ECAI 2010
H. Coelho et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-606-5-361

Uncertainty Propagation for Efficient Exploration in Reinforcement Learning

Alexander Hans[1,2] and **Steffen Udluft**[2]

Abstract. Reinforcement learning aims to derive an optimal policy for an often initially unknown environment. In the case of an unknown environment, exploration is used to acquire knowledge about it. In that context the well-known exploration-exploitation dilemma arises—when should one stop to explore and instead exploit the knowledge already gathered? In this paper we propose an uncertainty-based exploration method. We use uncertainty propagation to obtain the Q-function's uncertainty and then use the uncertainty in combination with the Q-values to guide the exploration to promising states that so far have been insufficiently explored. The uncertainty's weight during action selection can be influenced by a parameter. We evaluate one variant of the algorithm using full covariance matrices and two variants using an approximation and demonstrate their functionality on two benchmark problems.

1 INTRODUCTION

In the machine learning field of reinforcement learning (RL) [18] one is concerned with an agent acting in an initially often unknown environment. The agent chooses its actions based on the current state, after executing an action the agent observes a state transition and a reward. The aim is to maximize the expected sum of (possibly discounted) future rewards. In the case of an initially unknown environment, the agent must explore to gather knowledge needed to act optimally, i.e., choose those actions that maximize the reward. In that context the well-known exploration-exploitation dilemma arises: when should the agent stop trying to gain more information (explore) and start to act optimally w.r.t. to already gathered information (exploit)?

In this paper, we propose a method that combines existing (already gathered) knowledge and uncertainty about the environment to further explore areas that seem promising judging by the current knowledge. Moreover, by aiming at obtaining high rewards and decreasing uncertainty at the same time, good online performance is possible.

Our algorithms use the uncertainty of the Q-function, which is determined by applying uncertainty propagation (UP) to the Bellman iteration. In previous work, this approach was used to derive robust policies (quality assurance) [10, 16]. Here we show how the same principle can be applied to uncertainty-based exploration.

The exploration-exploitation dilemma has been of interest for a long time, resulting in many contributions, some of them also dealing with model uncertainty like we do here (e.g., [1, 5–7, 12, 17, 19]).

[1] Ilmenau University of Technology, Neuroinformatics & Cognitive Robotics Lab, P.O. Box 100565, D-98684 Ilmenau, Germany, email: alexander.hans.ext@siemens.com
[2] Siemens AG, Corporate Research and Technologies, Otto-Hahn-Ring 6, D-81739 Munich, Germany, email: steffen.udluft@siemens.com

The main contribution of this paper is to show that using a natural measure of the uncertainty obtained via UP it is possible to explore efficiently without relying on an artificial exploration bonus. Furthermore, in two variants of our algorithm the Q-function itself is not modified and still represents the followed policy and actually collected rewards. Moreover, no "optimistic" initialization of the Q-function is necessary. Finally and most importantly, as presented in [10, 16], the method that is used for exploration here can be used for quality assurance by simply changing the parameter that influences how the uncertainty is considered for action selection.

2 BACKGROUND

2.1 Reinforcement Learning

In RL one is interested in finding a policy $\pi : S \mapsto A$ that moves an agent optimally in an environment assumed to be a Markov decision process (MDP) $M := (S, A, P, R)$ with a state space S, a set of possible actions A, the system dynamics, defined as probability distribution $P : S \times A \times S \mapsto [0,1]$, which gives the probability of reaching state s' by executing action a in state s, and a reward function $R : S \times A \times S \mapsto \mathbb{R}$, which determines the reward for a given transition. Moving the agent optimally means maximizing the value function

$$V^{\pi}(s) = \sum_{s'} P(s'|s,a) \left[R(s,a,s') + \gamma V^{\pi}(s') \right], \tag{1}$$

where $\gamma \in [0,1]$ is the discount factor. Often a so-called Q-function $Q^{\pi}(s,a)$ is utilized that gives the expected discounted reward when choosing action a in state s and afterward following policy π. The Q-function for the optimal policy $Q^{\pi^*} = Q^*$ is given by a solution of the Bellman optimality equation

$$\begin{aligned} Q^*(s,a) &= \mathbf{E}_{s'} \left[R(s,a,s') + \gamma V^*(s') \right] && (2) \\ &= \mathbf{E}_{s'} \left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a') \right]. && (3) \end{aligned}$$

From Q^* the optimal policy follows as $\pi^*(s) = \arg\max_a Q^*(s,a)$, where π^* is a deterministic policy. It can be shown that for any MDP an optimal deterministic policy exists [14].

2.2 Uncertainty Propagation

Uncertainty propagation (UP), also known as Gaussian error propagation (see, e.g., [3]), is a common method in statistics to propagate the uncertainty of measurements to the results. It is based on a first-order Taylor expansion. Given a function $f(x)$ with $f : \mathbb{R}^M \mapsto \mathbb{R}^N$

and the uncertainty of the function arguments as covariance matrix $\mathrm{Cov}(x)$, the uncertainty of the function values $f(x)$ is determined as

$$\mathrm{Cov}(f) = \mathrm{Cov}(f,f) = D\mathrm{Cov}(x)D^{\mathrm{T}}. \tag{4}$$

D is the Jacobian matrix of f w.r.t. x consisting of the partial derivatives of f w.r.t. to each component of x, i.e., $D_{i,j} = \frac{\partial f_i}{\partial x_j}$.

When neglecting correlations of the arguments x as well as correlations of the components of $f(x)$ the argument's covariance matrix and the resulting covariance matrix $\mathrm{Cov}(f)$ are diagonal. In this case, a simplified expression for determining the uncertainty σf_i of values $f_i(x)$ can be used:

$$(\sigma f_i)^2 = \sum_j \left(D_{i,j}\right)^2 (\sigma x_j)^2 \tag{5}$$

$$= \sum_j \left(\frac{\partial f_i}{\partial x_j}\right)^2 (\sigma x_j)^2. \tag{6}$$

Here, $(\sigma f_i)^2$, $i = 1,2,\ldots,N$ corresponds to the diagonal elements of $\mathrm{Cov}(f)$.

3 RELATED WORK

There have been many contributions considering efficient exploration in RL. E.g., Dearden et al. presented *Bayesian Q-learning* [5], a Bayesian model-free approach that maintains probability distributions over Q-values. They either select an action stochastically according to the probability that it is optimal or select an action based on *value of information*, i.e., select the action that maximizes the sum of Q-value (according to the current belief) and expected gain in information. They later added a Bayesian model-based method [4] that maintains a distribution over MDPs, determines value functions for sampled MDPs, and then uses those value functions to approximate the true value distribution. In *model-based interval estimation* (MBIE) [17, 19] one tries to build confidence intervals for the transition probability and reward estimates and then optimistically selects the action maximizing the value within those confidence intervals. Strehl and Littman proved that MBIE is able to find near-optimal policies in polynomial time [17]. This was first shown by Kearns and Singh for their E^3 algorithm [12] and later by Brafman and Tennenholtz for the simpler R-Max algorithm [1].

R-Max takes one parameter C, which is the number of times a state-action pair (s,a) must have been observed until its actual Q-value estimate is used in the Bellman iteration. If it has been observed fewer times, its value is assumed as $Q(s,a) = R_{\max}/(1-\gamma)$, which is the maximum possible Q-value ($R_{\max}$ is the maximum possible reward). This way exploration of state-action pairs that have been observed fewer than C times is fostered.

In [17] Strehl and Littman present an additional algorithm called *model-based interval estimation with exploration bonus* (MBIE-EB) of which they also prove optimality. According to their experiments, it performs similarly to MBIE. MBIE-EB alters the Bellman equation to include an exploration bonus term $\beta/\sqrt{n_{s,a}}$, where β is a parameter of the algorithm and $n_{s,a}$ the number of times state-action pair (s,a) has been observed.

There have also been a number of contributions considering the incorporation of *risk* in RL [9, 11, 13, 15]. These approaches deal with the risk of obtaining a low return in the single run. A risk that even exists for an optimal policy due to the inherent stochasticity of the MDP. The consideration of *uncertainty* [10, 16] deals with the uncertainty of the estimated parameters, due to our incomplete knowledge about the MDP. While this uncertainty decreases with an increasing number of observation, the stochasticity of the MDP and therefore the risk of obtaining a low return in the single run remains.

4 UNCERTAINTY PROPAGATION FOR EXPLORATION

We want to determine the Q-function together with its uncertainty σQ. We can then use the knowledge of uncertainty to guide the exploration to select actions that are promising w.r.t. their Q-value but still insufficiently explored.

To determine σQ, we start with an initial covariance matrix $\mathrm{Cov}(Q^0,P,R)$ and apply UP (section 2.2) to the update equation of the Bellman iteration

$$Q^m(s,a) := \sum_{s'} \hat{P}(s'|s,a)\left[\hat{R}(s,a,s') + \gamma V^{m-1}(s')\right], \tag{7}$$

where $\hat{P}$ and $\hat{R}$ denote the estimators for the transition probabilities and rewards, respectively, and V^{m-1} the value function of the previous iteration. The Bellman equation takes $\hat{P}$, $\hat{R}$, and V^{m-1} as arguments and produces Q^m as result. To apply UP, we use equation (4) on the Bellman equation to obtain the update rule

$$\mathrm{Cov}(Q^m,P,R) := D^{m-1}\mathrm{Cov}(Q^{m-1},P,R)(D^{m-1})^{\mathrm{T}}, \tag{8}$$

where D^m denotes the Jacobian matrix (obtained by differentiating equation (7))

$$D^m = \begin{pmatrix} D^m_{Q,Q} & D^m_{Q,P} & D^m_{Q,R} \\ 0 & \mathrm{I} & 0 \\ 0 & 0 & \mathrm{I} \end{pmatrix} \tag{9}$$

$$(D^m_{Q,Q})_{(i,j),(k,l)} = \gamma\pi^m(s_k,a_l)\hat{P}(s_k|s_i,a_j)$$

$$(D^m_{Q,P})_{(i,j),(l,n,k)} = \delta_{i,l}\delta_{j,n}\left(\hat{R}(s_i,a_j,s_k) + \gamma V^m(s_k)\right)$$

$$(D^m_{Q,R})_{(i,j),(l,n,k)} = \delta_{i,l}\delta_{j,n}\hat{P}(s_k|s_i,a_j).$$

Note that in the above definition of D^m a stochastic policy $\pi : S \times A \mapsto [0,1]$ is assumed that gives the probability of choosing action a in state s.[3] Starting with an initial covariance matrix

$$\mathrm{Cov}(Q^0,P,R) = \begin{pmatrix} \mathrm{Cov}(Q^0) & \mathrm{Cov}(Q^0,P) & \mathrm{Cov}(Q^0,R) \\ \mathrm{Cov}(P,Q^0) & \mathrm{Cov}(P) & \mathrm{Cov}(P,R) \\ \mathrm{Cov}(R,Q^0) & \mathrm{Cov}(P,R)^T & \mathrm{Cov}(R) \end{pmatrix} \tag{10}$$

$$= \begin{pmatrix} 0 & 0 & 0 \\ 0 & \mathrm{Cov}(P) & \mathrm{Cov}(P,R) \\ 0 & \mathrm{Cov}(P,R)^T & \mathrm{Cov}(R) \end{pmatrix}, \tag{11}$$

the update rule (8) is used in parallel to the Bellman update equation in each iteration to update the covariance matrix.

Finally, from the covariance matrix we can extract the Q-function's uncertainty as $\sigma Q^m = \sqrt{\mathrm{diag}(\mathrm{Cov}(Q^m))}$.

For the diagonal version, called the *diagonal approximation of uncertainty incorporating policy iteration* (DUIPI) [10], the update equation for σQ^m simplifies to

$$(\sigma Q^m(s,a))^2 := \sum_{s'}(d_{QQ})^2(\sigma V^{m-1}(s'))^2 + \sum_{s'}(d_{QP})^2(\sigma\hat{P}(s'|s,a))^2 + \sum_{s'}(d_{QR})^2(\sigma\hat{R}(s,a,s'))^2, \tag{12}$$

[3] A deterministic policy π_d can easily be mapped to a stochastic one by setting $\pi(s,a) := 1$ if $\pi_d(s) = a$ and $\pi(s,a) := 0$ otherwise.

$$\begin{aligned} d_{QQ} &= \gamma \hat{P}(s'|s,a), \\ d_{QP} &= \hat{R}(s,a,s') + \gamma V^{m-1}(s'), \\ d_{QR} &= \hat{P}(s'|s,a). \end{aligned}$$

While the full-matrix version operates on matrices ($\mathrm{Cov}(\cdot)$ and D^m are matrices), DUIPI can be implemented using only vectors. Accordingly, equation (12), updating a single value $(\sigma Q(s,a))^2$, must be applied repeatedly to update the whole vector $(\sigma Q)^2$. Relating to the complete covariance matrix $\mathrm{Cov}(Q^m,P,R)$, the vector $(\sigma Q)^2$ corresponds to the diagonal of the sub-matrix $\mathrm{Cov}(Q^m)$. For the diagonal version the uncertainty is initialized as $\sigma Q^0 := 0$. $\mathrm{Cov}(P)$ and $\mathrm{Cov}(R)$ as well as σP and σR depend on the choice of estimators (section 4.2).

For both variants, when the iteration converges, we obtain Q^* and σQ^*.

For further details we refer to [10, 16].

4.1 Negative ξ for Exploration

Now that we have the uncertainty of the Q-function, we can use it to derive a policy that looks for high Q-values and high uncertainty by setting ξ to a negative value:

$$\pi^{\xi}(s) := \arg\max_a \left[Q^*(s,a) - \xi \sigma Q^*(s,a) \right]. \tag{13}$$

With negative ξ to a Q-value its uncertainty σQ is added (weighted by ξ), thus making state-action pairs with high uncertainty more attractive. However, this way the uncertainty is only considered for one step, since Q^* does not contain the new action selection rule including σQ^*. Instead, Q^* represents the values of the expectation optimal policy, π^{ξ} is therefore inconsistent in considering the uncertainty.

To overcome this problem, the policy must be updated within the iteration, just like $V^m(s') = \max_{a'} Q^m(s',a')$ in value iteration implements policy improvement (resulting in Q^*). There are at least two ways of implementing policy improvement within the iteration:

Updating Q-values with uncertainty. The most obvious way of accounting for uncertainty is to modify the Q-values by adding or subtracting the ξ-weighted uncertainty in each iteration. However, this leads to a Q-function that is no longer the Q-function of the policy, as it contains not only the sum of (discounted) rewards, but also uncertainties. Therefore, using this Q and σQ it is not possible to reason about expected rewards and uncertainties when following this policy. Moreover, for the exploration case with negative ξ the Q-function does not converge in general for this update scheme, because in each iteration the Q-function is increased by the ξ-weighted uncertainty, which in turn leads to higher uncertainties in the next iteration. On the other hand, by choosing ξ and γ to satisfy $\xi + \gamma < 1$ it is possible to keep Q and σQ from diverging. In combination with DUIPI for exploration this has proven useful in our experiments, as it allows to use DUIPI successfully even for environments that exhibit high correlations between different state-action pairs, because by updating the Q-values the uncertainty is propagated through them.

Considering uncertainty for policy improvement. Instead of mixing Q-values and their uncertainty, it is also possible to change the update of the policy in each iteration according to equation (13). In the Bellman update equation (7) then $V^m(s) = Q^m(s,\pi^m(s))$, for the update of uncertainty $\sigma V^m(s) = \sigma Q^m(s,\pi^m(s))$. Using this method, the Q-values and their uncertainty are not mixed and the Q-function remains a valid Q-function for the resulting policy. This method was used in [10, 16].

From this follow three variants of the algorithm:

1. *Full-matrix UP* which considers the full covariance matrix and does not mix Q-values and their uncertainty,
2. *Classic DUIPI* (DUIPI) which neglects the non-diagonal elements of the covariance matrix and does not mix Q-values and their uncertainty,
3. *DUIPI with Q-modification* (DUIPI-QM) which also neglects the non-diagonal elements of the covariance matrix, but does modify the Q-values with their corresponding ξ-weighted uncertainty in each iteration.

4.2 Choice of Estimators

There are many possible ways for modeling estimators and their uncertainty for the transition probabilities P and the rewards R.

A popular method is the frequentist approach (sampling theory) using relative frequency as the estimator, i.e., $\hat{P}(s'|s,a) = \frac{n_{s,a,s'}}{n_{s,a}}$, where $n_{s,a,s'}$ denotes the number of observed transitions to state s' after being in state s and executing action a and $n_{s,a} = \sum_{s'} n_{s,a,s'}$ the total number of times when in state s action a was executed. Assuming that all transitions from different state-action pairs are independent of each other and the rewards, the transition probabilities can be modeled as multinomial distributions. In the corresponding covariance matrix exist therefore correlations only between transitions from identical state-action pairs. The covariance matrix is filled with values

$$(\mathrm{Cov}(P))_{(i,j,k),(l,m,n)} = \delta_{i,l}\delta_{j,m} \frac{P(s_k|s_i,a_j)(\delta_{k,n} - P(s_n|s_i,a_j))}{n_{s_i,a_j}}, \tag{14}$$

with the Kronecker delta ($\delta_{i,j} = 1$ if $i = j$ and $\delta_{i,j} = 0$ otherwise) realizing a correlation of 0 between transitions from different state-action pairs.

Similarly, the rewards can be estimated by their sample means, the corresponding covariance matrix is

$$(\mathrm{Cov}(R))_{(i,j,k)(l,m,n)} = \delta_{i,l}\delta_{j,m}\delta_{k,n} \frac{\mathrm{var}(R(s_i,a_j,s_k))}{n_{s_i,a_j,s_k} - 1}. \tag{15}$$

Another possibility is using a Bayesian setting. Again assuming all transitions from different state-action pairs to be independent of each other and the rewards, the transitions are modeled as multinomial distributions. The corresponding prior over the parameter space $P(s_k|s_i,a_j)$ for given i and j is then modeled as a Dirichlet distribution with prior parameters $\alpha_{ij1}, \alpha_{ij2}, \ldots, \alpha_{ij|S|}$ (note that we set $\alpha_{ij1} = \alpha_{ij2} = \ldots = \alpha_{ij|S|} = \alpha$). In the light of the observations these are updated to posterior parameters $\alpha^d_{ijk} = \alpha_{ijk} + n_{s_i,a_j,s_k}$. Assuming the posterior estimator $\hat{P}(s_k|s_i,a_j) = \alpha^d_{ijk} / \sum_{k=1}^{|S|} \alpha^d_{ijk}$, the covariance matrix for P then becomes

$$(\mathrm{Cov}(P))_{(i,j,k),(l,m,n)} = \delta_{i,l}\delta_{j,m} \frac{\alpha^d_{k,i,j}(\delta_{k,n}\alpha^d_{i,j} - \alpha^d_{n,i,j})}{(\alpha^d_{i,j})^2(\alpha^d_{i,j}+1)}. \tag{16}$$

Setting $\alpha = 0$ results in a prior that leads to the same estimates and slightly lower uncertainties compared to the frequentist approach. On the other hand, setting $\alpha = 1$ leads to a flat maximum entropy prior that assumes all transitions from a state to all other states equally probable. In [10] setting $\alpha = \frac{m}{|S|}$ is suggested, where m is the number of expected successor states. This way most of the probability mass is distributed among the first observed m successor states after

a few observations. It might also be worthwhile to use a hierarchical prior as proposed in [8]. The approach also tries to distribute the probability mass among a relatively low number of actually observed outcomes compared to the number of possible outcomes. Instead of directly estimating $\alpha = \frac{m}{|S|}$, one first uses a prior over the feasible sets of possible outcomes.

4.3 Putting It Together

Having selected an estimator for the transition probabilities and rewards, one first obtains estimations $\hat{P}$ and $\hat{R}$ along with uncertainties σP and σR. Starting with an arbitrarily initialized Q^0 and σQ^0, e.g. $Q^0 = \sigma Q^0 = 0$, as well as arbitrary policy π^0, equation (7) is used to obtain Q^{m+1}, where $V^m(s) = Q^m(s, \pi^m(s))$. For full-matrix UP equation (8) is used to obtain $\mathrm{Cov}(Q^{m+1})$, while for DUIPI equation (12) is used to obtain σQ^{m+1}. The updated Q^m and σQ^m are then used to update the policy:

$$\pi^{m+1}(s) := \arg\max_a Q^{m+1}(s,a) - \xi\, \sigma Q^{m+1}(s,a). \tag{17}$$

For DUIPI-QM, after having obtained Q^{m+1} and σQ^{m+1}, Q^{m+1} is modified using the weighted uncertainty:

$$Q^{m+1'}(s,a) := Q^{m+1}(s,a) - \xi\, \sigma Q^{m+1}(s,a). \tag{18}$$

$Q^{m+1'}$ is then used to update the policy:

$$\pi^{m+1}(s) := \arg\max_a Q^{m+1'}(s,a). \tag{19}$$

When used online, ideally a new policy is generated after each new observation. If this is too expensive, it is also possible to recalculate the policy only every n time steps. Naturally, unless $n = 1$ new information is not used as soon as it is available.

For our experiments (section 6) we used a Bayesian estimator with a Dirichlet prior for the transition probabilities and the expected value and the corresponding (frequentist) uncertainty for the rewards. Since all our MDPs exhibit deterministic rewards, using frequentist uncertainties yields correct uncertainties.

5 COMPUTATIONAL COMPLEXITY

Time Complexity. The time complexity of the standard Bellman iteration lies in $O(|S|^2|A|)$. DUIPI and DUIPI-QM add the step of updating the Q-function's uncertainty σQ, having a time complexity of also $O(|S|^2|A|)$. Thus DUIPI and DUIPI-QM have a time complexity of $O(|S|^2|A|)$. Full-matrix UP adds a time complexity of $O((|S||A|)^{2.376})$ for the update of the covariance matrix [2], thus having a higher time complexity than the standard Bellman iteration.

Space Complexity. The space complexity of the standard Bellman iteration is dominated by the transition probabilities P and the reward estimates R, both needing $O(|S|^2|A|)$ space each. The requirements of Q are $O(|S||A|)$, hence the total space complexity of the standard bellman iteration is $O(|S|^2|A|)$. DUIPI and DUIPI-QM add a complexity of $O(|S|^2|A|)$ for σP and σR each as well as $O(|S||A|)$ for σQ. Therefore, the total space complexity remains $O(|S|^2|A|)$. The full-matrix variant needs to hold the complete covariance matrix consisting of sub-matrices $\mathrm{Cov}(Q)$, $\mathrm{Cov}(Q,P)$, $\mathrm{Cov}(Q,R)$, $\mathrm{Cov}(P)$, $\mathrm{Cov}(P,R)$, and $\mathrm{Cov}(R)$, which equates to a space complexity of $O(|S|^5|A|^3)$.

Although those are only upper bounds (especially the covariance matrix is in practice often sparse), it is apparent that the computational burden of full-matrix UP is considerably higher than that of DUIPI and DUIPI-QM.

	full-matrix UP	DUIPI	DUIPI-QM
Time Complexity	$O((\lvert S\rvert\lvert A\rvert)^{2.376})$	$O(\lvert S\rvert^2\lvert A\rvert)$	$O(\lvert S\rvert^2\lvert A\rvert)$
Space Complexity	$O(\lvert S\rvert^5\lvert A\rvert^3)$	$O(\lvert S\rvert^2\lvert A\rvert)$	$O(\lvert S\rvert^2\lvert A\rvert)$

Table 1. Time and space complexities of the algorithms.

6 EXPERIMENTS

To demonstrate the functionality of our approach we conducted experiments using two benchmark applications from the literature.[4] We compare the full-matrix version, classic DUIPI, DUIPI with Q-function modification, and two established algorithms for exploration, R-Max [1] and MBIE-EB [17]. Furthermore, we present some insight of how the parameter ξ influences the agent's behavior.

6.1 Benchmarks

The first benchmark is the *RiverSwim* domain from [17], which is an MDP consisting of six states and two actions. The agent starts in one of the first two states (at the beginning of the row) and has the possibility to swim to the left (with the current) or to the right (against the current). While swimming to the left always succeeds, swimming to the right most often leaves the agent in the same state, sometimes leads to the state to the right, and occasionally (with small probability) even leads to the left. When swimming to the left in the very left state, the agent receives a small reward. When swimming to the right in the very right state, the agent receives a very large reward, for all other transitions the reward is zero. The optimal policy thus is to always swim to the right. See figure 1 for an illustration.

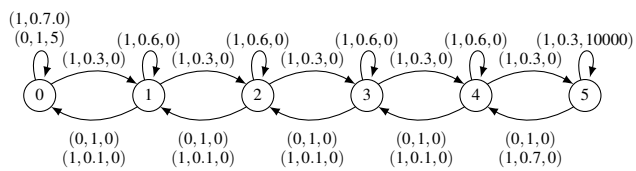

Figure 1. Illustration of the RiverSwim domain. In the description (a,b,c) of a transition a is the action, b the probability for that transition to occur, and c the reward.

The other benchmark is the *Trap* domain from [4]. It is a maze containing 18 states and four possible actions. The agent must collect flags and deliver them to the goal. For each flag delivered the agent receives a reward. However, the maze also contains a trap state. Entering the trap state results in a large negative reward. With probability 0.9 the agent's action has the desired effect, with probability 0.1 the agent moves in perpendicular direction (chosen randomly with equal probability). See figure 2 for an illustration.

For each experiment we measured cumulative reward for 5000 steps. The discount factor was set $\gamma = 0.95$ for all experiments.

6.2 Results

Table 2 shows the results for the considered domains and algorithms obtained with the respective parameters set to the optimal ones found.

[4] Source code for all algorithms and benchmark applications can be obtained at `http://ahans.de/publications/ecai2010`

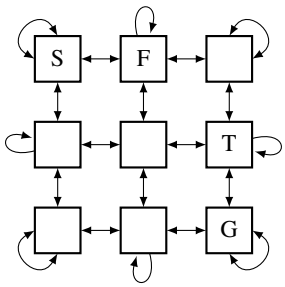

Figure 2. Illustration of the Trap domain. Starting in state S the agent must collect the flag from state F and deliver it to the goal state G. Once the flag is delivered to state G, the agent receives a reward of 1 and is transferred to the start state S again. Upon entering the trap state T a large negative reward of -10 is given. All other states yield a reward of 0. In each state the agent can move in all four directions. With probability 0.9 it moves in the desired direction, with probability 0.1 it moves in one of the perpendicular directions with equal probability.

Reported are results averaged over multiple trials, for each average its uncertainty is given as well.

	RiverSwim	Trap
R-Max	$3.02 \pm 0.03 \times 10^6$	469 ± 3
MBIE-EB	$3.13 \pm 0.03 \times 10^6$	558 ± 3
full-matrix UP	$2.59 \pm 0.08 \times 10^6$	521 ± 20
DUIPI	$0.62 \pm 0.03 \times 10^6$	554 ± 10
DUIPI-QM	$3.16 \pm 0.03 \times 10^6$	565 ± 11

Table 2. Best results obtained using the various algorithms in the RiverSwim and Trap domains. The used parameters for R-Max were $C = 16$ (RiverSwim) and $C = 1$ (Trap), for MBIE-EB $\beta = 0.01$ (RiverSwim) and $\beta = 0.01$ (Trap), for full-matrix UP $\alpha = 0.3$, $\xi = -1$ (RiverSwim) and $\alpha = 0.3$, $\xi = -0.05$ (Trap), for DUIPI $\alpha = 0.3$, $\xi = -2$ (RiverSwim) and $\alpha = 0.1$, $\xi = -0.1$ (Trap), and for DUIPI-QM $\alpha = 0.3$, $\xi = -0.049$ (RiverSwim) and $\alpha = 0.1$, $\xi = -0.049$ (Trap).

For RiverSwim all algorithms except classic DUIPI perform comparably. By considering only the diagonal of the covariance matrix, DUIPI neglects the correlations between different state-action pairs. Those correlations are large for state-action pairs that have a significant probability of leading to the same successor state. In RiverSwim many state-action pairs have this property. Neglecting the correlations leads to an underestimation of the uncertainty, which prevents DUIPI from correctly propagating the uncertainty of Q-values of the right most state to states further left. Thus, although Q-values in state 5 have a large uncertainty throughout the run, the algorithm settles for exploiting the action in the left most state giving the small reward if it has not found the large reward after a few tries. DUIPI-QM does not suffer from this problem as it modifies Q-values using uncertainty. In DUIPI-QM, the uncertainty is propagated through the state space by means of the Q-values.

In the Trap domain the correlations of different state-action pairs are less strong. As a consequence, DUIPI and DUIPI-QM perform equally well. Also the performance of MBIE-EB is good in this domain, only R-Max performs worse than the other algorithms. R-Max is the only algorithm that bases its explore/exploit decision solely on the number of executions of a specific state-action pair. Even with its parameter set to the lowest possible value, it often visits the trap state and spends more time exploring than the other algorithms.

Although full-matrix UP performed worse than the approximate algorithm DUIPI-QM, we expect it in general to be the best performing algorithm and believe that the results here are due to peculiarities of the test domains.

6.3 Discussion

Figure 3 shows the effect of ξ for the algorithms. Except DUIPI-QM the algorithms show "inverted u"-behavior. If ξ is too large (its absolute value too small), the agent does not explore much and quickly settles on a suboptimal policy. If, on the other hand, ξ is too small (its absolute value too large), the agent spends more time exploring. We believe that DUIPI-QM would exhibit the same behavior for smaller values for ξ, however, those are not usable as they would lead to a divergence of Q and σQ.

Figure 4 shows the effect ξ using DUIPI in the Trap domain. While with large ξ the agent quickly stops exploring the trap state and starts exploiting, with small ξ the uncertainty keeps the trap state attractive for more time steps, resulting in more negative rewards.

Using uncertainty as a natural incentive for exploration is achieved by applying uncertainty propagation to the Bellman equation. Our experiments indicate that it performs at least as good as established algorithms like R-Max and MBIE-EB. While most other approaches to exploration assume a specific statistical paradigm, our algorithm does not make such assumptions and can be combined with any estimator. Moreover, it does not rely on state-action pair counters, optimistic initialization of Q-values, or explicit exploration bonuses. Most importantly, when the user decides to stop exploration, the same method can be used to obtain certain-optimal policies for quality assurance [10, 16] by setting ξ to a positive value.

While the full-matrix UP is the more fundamental and theoretically more sound method, its computational cost is considerable. If used with care, however, DUIPI and DUIPI-QM constitute valuable alternatives that proved well in practice. Although our experiments are rather small, we expect DUIPI and DUIPI-QM to also perform well on larger problems.

	full-matrix UP	DUIPI	DUIPI-QM
time	7 min	14 s	14 s

Table 3. Computation time for 5000 steps in the RiverSwim domain using a single core of an Intel Core 2 Quad Q9500 processor. The policy was updated in every time step.

7 CONCLUSION

In this paper we presented approaches to exploration based on uncertainty propagation. We developed two principal variants of the algorithm, one using the full covariance matrix and an approximate algorithm only considering the diagonal of the covariance matrix (DUIPI). While DUIPI lies in the same complexity class as the Bellman iteration and is thus computationally feasible, it fails to propagate uncertainties properly in domains with high correlations between different state-actions pairs. To overcome this problem, we modify the Q-values using their uncertainty, thus using the Bellman iteration to propagate uncertainty. We evaluated the algorithms using two benchmark MDPs from the literature and compared them to R-Max and MBIE-EB.

We showed that using a natural measure of uncertainty it is possible to explore efficiently. The method used here for exploration was previously used in for quality assurance [10, 16]. One can therefore use the method to first explore efficiently and later, when the exploration phase is over, change the parameter ξ to a positive value and use the gathered observations to determine a quantile optimal policy.

Future work will include additional experiments with different domains and further theoretical analysis. Moreover, it will be interest-

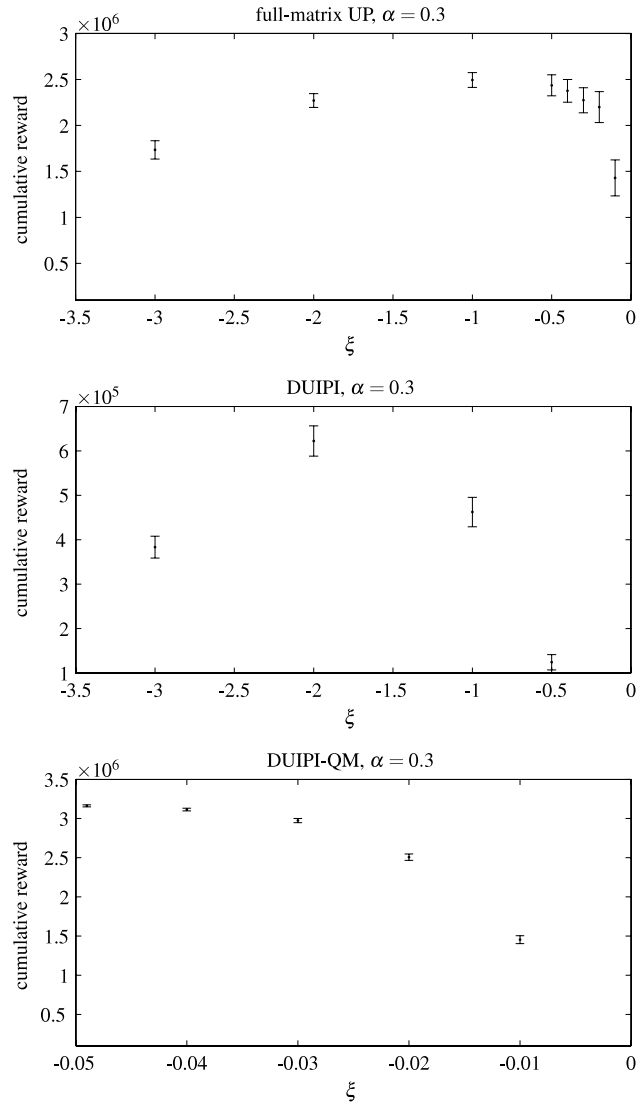

Figure 3. Cumulative rewards for RiverSwim obtained by the algorithms for various values of ξ. The values for full-matrix UP are averaged over 50 trials, for the values for DUIPI and DUIPI-QM 1000 trials of each experiment were performed.

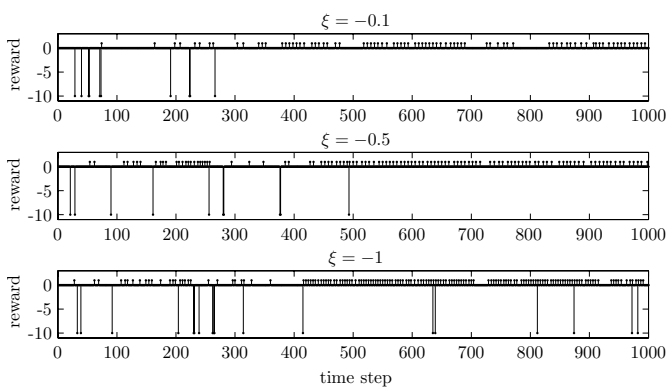

Figure 4. Immediate rewards of exemplary runs using DUIPI in the Trap domain. When delivering a flag, the agent receives reward 1, when entering the trap state it receives -10. While with $\xi = -0.1$ after less than 300 steps the trap state does not seem worth exploring anymore, setting $\xi = -0.5$ makes the agent explore longer due to uncertainty. With $\xi = -1$ the agent does not stop exploring the trap state in the depicted 1000 time steps.

ing to see whether DUIPI-QM also performs well in a quality assurance context.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their very valuable comments.

REFERENCES

[1] R.I. Brafman and M. Tennenholtz, 'R-max - a general polynomial time algorithm for near-optimal reinforcement learning', *Journal of Machine Learning Research*, **3**, (2003).
[2] D. Coppersmith and S. Winograd, 'Matrix multiplication via arithmetic progressions', *Journal of Symbolic Computation*, **9**, (1990).
[3] G. D'Agostini, *Bayesian Reasoning in Data Analysis: A Critical Introduction*, World Scientific Publishing, 2003.
[4] R. Dearden, N. Friedman, and D. Andre, 'Model based Bayesian exploration', in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, (1999).
[5] R. Dearden, N. Friedman, and S.J. Russell, 'Bayesian Q-learning', in *Proceedings of AAAI/IAAI*, (1998).
[6] E. Delage and S. Mannor, 'Percentile optimization in uncertain Markov decision processes with application to efficient exploration', in *Proceedings of the International Conference on Machine Learning*, (2007).
[7] A. Epshteyn, A. Vogel, and G. DeJong, 'Active reinforcement learning', in *Proceedings of the International Conference on Machine Learning*. Omnipress, (2008).
[8] N. Friedman and Y. Singer, 'Efficient Bayesian parameter estimation in large discrete domains', in *Advances in Neural Information Processing Systems*. MIT Press, (1999).
[9] P. Geibel, 'Reinforcement learning with bounded risk', in *Proceedings of the International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, (2001).
[10] A. Hans and S. Udluft, 'Efficient uncertainty propagation for reinforcement learning with limited data', in *Proceedings of the International Conference on Artificial Neural Networks*, (2009).
[11] M. Heger, 'Consideration of risk in reinforcement learning', in *Proceedings of the International Conference on Machine Learning*. Morgan Kaufmann, (1994).
[12] M. Kearns and S. Singh, 'Near-optimal reinforcement learning in polynomial time', in *Proceedings of the International Conference on Machine Learning*, (1998).
[13] R. Neuneier and O. Mihatsch, 'Risk sensitive reinforcement learning', in *Advances in Neural Information Processing Systems*, (1998).
[14] M.L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley & Sons Canada, Ltd., 1994.
[15] M. Sato and S. Kobayashi, 'Variance-penalized reinforcement learning for risk-averse asset allocation', in *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning, Data Mining, Financial Engineering, and Intelligent Agents*, London, UK, (2000). Springer-Verlag.
[16] D. Schneegass, S. Udluft, and T. Martinetz, 'Uncertainty propagation for quality assurance in reinforcement learning', in *Proceedings of the International Joint Conference on Neural Networks*, (2008).
[17] A.L. Strehl and M.L. Littman, 'An analysis of model-based interval estimation for markov decision processes.', *Journal of Computer and System Sciences*, **74**(8), (2008).
[18] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.
[19] M. Wiering and J. Schmidhuber, 'Efficient model-based exploration', in *Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats 5*, Montreal, (1998). MIT Press/Bradford Books.