. In the tested scenarios EvC is more robust than the said approaches suggesting that sub-optimally solving an Offline Risk-sensitive Bayesian MDP (ORBMDP) could define a sound framework for planning under model uncertainty.

1 Introduction

The development of autonomous agents in an unknown, and possibly stochastic, environment is a delicate task that usually requires a continuous agent-environment interaction. Notably, the Reinforcement Learning (RL) community has developed performing algorithms using specifically suited virtual environments as the ones of the OpenAI Gym Learning Suite [1] and ATARI games contained in the Arcade Learning Environment [2] to benchmark their performances. Nevertheless, a continuous and multiple interaction with the environment is not always affordable in real life situations. Indeed, in multiple applications like the training of medical robots and automated vehicles [3-4] the interaction with the environment can be both too risky and expensive since 1) any mistake could lead to catastrophic aftermaths or 2) the data collection phase requires a direct human involvement, which is not always available. Hence, it can be convenient to exploit previously collected data sets in order to limit additional (dangerous or superfluous) interaction.

Offline learning is the branch of machine learning that leverages previously collected batches of experiences with the aim of establishing an optimal behavioural policy. In recent years, the RL community published a great number of papers on the subject [5-11], demonstrating the growing

Preprint. Under review.
interest in the field. The proposed algorithms try to improve the performance of a policy obtained
either with model-free or with model-based RL approaches. The intuition behind these methods is
always the same: optimizing a trade-off between exploitation and caution. The policy optimization
procedure is constrained in order to generate strategies that are not too distant from the one originally
used to collect the batch. In this way, the agent will follow a strategy that will not drive him towards
regions of the state-action space for which it possesses a high degree of uncertainty. More specifically,
building upon the Markov Decision Process (MDP), such a constraint is implemented: (i) in [12] as a
penalty in the value function proportional to an estimate of the policies’ distributional shift, and (ii)
in [9] as a penalization of the reward function proportional to an estimate of the distributional shift in
the dynamical evolution of the system, also called epistemic (model) error.

Interestingly, [12] showed that planning using a discount factor \( \gamma^* \) lower than the one used in the final
evaluation phase \( \gamma_{ev} \) yields more performing policies when a trivially learnt model is considered. A
trivially learnt model is said to be the one which maximizes the likelihood of the transitions collected
in the batch. Nevertheless, the mathematical expression that should be optimized in order to find
\( \gamma^* \) is intractable. Therefore, the optimal discount factor is finally found by cross-validation which
requires additional interaction with the true environment.

In parallel, with the aim of finding a policy that optimizes the trade-off between exploitation and
evaluation in an online setting, model uncertainty has been included in a Bayesian extension
of the MDP framework called Bayesian (Adaptive) MDP (BAMDP) [13]. Fixing a prior for the
distribution of transition models, a posterior distribution is computed from the likelihood of the
sampled trajectories. Intuitively, a BAMDP can be seen as a Partially Observable MDP (POMDP)
where the belief (the posterior) is over transition models and not over states [14]. As suggested by
[13], risk-sensitive utility functions can replace the common BAMDP value function. Doing so, the
said work proposes an algorithm that trades off exploration, exploitation and robustness (caution).

In this context, the present work aims to show that solving an Offline Risk-sensitive Bayesian
(non-adaptive) MDP (ORBMDP) could define the state-of-the-art standard of both optimization
and evaluation of model-based sequential decision making under uncertainty in the offline setting.
Therefore, taking inspiration from [12] and [15], this study proposes Exploitation vs Caution (EvC):
an algorithm which balances exploitation and robustness by finding a reasonable solution for an
ORBMDP while constraining the quest for the policy to a limited set. More specifically, the said
strategies are the ones obtained by solving several MDPs with different discount factors and different
transition functions which are sampled from the Bayesian posterior inferred from the fixed batch. In
conclusion, EvC aims to be a proof-of-concept paradigm. It serves the purpose of supporting the
thesis that in the offline setting a Risk-sensitive BMDP is the right problem to be solved if the goal is
to obtain a policy balancing the exploitation-robustness trade-off.

The paper is organized as follows: Section 2 starts with a recap of the MDP and of the Bayesian
MDP (BMDP) formalisms; Section 3 defines risk-sensitive measures following the prescriptions of
[16] and introduces the Risk-Sensitive BMDP; Section 4 proposes the EvC algorithm that obtains a
risk-sensitive policy for ORBMDP; In Section 5, EvC performance is compared against a modified
version of MOPO [9] and MOReL [10] with the epistemic uncertainties given by an oracle; Section 6
concludes by discussing the limitations of the approach and points future work perspectives.

2 Background

Definition 2.1. A Markov Decision Process (MDP) is a 6-tuple \( M \overset{\text{def}}{=} (S, A, T, R, \gamma, \mu_0) \)
where \( S \) is the set of states, \( A \) the set of actions, \( T : A \times S \times S \rightarrow [0, 1] \) is the state transition function
defining the probability that dictates the evolution from \( s, a \in A \), \( R : A \times S \rightarrow [R_{\min}, R_{\max}] \)
with \( R_{\max}, R_{\min} \in \mathbb{R} \) is the expected reward function that indicates what
the agent gains, on average, when the system state is \( s \) and action \( a \) is applied, \( \gamma \in [0, 1] \) is
called the discount factor and \( \mu_0 : S \rightarrow [0, 1] \) is the distribution over the initial state.

Definition 2.2. A policy is a function that maps states to a probability distribution over actions, such as \( \pi : A \times S \rightarrow [0, 1] \).

Definition 2.3. Solving an MDP amounts to finding a policy \( \pi^* \) which, \( \forall s \in S \), maximizes the value function:

\[
V_M^\pi(s) \overset{\text{def}}{=} \mathbb{E}_{s_t \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s \right]. 
\]
It has been proved that an MDP for which the value function is defined as Eq. (1) admits a deterministic optimal policy (a map from states to actions) [17]:

$$\pi^* (s) = \text{argmax}_\pi V^*_M (s).$$

**Definition 2.4.** The performance of a policy $\pi$ in an MDP $M$ with value function $V^*_M$ is:

$$U^*_M = \mathbb{E}_{s \sim \rho_0} [V^*_M (s)].$$

**Definition 2.5.** A BMDP is a 8-tuple $\beta \overset{\text{def}}{=} \langle S, A, T, \mathcal{R}, \tau, \rho, \gamma, \mathcal{B} \rangle$ where $S$ is the set of states, $A$ the set of actions, and $\gamma \in [0, 1)$ is a discount factor. $T$ is a parametric family of transition functions of any MDP compatible with $S$ and $A$. $\mathcal{R}$ is a parametric family of reward functions of any MDP compatible with $S$ and $A$. $\tau$ is a non-informative prior distribution uniform over $T$. $\rho$ is a non-informative prior distribution uniform over $\mathcal{R}$. $\mathcal{B} = \{(s_t, a_t, r_t, s_{t+1})\}$ is a batch of transitions generated by acting in an unknown MDP with transition function $T$, reward $R$ and initial state distribution $\rho_0$.

For instance, in a finite state and action spaces environment, $T$ is the set of all $|S| \times |A|$ different categorical distributions and $\tau$ is made of $|S| \times |A|$ Dirichlet probability density functions – the conjugate prior of the said distribution.

**Definition 2.6.** $\tau_p$ is a posterior distribution over $T$ obtained by updating the prior $\tau$ with the information contained in $\mathcal{B}$.

In particular, the $|S|$ probability values $x$ describing $(s = s^*, a = a^*) \rightarrow s'$ can be distributed as:

$$\tau^*_{p^*} (x_1, \ldots, x_{|S|} | n_1, \ldots, n_{|S|}) = \Gamma (\nu) \prod_{i=1}^{|S|} \frac{x_i^{n_i}}{\Gamma (n_i + 1)}$$

where, $\Gamma$ is the Euler gamma function, $n_i$ counts how many times the transition $(s^*, a^*) \rightarrow s_i$ appears in $\mathcal{B}$ and $\nu = \sum_{k=1}^{|S|} (n_k + 1)$.

**Remark.** Notice that the mode (the most likely configuration) of the posterior in Eq. (4) is given by $\hat{x}_i = \frac{n_i}{\sum_{k=1}^{|S|} n_k}$ while its expected value is $\mathbb{E}_{\tau_p} [X_i] = \frac{n_i + 1}{\nu}$.

In discrete environments the most likely transition model with respect to $\tau_p$ is the one for which the transition probabilities are given by the transition frequencies in $\mathcal{B}$. We will refer to these distributions also as $\tilde{T}$ or as the trivial model. A similar reasoning can be done for $\mathcal{R}$ and $\rho$ but for simplicity’s sake we will assume to know the reward function $R$.

**Remark.** It would be possible to define a prior over the initial states and obtain a posterior taking into account the information contained in the batch $\mathcal{B}$. For simplicity we will also assume that $\rho_0$ is known.

**Definition 2.7.** A solution to a BMDP $\beta$ is a policy which maximizes the following utility function:

$$U^*_\beta \overset{\text{def}}{=} \mathbb{E}_{s \sim \rho_0} [U^*_M]$$

where, $U^*_M \overset{\text{def}}{=} \mathbb{E}_{s \sim \rho_0} [V^*_M (s)]$ is the value function of an MDP, averaged on the initial state, with transition function sampled from $\tau_p$.

**Remark.** Since the true MDP is unknown, leveraging the Bayesian framework is an elegant way to incorporate uncertainty. However, the additional expected value makes Eq. (5) hard to be computed with Bellman’s recursive approaches or approximated with temporal differences methods.

The optimal performance with respect to Eq. (5) will be the one that, on average, works the best on the BMDP $\beta$ when the model is distributed according to the Bayesian posterior:

$$U^*_{\beta} = \max_\pi U^*_\beta$$

### 3 Risk-sensitive measures and solutions

Solving a BMDP deals with epistemic uncertainty more elegantly than solving an MDP for the most likely model $\hat{M} = (\hat{T}, \hat{R})$, but the utility function defined in Eq. (5) does not minimize the risk of obtaining a bad performing policy in the real environment.
With this in mind, it can be useful to define risk-sensitive utility functions. Risk measures are widely used, especially in fields such as finance and engineering. As stated in the end of Section 2, the expectation over the distribution of models makes the resolution of the BMDP a computational problem which is often intractable. Moreover, a Risk-sensitive BMDP also presents an additional difficulty: the risk measure may require an estimate of the quantiles of the distribution up to the $q$-quantile. This problem should be as small as possible because Policy Evaluation has to be performed $L_n$ times in order to obtain the Bayesian posterior distribution of values assumed by the Value Functions. This problem has been addressed by [19]. The said procedure allows to iteratively sample values from a distribution.

**Definition 3.1 (Value at Risk).** Let $X$ be a continuous random variable with values in $[A, B]$ distributed according to a pdf $p : [A, B] \rightarrow \{0, +\infty\}$ and let $q \in [0, 1]$, then VaR$_q$ is the solution to the equation $\int_A^x dx' p(x') = q$.

**Definition 3.2 (Conditional Value at Risk).** Let $X$ be a continuous random variable with values in $[A, B]$ distributed according to a pdf $p : [A, B] \rightarrow \{0, +\infty\}$ and let $q \in [0, 1]$, then CVaR$_q = E_{p}[X | X \leq \text{VaR}_q]$.

Strictly speaking, the VaR$_q$ is the $q$-quantile of the distribution, while the CVaR$_q$ is the expected value of the distribution up to the $q$-quantile.

### 3.1 Risk-sensitive Solutions to BMDPs

In the following, the BMDP utility of Eq. (5) taking also the risk into account is generalized.

**Definition 3.3.** Let $\beta$ be a BMDP and let $V_M^\pi(s)$ be the value function at state $s$ while following a policy $\pi$ in the MDP $M$ with transitions distributed according to the posterior. Let also $p(U_M^\pi|B)$ be the pdf over the possible values assumed by $U_M^\pi = E_{s \sim \mu_0}[V_M^\pi(s)]$. Then a risk-sensitive utility function is defined as:

$$U_M^\pi_{\sigma} \overset{\text{def}}{=} E_{s \sim \mu_\sigma} [U_M^\pi]$$

where, $\sigma$ is a risk measure.

**Remark.** As a consequence of Definition 3.2 if $\sigma = \text{CVaR}_1$, considering that $\text{CVaR}_1 = E_p[X]$, then the BMDP utility of Eq. (5) is a particular case of (7).

### 4 Solving a risk-sensitive BMDP in the offline setting

#### 4.1 Monte Carlo Confident Policy Selection

As stated in the end of Section 2, the expectation over the distribution of models makes the resolution of a BMDP a computational problem which is often intractable. Moreover, a Risk-sensitive BMDP also presents an additional difficulty: the risk measure may require an estimate of the quantiles of the unknown value distribution. An analytical maximization of the performance defined in (Eq. 7) is often either impossible or too computationally demanding. In order to tackle the maximization problem, a valuable choice can be resorting to a Monte Carlo estimate of the performance. We will then look for a sub-optimal policy, rather than an optimal one, by constraining the search to a set of policies $\Pi$. Which number $L_\pi \in \mathbb{N}$ of models would be necessary to sample in order to have an accurate estimate of the performance of a policy within a chosen confidence interval? Ideally, $L_\pi$ should be as small as possible because Policy Evaluation has to be performed $L_\pi$ times in order to obtain the Bayesian posterior distribution of values assumed by the Value Functions. This problem has been addressed by [19]. The said procedure allows to iteratively sample values from a distribution.
whose quantile is required until the estimate of the quantile will fall within an interval with a required probabilistic significance. We exploit this idea to propose a Monte Carlo Confident Policy Selection (MC2PS) algorithm, which is presented in Algorithm 1, to identify a robust policy for an ORBMDP among a set of candidate policies.

**Remark.** Also for big MDPs, also considering that the Policy Evaluations are carried out in parallel. Nevertheless, restricting the research to a subset of policies could be a viable solution of the space of all applicable policies of a finite state and action space MDP is \( \Lambda \) be the total number of models sampled. Then it leverages the estimate of both the quantile and of the all sampled values to obtain an estimate of the utility function \( U_{\beta, \sigma}^* \) for a specific risk measure \( \sigma \): \( \text{VaR} \) or \( \text{CVaR} \) (Lines 25-28). Finally, when the utility function has been estimated for every policy, it outputs the policy which maximizes it (Line 4).

**Remark.** Let \( \Lambda \) be the total number of models sampled to estimate the quantile of the performance distribution among policies: \( \Lambda = \sum_{\pi \in \Pi} L_{\pi} \). MC2PS performs Policy Evaluation \( \Lambda \) times. The size of the space of all applicable policies of a finite state and action space MDP is \( |\Pi| = |A|^{|S|} \). It goes without saying that looking over the whole policy space can be practically intractable even for not so big MDPs. Nevertheless, restricting the research to a subset of policies could be a viable solution also for big MDPs, also considering that the Policy Evaluations are carried out in parallel.

---

**Algorithm 1 Monte Carlo Confident Policy Selection**

**Input:** set of policies \( \Pi \), significance level \( \alpha \in [0, 1] \), sampling batch size \( k \in \mathbb{N} \), relative error tolerance \( \varepsilon_{rel} \in [0, 1] \), posterior distribution \( \tau_p \), quantile order \( q \in [0, 1] \), risk measure \( \sigma \), initial state distribution \( \mu_0 \).

**Output:** best policy \( \pi^* \).

1: for \( \pi \in \Pi \) do
2: \( U_{\beta, \sigma}^j \leftarrow \text{RISK EVALUATION} (\pi, \sigma, \tau_p, \mu_0, \varepsilon_{rel}, \alpha, q, k) \)
3: end for
4: return \( \pi^* = \arg \max_{\pi \in \Pi} U_{\beta, \sigma}^j \)

5: procedure \text{RISK EVALUATION}
6: **Input:** policy \( \pi \), risk measure \( \sigma \), posterior distribution \( \tau_p \), initial state distribution \( \mu_0 \), relative error threshold \( \varepsilon_{rel} \in [0, 1] \), significance level \( \alpha \in [0, 1] \), quantile order \( q \in [0, 1] \), sampling batch size \( k \in \mathbb{N} \).
7: Initialize \( U^\pi = \emptyset \) (empty list that will contain perf. (Eq. 3) for many models and fixed \( \pi \))
8: \( \text{(the next loop estimates the quantile needed to compute (Eq. 7) as prescribed in [19])} \)
9: repeat
10: for \( j \in \{1, \ldots, k\} \) in parallel do
11: Sample \( M_j \sim \tau_p \)
12: \( V_{M_j}^{\pi}(s) \leftarrow \text{Policy Evaluation on model } M_j \)
13: \( U_{\pi}^{M_j} \leftarrow \mathbb{E}_{s \sim \mu_0}[V_{M_j}^{\pi}(s)] \) (Equation 3)
14: \( U^\pi \leftarrow \text{append } U_{\pi}^{M_j} \)
15: end for
16: \( L_{\pi} \leftarrow |U^\pi| \) (since \( U^\pi \) grows (see Line 17), \( L_{\pi} \) increases)
17: Sort \( U^\pi \) in increasing order
18: Find indices \( (s, r) \in \mathbb{N}^2 \) such that \(|s - r|\) is minimal and:
19: \[ \sum_{i=r}^{s-1} (L_{\pi}^i) \cdot (1 - q) \cdot (1 - \varepsilon_{rel}) > 1 - \alpha \]
20: until:
21: \( U_{s}^\pi - U_{r}^\pi < \varepsilon_{rel} \cdot (U_{L_{\pi}}^\pi - U_{s}^\pi) \)
22: if \( \sigma = \text{VaR}_q \) then
23: return \( U_{[qL_{\pi}]}^\pi \)
24: else if \( \sigma = \text{CVaR}_q \) then
25: return mean \((U_{[1]}^\pi, \ldots, U_{[qL_{\pi}]}^\pi)\)
26: end procedure

For a given set of policies \( \Pi \) and for every policy \( \pi \in \Pi \), the algorithm incrementally samples \( k \) transition models from \( \tau_p \) and performs Policy Evaluation in parallel for each one of them until the stopping criterion is reached (see the for loop in Lines 1-3 calls Lines 6-29). The stopping criterion guarantees that the estimate of the \( q \)-quantile is statistically well approximated with a significance level \( \alpha \) within a dynamically sampled confidence interval whose width is proportional to \( \varepsilon_{rel} \) (Lines 21-24) given the total \( L_{\pi} \) models sampled. Then it leverages the estimate of both the quantile and of the all sampled values to obtain an estimate of the utility function \( U_{\beta, \sigma}^\pi \) for a specific risk measure \( \sigma \): \( \text{VaR} \) or \( \text{CVaR} \) (Lines 25-28). Finally, when the utility function has been estimated for every policy, it outputs the policy which maximizes it (Line 4).
4.2 The Exploitation vs Caution (EvC) Algorithm

Reference [12] shows that the policy obtained by solving an MDP \( \hat{M} = (S, A, \hat{T}, R, \gamma^*) \), trivially learnt from a batch of experiences collected from another MDP \( M = (S, A, T, R, \gamma) \), with \( \gamma^* \) a discount factor such as \( \gamma^* \leq \gamma_{ev} \), is more efficient in \( M \) than the policy obtained by solving \( \hat{M} \) using \( \gamma_{ev} \). The reason is that \( \hat{T} \) is an approximation of \( T \) and may not be trusted for the longest planning horizon. The selection of the best \( \gamma^* \) optimizes a trade-off between the exploitation of the information contained in the batch and the necessity of being cautious since the model estimate is not perfect.

Inspired by [12]'s conclusions, and also guided by the intuition that the true model \( M \) will be different from \( \hat{M} \), but close to it, we hope that the policies obtained by solving an MDP \( \hat{M} = (S, A, \hat{T}, R, \gamma) \) with \( \hat{T} close to \hat{T} \) and \( \gamma \leq \gamma_{ev} \) can be viable solutions for the ORBMDP.

Henceforth, the Exploitation versus Caution (EvC) algorithm is presented and schematized in Algorithm 2. This algorithm will search for a promising risk-sensitive policy by focusing the search in a set of candidate policies \( \Pi \) computed with different MDPs \( M \) and \( \gamma \leq \gamma_{ev} \) values. As already stated, we do not aim to find the optimal solution to the Bayesian MDP, but rather, to find a policy that is more robust than the one that we would obtain by optimally solving the trivial MDP.

Algorithm 2 Exploitation vs Caution

Input: quantile order \( q \in [0, 1] \), significance level \( \alpha \in [0, 1] \), sampling batch size \( k \in \mathbb{N} \), relative error tolerance \( \varepsilon_{rel} \in [0, 1] \), posterior distribution \( \tau_{p} \), risk measure \( \sigma \), initial state distribution \( \mu_{0} \), set of discount factors \( G \), number of models to solve \( l \in \mathbb{N} \).

Output: best policy \( \pi^{*} \).

1: \( \Pi \leftarrow \text{GeneratePolicies}(\tau_{p}, G, l) \)
2: return \( \pi^{*} = \text{MonteCarloConfidentPolicySelection}(q, \alpha, k, \varepsilon_{rel}, \tau_{p}, \Pi, \sigma, \mu_{0}) \) (Algorithm 1)
3:
4: procedure GeneratePolicies
5: \( \text{Input:} \) posterior distribution \( \tau_{p} \), set of discount factors \( G \), number of models to solve \( l \in \mathbb{N} \).
6: Initialize \( \mathbb{M} = \{l \text{ transition models} \sim \tau_{p}\} \cup \{\hat{T}\} \) \((\hat{T} \text{ is the trivial transition model})\)
7: Initialize \( \Pi = \emptyset \) \( \text{(an empty set)} \)
8: for \( (\gamma \in G, T \in \mathbb{M}) \) do
9: \( \pi(T, \gamma) \) = solution to the MDP with \( T \) and \( \gamma \)
10: Append \( \pi(T, \gamma) \) to \( \Pi \) if there is not already an identical policy in the set
11: end for
12: return \( \Pi \)
13: end procedure

In detail, EvC first generates candidate policies that will constitute the set \( \Pi \) (Line 1 calls the procedure defined in Lines 4-13 of Algorithm 1). For this, the trivial MDP \( \hat{M} \) and \( l \) additional MDPs are sampled from the Bayesian posterior \( \tau_{p} \) obtained from the batch (Line 6), and then solved with different values of \( \{\gamma \in G|\gamma \leq \gamma_{ev}\} \) (Lines 8-11). Recalling that \( \gamma_{ev} \) is the discount factor of the ORBMDP. Note that the obtained set \( \Pi \) has unrepeated solutions (Line 10). As a last step, MC2PS is launched with the obtained set of candidate policies \( \Pi \) returning the best risk-sensitive solution \( \pi^{*} \in \Pi \) (Line 2 calls Algorithm 1). For instance if we test over 9 different discount factors, such as \( G = \{0.1, 0.2, \ldots, \gamma_{ev} = 0.9\} \), and 5 different \( (l = 5) \) MDPs \( \hat{M} \) (including \( M \)), then we have to solve \( |\Pi| \leq 9 \cdot 45 = 405 \) MDPs to obtain the set of candidate policies.

4.3 Theoretical guarantees

Since EvC searches for the policy \( \pi \in \Pi \) that maximizes the criterion of Eq. (7), Algorithm 2 in principle yields a sub-optimal solution to the ORBMDP. Additionally, the lack of an exact analytical expression for Eq. (7), and the subsequent Monte Carlo estimate of the quantile, which is needed to compute the risk-sensitive metrics, makes the theoretical guarantee of the EvC paradigm loose.

In any case, assuming that the Bayesian posterior \( \tau_{p} \) efficiently encodes the model uncertainty, EvC outputs a policy whose performance in the real environment is guaranteed in probability to be greater than some value that changes with respect to the chosen risk-sensitive measure. We expect that for a
sufficiently big batch $B$ the optimal trivial policy should be more performing than the one obtained through EvC since the most-likely model should converge to real one.

**Theorem 1.** Let $\pi^*$ be the policy obtained by EvC and $U_{[q_{L\pi^*}]}^\pi$ be its $q$-quantile calculated through EvC. Let $U_M^\pi$ be the performance of $\pi^*$ with $M$ distributed according to the Bayesian posterior $\tau_p$. Assuming that sampling $U_M^\pi$ amounts to evaluating the said policy in the real MDP, then the sampled performance is greater than its $q$-quantile with probability $Pr(U_M^\pi \geq U_{[q_{L\pi^*}]}^\pi - \epsilon^\pi) \geq (1 - q)(1 - \alpha)$, where $\epsilon^\pi \overset{\text{def}}{=} \epsilon_{rel}(U_{[L\pi^*]}^\pi - U_{[L\pi^*]}^\pi)$ as defined in Algorithm 2.

**Proof.** Note that $\{U_M^\pi > \text{VaR}_{q\pi}^\pi\} \cap \{\text{VaR}_{q\pi}^\pi \geq U_{[q_{L\pi^*}]}^\pi - \epsilon^\pi\} \subseteq \{U_M^\pi \geq U_{[q_{L\pi^*}]}^\pi - \epsilon^\pi\}$, where $\text{VaR}_{q\pi}^\pi$ denotes the theoretical $q$-quantile. The events of the intersection depends on independent random variables – a future performance $U_M^\pi$ and a quantile estimation $U_{[q_{L\pi^*}]}^\pi$ – which allows to write $Pr(U_M^\pi \geq U_{[q_{L\pi^*}]}^\pi - \epsilon^\pi) \geq Pr(U_M^\pi \geq \text{VaR}_{q\pi}^\pi) \cdot Pr(\text{VaR}_{q\pi}^\pi \geq U_{[q_{L\pi^*}]}^\pi - \epsilon^\pi) \geq (1 - q)Pr(\epsilon^\pi \geq |\text{VaR}_{q\pi}^\pi - U_{[q_{L\pi^*}]}^\pi|) \geq (1 - q)(1 - \alpha)$. The last inequality is ensured by the quantile estimation (lines 21-24 in Algorithm 1), and the previous one by the definition of $\text{VaR}_q$. \qed

**Remark.** When the risk-sensitive measure used in EvC is $\text{VaR}_q$ the lower bound on $U_M^\pi$ in the Proof of Theorem 1 is maximized. If the risk-sensitive measure is $\text{CVaR}_q$ the empirical expected value over the $q$-fraction of low performing policies is maximized.

### 4.4 Consequences and applications

The purpose of Offline Learning is that of developing algorithms that can provide behavioural policies to be applied by real-world automated agents like autonomous cars or automated medical robots. Thus reducing the risk at the expense of a longer computational phase is not only commendable but compulsory. Will the policy obtained through MC2PS and EvC be good or entirely-risk free? This goes beyond the theoretical guarantees provided by the algorithms since its outputs depend not only on the characteristics of the environment and on the set of candidate policies but also on the quality and variety of the batch. A batch of transitions that is too small or too concentrated in the same region of the state-action space may result in policies that, even if they are guaranteed to handle the risk better than the trivial one, can still be catastrophic. Last but not least, several real-world problems involve non-stationary environments. The ORBMDP formalism is not suited to deal with this kind of problem. With this in mind, it would be appropriate to double-check the policy obtained from every offline learning algorithm before applying it in real-world scenarios.

### 5 Experiments

For the experiments, we selected three small and hence easy to study stochastic environments with different characteristics: two planning environments without absorbing states, Ring (5 states, 3 actions) and Chain (5 states, 2 actions), the former consisting in the stabilization of the agent in a particular non-absorbing goal with stochastic drift and the latter presenting cycles; and the Random Frozen Lake (RFL) environment, a re-adaptation of Frozen Lake from Open AI Gym suite [11] (8×8 grid world with fatal absorbing states). A detailed description of these environments is provided in the Supplementary Material.

#### 5.1 Setup

Let $(n, m) \in \mathbb{N}_2^+$, $m$ trajectories with $n$ steps each are generated following a random policy in each environment. We opted for a random data collecting procedure because we imagine using EvC in a scenario where both the developers and the autonomous agent are completely agnostic about the model dynamics and have no prior knowledge. The optimal performance for the true environment is computed by first obtaining the optimal policy with Policy Iteration and then the optimal value function with Policy Evaluation. Afterwards, the most likely transition model is inferred from the batch. The trivial MDP was then solved with the Policy Iteration algorithm and its relative performance in the true environment is obtained by Policy Evaluation. EvC data was computed with $\text{CVaR}_{0.25}$, $\text{VaR}_{0.25}$ and $\text{CVaR}_1$.

For comparison purposes, we compute the performances of MOPO [9] and MOReL [10], two state-of-the-art approaches that learn and solve MDPs in the offline setting. MOPO reshapes the reward assigned to transitions by penalizing it with a factor proportional to an estimate of the model
uncertainty for that specific transition. MOReL creates an additional and highly penalized absorbing state and makes the agent transit to it when the model uncertainty for a specific state-action pair is above a given threshold. In this way the optimal policies provided by both MOPO and MOReL avoid regions of the state-action spaces for which the model is expected to be not sufficiently well known. In our implementation of these baselines we utilize different hyperparameters, but the uncertainty dependent term is given by an oracle. With this in mind we compare EvC with the best possible applications of MOPO and MOReL in the said environments. More specifically: (1) In our implementation of MOPO, the penalized MDP is built starting from the trivial MDP with the reward decreased by $V_{\max} - \gamma_{\text{ev}} \cdot D_{\text{TV}} \left( T (\cdot|a,s), \hat{T} (\cdot|a,s) \right)$ where $V_{\max}$ is the maximum of the optimal Value Function of the true MDP and $D_{\text{TV}}$ is the total variation distance between the transitions of the trivial and the true MDP. Since in the practical implementation of MOPO in [9] a roll-out horizon is introduced as a hyperparameter we will take this aspect into account by solving the penalized MDP with several values of $\gamma$, corresponding to different planning horizons; (2) In our implementation of MOReL, we keep the penalization $\kappa$ fixed. The uncertainty is given by $D_{\text{TV}} \left( T (\cdot|a,s), \hat{T} (\cdot|a,s) \right)$ and the threshold $\alpha_{\text{MOReL}}$ varies between the simulations. The table containing the parameters of the simulations is provided in the Supplementary Material.

5.2 Results and discussion

For every environment we plot the performances $U_{\beta,\sigma}^{\pi_{\text{ev}}}$ of the policies obtained with a specific algorithm (Eq. 5) using the utility function defined in Eq. (7) normalized by the optimal one in a Letter-Value fashion [20]. The results are shown in Figure 2. The $N$ on the $x$-axis is the total number of transitions in each batch ($N = nm$). Statistics are performed over 50 batches for each $N$. In the case of RFL, we generated 4 different environments. We then plot the aggregate results where the statistics are performed over 200 batches for each $N$. The median of the distribution is displayed as a horizontal black line. Different quantiles are plotted as boxes of different sizes and shades, becoming smaller and more transparent the more extremal the quantile is. Outliers are shown as diamonds. In this way we keep track: (i) of the typical behaviour of an algorithm; (ii) of the whole distribution over the analyzed batches; and, (iii) of the minimal performance achieved over the said batches, hence visualizing how well each algorithm handles the risk.

In any tested scenario, the policies obtained by MOPO and MOReL surprisingly perform almost always worse than the one computed using the trivial MDP. Taking into account that the distributional shift was oracle given these results scale back the validity of the said approaches in the considered environments. In fact, even though the theoretical guarantees of MOReL and MOPO work in any MDP, it has to be said that they have usually been tested on continuous state MDPs driven by a deterministic dynamics, while here we are tackling non-deterministic environments.

There is not a substantial difference between the several risk-metrics used by EvC. CVaR$_1$ seems to be overall the most performing one, abiding by the non written rule of optimism in face of uncertainty. Still, CVaR$_1$ does not provide the same risk management probabilistic guarantees of VaR$_{0.25}$ and CVaR$_{0.25}$ because it maximizes the expected value over the all distribution (see Eq. 7) without focusing on the worst case scenarios. Both in Ring and in Chain (Figures 2a and 2b) the median is greater than the one of the trivial optimal policy. For RFL, 8 × 8 (Figure 2c) the EvC policies yield the highest average rewards for a batch made of 30 transitions and keep handling the risk better than the other baselines up to $N = 150$. Both MOPO and MOReL are outperformed by EvC and by the optimal trivial policy in this scenario. The minimum value of the normalized performance of the EvC methods is almost always higher than all the other ones. This makes EvC the most robust technique between the tested ones. However, in environments characterized by fatal absorbing states, like RFL, using EvC over the trivial optimal policy is not as effective as resorting to EvC in the other cases. We speculate that the information contained in a batch, with trajectories ending in a penalized absorbing state, is already enough to obtain a trivial model that, when solved, results in a good policy.

The major drawbacks of EvC are the heavy computational demands, the inefficient scalability to high dimensional environments and the requirement of a prior specification which could be not so straightforward, in particular for continuous MDPs. Moreover, it is worth recalling that the quality of EvC results is limited by the quality of the candidate policy set $\Pi$. Since the optimal trivial policy, the application of EvC to real world problems is safer than directly following the trivial optimal policy in terms of the considered risk measure. Yet, whether or not EvC can be used to obtain solutions that are also good and not only safer than the trivial one depends on the set of candidate policies.
(a) Normalized performance for the Ring Environment.

(b) Normalized performance for the Chain Environment.

(c) Normalized performance for the aggregate of 4 different RFL 8x8.

Figure 2: Letter-Value plot of the normalized performance $U$ of EvC with $CVaR_{0.25}$, $VaR_{0.25}$, $CVaR_1$, Trivial and both MOPO and MOREL with oracle given uncertainties. $N$ is the total number of transitions in a batch. Statistics are performed over 50 batches (200 in the case of RFL) for each $N$. The median is a horizontal line. Boxes of different sizes represent quantiles. Diamonds are outliers.

6 Conclusion and perspectives

In this work we showed that the ORBMDP defines an elegant mathematical framework that balances a trade-off between exploitation and caution in offline model-based sequential decision making under uncertainty. With the aim of achieving a promising solution for ORBMDPs, we developed EvC, an algorithm which aims to provide reasonably effective solutions to an ORBMDP. EvC finds a solution policy that maximizes the risk-sensitive utility function of Eq. (7) in a set of candidate policies. This set contains the strategies obtained by solving the trivially learnt MDP and other MDPs with transition dynamics sampled from the Bayesian posterior (e.g. the one shown in Eq. (4)) using different discount factors. If the Bayesian posterior efficiently encodes the uncertainty about the true models, then EvC guarantees to output a policy that performs in probability better than (or equal to) the trivial optimal one (Sec. 4.3).

Even if the EvC approach was tested in toy, hence easy to study, environments, it yielded sub-optimal policies that are more robust than the ones obtained with both MOPO and MOREL with an oracle given estimate of the distributional shift. Nevertheless, since EvC is based on the parallel resolution of a great number of models sampled from the Bayesian posterior in a Monte Carlo fashion, we doubt that it could efficiently scale to solve MDPs with a great number of states and actions.

In the short future we aim to improve the EvC approach by incrementally enriching the set of candidate policies following a heuristic. Also, we plan to extend EvC to tackle Partially Observable MDPs. Another promising line of research could concern the development of scalable algorithms that can efficiently solve big ORBMDPs. Moreover, investigating whether function approximators could be utilized to solve ORBMDP could also be an interesting track.
References

[1] Brockman, G., Cheung, V., Pettersson, L., et al. OpenAI Gym. ArXiv, abs/1606.01540, 2016.

[2] Bellemare, M. G., Naddaf, Y., Veness, J., et al. The Arcade Learning Environment: An Evaluation Platform for General Agents (Extended Abstract). J. Artif. Intell. Res., 47:253–279, 2013.

[3] Jonsson, A. Deep Reinforcement Learning in Medicine. Kidney Diseases, 5:1–5, 2018.

[4] Mirchevska, B., Pek, C., Werling, M., et al. High-level Decision Making for Safe and Reasonable Autonomous Lane Changing using Reinforcement Learning. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pages 2156–2162. 2018.

[5] Fujimoto, S., Conti, E., Ghavamzadeh, M., et al. Benchmarking Batch Deep Reinforcement Learning Algorithms. ArXiv, abs/1910.01708, 2019.

[6] Kumar, A., Fu, J., Soh, M., et al. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché Buc, E. Fox, R. Garnett, eds., Advances in Neural Information Processing Systems 32, pages 11784–11794. 2019.

[7] Wu, Y., Tucker, G., Nachum, O. Behavior Regularized Offline Reinforcement Learning. ArXiv, abs/1911.11361, 2019.

[8] Chen, J., Jiang, N. Information-Theoretic Considerations in Batch Reinforcement Learning. vol. 97 of Proceedings of Machine Learning Research, pages 1042–1051. PMLR, Long Beach, California, USA, 2019.

[9] Yu, T., Thomas, G., Yu, L., et al. MOPO: Model-based Offline Policy Optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin, eds., Advances in Neural Information Processing Systems, vol. 33, pages 14129–14142. 2020.

[10] Kidambi, R., Rajeswaran, A., Netrapalli, P., et al. MOREL: Model-Based Offline Reinforcement Learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin, eds., Advances in Neural Information Processing Systems, vol. 33, pages 21810–21823. 2020.

[11] Levine, S., Kumar, A., Tucker, G., et al. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. ArXiv, abs/2005.01643, 2020.

[12] Jiang, N., Kulesza, A., Singh, S., et al. The dependence of effective planning horizon on model accuracy. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, pages 1181–1189. 2015.

[13] Strens, M. A Bayesian framework for Reinforcement Learning. In In Proceedings of the Seventeenth International Conference on Machine Learning, pages 943–950. ICML, 2000.

[14] Duff, M. O., Barto, A. Optimal Learning: Computational Procedures for Bayes-Adaptive Markov Decision Processes. Ph.D. thesis, 2002. AAI3039353.

[15] Sharma, A., Harrison, J., Tsao, M., et al. Robust and Adaptive Planning under Model Uncertainty. Proceedings of the International Conference on Automated Planning and Scheduling, 29(1):410–418, 2019.

[16] Majumdar, A., Pavone, M. How Should a Robot Assess Risk? Towards an Axiomatic Theory of Risk in Robotics. In N. M. Amato, G. Hager, S. Thomas, M. Torres-Torriti, eds., Robotics Research, pages 75–84. Springer International Publishing, Cham, 2020.

[17] Kolobov, M., Kolobov, A. Planning with Markov Decision Processes: An AI perspective. Synthesis Lectures on Artificial Intelligence and Machine Learning, 6(1):1–210, 2012.

[18] Artzner, P., Delbaen, F., Jean-Marc, E., et al. Coherent Measures of Risk. Mathematical Finance, 9:203 – 228, 1999.

[19] Briggs, K., Ying, F. How to estimate quantiles easily and reliably. Mathematics Today, 2018(February):26–29, 2018.

[20] Heike, H., Wickham, H., Kafadar, K. Letter-Value Plots: Boxplots for Large Data. Journal of Computational and Graphical Statistics, 26, 2017.
A Environments

**Ring.** This environment is described by five states forming a single loop. Three actions are possible: $a$, $b$, $c$. The agent starts in state 0. $a$ will move it to the state $s-1$ if $s=0,1,3$ and with probability 0.5 if it is elsewhere. With $b$ the agent will not move with probability 0.8 and move to the left or to the right with probability 0.1 if it is in state $s=0,1,3$, if it is in state 2 or 4 it won’t move with probability 1. $c$ will move the agent to the right with probability 0.9 and it won’t move with probability 0.1 if it is in state $s=0,1,3$. Otherwise the same effects will apply, but with probability 0.5. The agent earns an immediate reward $r=0.5$ if it moves from 2 → 3 or 4 → 3 and $r=1$ for any transition 3 → 3. Elsewhere $r=0$. A graphical representation is shown in Figure 3.

![Figure 3: Representation of the Ring environment.](image)

**Chain.** The environment proposed in [13] was adapted. There are five states with the topology of an open chain and two actions. The agent starts from the state most to the left. With action $a$ the agent moves to the right and receives an immediate reward $r=0$ with probability 0.8. Once the agent is in the rightmost state, performing the first action lets him stay there and receive a reward $r=10$ with probability 0.8. It slips back to the origin earning a reward $r=2$ with probability 0.2. Action $b$ teleports the agent to the origin with probability 0.8 receiving a reward $r=2$ or let it go right with probability 0.2 earning $r=0$. The optimal policy consists of applying action $b$ in the first state and action $a$ in the others. A graphical representation is shown in Figure 4.

![Figure 4: [13]'s representation of the Chain environment.](image)

**Random Frozen Lake (RFL).** The Frozen Lake Environment of the Open AI Gym suite [1] was edited. The agent moves in a grid world ($4 \times 4$ or $8 \times 8$). It starts in the utmost left corner and it must reach a distant absorbing goal state that yields a reward $r=1$. In the grid there are some holes. If it falls in a hole it’s blocked there and can not move anymore, obtaining from that moment an immediate reward $r=0$. Unfortunately, the field is covered with ice and hence it is slippery. When the agent wants to move towards a nearby state it can slip with fixed probability $p$ and ends up in an unintended place. The grid is generated randomly assuring that there always exists a hole-free path connecting the start and the goal. Moreover, to each couple of action and non terminal state $(a,s)$ is assigned a different immediate reward $r$ sampled at random between $(0,0.8)$ at the moment of the generation of the MDP. The MDP itself does not have a stochastic reward, but the map and the rewards are randomly generated. A graphical representation (for a $3 \times 3$ grid) is shown in Figure 5.
Figure 5: Frozen Lake environment example with grid of size $3 \times 3$. The agent has to reach the goal, paying attention to slippery (blue) states and avoiding holes (black).

## B Parameters and Hyperparameters of the Experiments

Table 1: Parameters and hyperparameters used during the simulations: Env. is the environment type; RFL stands for Random Frozen Lake; $\gamma_{ev}$ is the evaluation gamma used to compute the optimal performance in the true MDP, in the trivial MDP and in MOReL; $n$ is the number of steps in each trajectory contained in a batch; $b$ is the number of different batches sampled for fixed batch size; $\alpha$ is the significance level of the Monte Carlo Confident Policy Selection (Algorithm 1 in the main paper); $\varepsilon_{rel}$ is the relative tolerance in Algorithm 1; $l$ is the number of different models sampled from the prior in EvC (Algorithm 2); $\{\gamma_{EvC}\}$ is the set of different discount factors used to solve the models sampled in EvC (Algorithm 2); The bold values are the ones displayed in the plots.

| Env.     | $\gamma_{ev}$ | $n$ | $b$   | $\alpha$ | $\varepsilon_{rel}$ | $l$ | $\{\gamma_{EvC}\}$       |
|----------|---------------|-----|-------|----------|----------------------|-----|---------------------------|
| Ring     | 0.9           | 8   | 50    | 0.01     | 0.01                 | 3   | {0.2, 0.4, 0.6, 0.8, 0.9}  |
| Chain    | 0.9           | 8   | 50    | 0.01     | 0.01                 | 3   | {0.2, 0.4, 0.6, 0.8, 0.9}  |
| RFL 1 (4 × 4) | 0.9       | 15  | 50    | 0.01     | 0.01                 | 3   | {0.2, 0.4, 0.6, 0.8, 0.9}  |
| RFL 2 (4 × 4) | 0.9       | 15  | 50    | 0.01     | 0.01                 | 5   | {0.2, 0.4, 0.6, 0.8, 0.9}  |
| RFL 3 (4 × 4) | 0.9       | 15  | 50    | 0.01     | 0.01                 | 10  | {0.2, 0.4, 0.6, 0.8, 0.9}  |
| RFL 4 (4 × 4) | 0.9       | 15  | 50    | 0.01     | 0.01                 | 10  | {0.2, 0.4, 0.6, 0.8, 0.9}  |
| RFL 1 (8 × 8) | 0.9       | 15  | 50    | 0.01     | 0.01                 | 3   | {0.2, 0.4, 0.6, 0.8, 0.9}  |
| RFL 2 (8 × 8) | 0.9       | 15  | 50    | 0.01     | 0.01                 | 5   | {0.2, 0.4, 0.6, 0.8, 0.9}  |
| RFL 3 (8 × 8) | 0.9       | 15  | 50    | 0.01     | 0.01                 | 10  | {0.2, 0.4, 0.6, 0.8, 0.9}  |
| RFL 4 (8 × 8) | 0.9       | 15  | 50    | 0.01     | 0.01                 | 10  | {0.2, 0.4, 0.6, 0.8, 0.9}  |
Table 2: Parameters and hyperparameters for MOPO and MOReL used during the simulations: Env. is the environment type; RFL stands for Random Frozen Lake; \{\alpha_{\text{MOReL}}\} is the set of different thresholds tested with MOReL, the bold value is the one that worked the best according to human eye inspection; \(\kappa\) is the absolute value of the penalization of the absorbing state in MOReL; \{\gamma_{\text{MOPO}}\} is the set of different discount factors tested with MOPO, the bold value is the one that worked the best according to human eye inspection. The bold values are the ones displayed in the plots.

| Env.     | \{\alpha_{\text{MOReL}}\} | \(\kappa\) | \{\gamma_{\text{MOPO}}\} |
|----------|-----------------------------|------------|--------------------------|
| Ring     | \{0.1, 0.2, 0.3, 0.4, 0.5\} | 100        | \{0.2, 0.4, 0.6, 0.8, 0.9\} |
| Chain    | \{0.1, 0.2, 0.3, 0.4, 0.5\} | 100        | \{0.2, 0.4, 0.6, 0.8, 0.9\} |
| RFL 1 (4 × 4) | \{0.1, 0.2, 0.3, 0.4, 0.5\} | 100        | \{0.2, 0.4, 0.6, 0.8, 0.9\} |
| RFL 2 (4 × 4) | \{0.1, 0.2, 0.3, 0.4, 0.5\} | 100        | \{0.2, 0.4, 0.6, 0.8, 0.9\} |
| RFL 3 (4 × 4) | \{0.1, 0.2, 0.3, 0.4, 0.5\} | 100        | \{0.2, 0.4, 0.6, 0.8, 0.9\} |
| RFL 4 (4 × 4) | \{0.1, 0.2, 0.3, 0.4, 0.5\} | 100        | \{0.2, 0.4, 0.6, 0.8, 0.9\} |
| RFL 1 (8 × 8) | \{0.1, 0.2, 0.3, 0.4, 0.5\} | 100        | \{0.2, 0.4, 0.6, 0.8, 0.9\} |
| RFL 2 (8 × 8) | \{0.1, 0.2, 0.3, 0.4, 0.5\} | 100        | \{0.2, 0.4, 0.6, 0.8, 0.9\} |
| RFL 3 (8 × 8) | \{0.1, 0.2, 0.3, 0.4, 0.5\} | 100        | \{0.2, 0.4, 0.6, 0.8, 0.9\} |
| RFL 4 (8 × 8) | \{0.1, 0.2, 0.3, 0.4, 0.5\} | 100        | \{0.2, 0.4, 0.6, 0.8, 0.9\} |

C Extra Plots

The results for the single RFL 8 × 8 environments are displayed in Figure 8. The aggregate plot for the RFL 4 × 4 environment can be found in Figure 6 along with the single results in Figure 7.

![Figure 6: Letter-Value plot of the normalized performance for the aggregate of 4 different RFL 4x4. EvC computed with $CVaR_{0.25}$, $VaR_{0.25}$, $CVaR_1$, Trivial and both MOPO and MOReL with oracle given uncertainties. $N$ is the total number of transitions in a batch. Statistics are performed over 50 batches for each $N$. The median is an horizontal black line. Boxes of different sizes represent quantiles. Diamonds are outliers.](image-url)



Figure 7: Letter-Value plot of the normalized performance $U$ of EvC with $CVaR_{0.25}$, $VaR_{0.25}$, $CVaR_1$, Trivial and both MOPO and MOREL with oracle given uncertainties. $N$ is the total number of transitions in a batch. Statistics are performed over 50 batches for each $N$. The median is an horizontal black line. Boxes of different sizes and shades represent quantiles of the distributions. Diamonds are outliers.
Figure 8: Letter-Value plot of the normalized performance $U$ of EvC with CVaR$_{0.25}$, VaR$_{0.25}$, CVaR$_1$, Trivial and both MOPO and MOREl with oracle given uncertainties. $N$ is the total number of transitions in a batch. Statistics are performed over 50 batches for each $N$. The median is an horizontal black line. Boxes of different sizes and shades represent quantiles of the distributions. Diamonds are outliers.