Approximate Dynamic Programming With Correlated Bayesian Beliefs

Ilya O. Ryzhov and Warren B. Powell

***Abstract*— In approximate dynamic programming, we can represent our uncertainty about the value function using a Bayesian model with correlated beliefs. Thus, a decision made at a single state can provide us with information about many states, making each individual observation much more powerful. We propose a new exploration strategy based on the knowledge gradient concept from the optimal learning literature, which is currently the only method capable of handling correlated belief structures. The proposed method outperforms several other heuristics in numerical experiments conducted on two broad problem classes.**

I. INTRODUCTION

In classical dynamic programming, the infinite-horizon value $V(S)$ of a state $S \in \mathcal{S}$ can be found by iteratively solving a version of Bellman's equation [1], given by

$$V^n(S) = \max_{x \in \mathcal{X}} \mathbb{E}\left[C(S,x,W) + \gamma V^{n-1}(S') \mid S\right], \quad (1)$$

for every $S \in \mathcal{S}$. Here, $C(S,x,W)$ is a possibly random reward obtained by taking action x out of state S and observing some random information W, and $S' = S^M(S,x,W)$ is the next state of the dynamic program. This and other classical DP algorithms are discussed in [2].

In many realistic applications, the size of the state space $\mathcal{S}$ is too large to allow us to solve (1). For example, consider an energy storage problem in which the spot price of energy is modeled using a stochastic differential equation (see e.g. [3]). The price is part of the state variable, and it is continuous, making it impossible to solve (1) for every $S_t \in \mathcal{S}$. In such situations, we can employ techniques referred to under the names of reinforcement learning [4], neuro-dynamic programming [5], or approximate dynamic programming [6]. Such techniques typically compute an approximate observation

$$\hat{v}^n = \max_x C(S^n,x) + \gamma V^{n-1}\left(S^{M,x}(S^n,x)\right), \quad (2)$$

for the particular state S^n of the dynamic program in the nth time step. The function V^n is an approximation of V, and $S^{M,x}$ is a deterministic function mapping S^n and x to the *post-decision state* $S^{x,n}$. The concept of the post-decision state was first introduced by [7] and is extensively discussed by [6]; it allows us to handle problems where the expectation in (1) is difficult to compute. After solving (2), the observation $\hat{v}^n$ is used to update our approximation of the value of the previous post-decision state $S^{x,n-1}$ using stochastic approximation methods [8], and the next state $S^{n+1} = S^{M,W}\left(S^{x,n}, W^{n+1}\right)$ is obtained from some external process or via simulation.

A fundamental problem of approximate dynamic programming is known as *exploration vs. exploitation*. The choice of x that solves (2) is the decision that seems to be optimal under the most recent value function approximation. However, because the approximation is inaccurate, it may be desirable to make a different decision simply to learn something about a part of the state space that would otherwise remain unexplored, in the hope that this information will enable us to make better decisions in the future. There exist many general heuristics for exploration, such as ε-greedy policies or the soft-max method [4], as well as interval estimation [9].

The exploration/exploitation problem is also studied in the literature on optimal learning. Simple information collection problems such as ranking and selection [10] and multi-armed bandits [11] take place in a setting where there are finitely many reward processes with unknown stationary means. It is possible to perform "measurements" of individual processes to obtain noisy samples of the mean rewards. Each sample changes existing estimates of the means, which can be represented using a Bayesian belief structure. The goal is to efficiently allocate a finite measurement budget to discover the highest mean reward (ranking and selection) or to maximize the discounted sum of rewards (multi-armed bandits).

There have been several attempts to incorporate optimal learning techniques into exploration strategies for dynamic programming. Much of the work in this area focuses on Bayesian models for unknown transition probabilities in Markov decision processes, typically involving the Dirichlet prior distribution; an early study of this problem can be found in [12]. The work by [13] makes an explicit connection to multi-armed bandits for this setting. Other heuristics for the same problem include the value of information exploration technique by [14] and the BEETLE algorithm of [15], as well as work by [16] and [17]. The bias and variance of the value function approximations resulting from this method is studied by [18].

Another approach, first advanced by [19], is to place a Bayesian prior on the value $V(S)$ itself, rather than on specific problem parameters such as transition probabilities. The main appeal of this approach is its generality, as the value function can be viewed as encoding all of the uncertainty in the problem. The resulting Bayesian Q-learning method is more easily computable than the difficult calculations used in the studies on unknown transition probabilities. However, this work does not consider the correlations between values

This work was supported in part by AFOSR contract FA9550-08-1-0195 through the Center for Dynamic Data Analysis.

The authors are with the Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ (email: {iryzhov,powell}@princeton.edu).

of different states. For example, in the energy storage problem, we would expect the values of two states to be close together if the prices associated with those states are similar.

Our own approach is based on the same idea of placing a Bayesian belief structure on the value function. However, we adopt the correlated prior model from [20]. Correlations allow us to obtain information about many states from a single decision, and thus learn more quickly in fewer observations. We propose a new exploration strategy for dynamic programming based on the knowledge gradient (KG) method. This method was developed by [21] and [22] for ranking and selection, and extended by [23] and [24] to the multi-armed bandit problem. It consists of choosing the decision that yields the greatest value from a single measurement, and can yield easily computable decision rules with good numerical performance, even for more complex objective functions [25].

Section II describes the mathematical model for correlated beliefs about the value function. Section III presents our knowledge gradient policy for making decisions in approximate dynamic programming. In Section IV, we compare the KG policy to other exploration strategies on two classes of problems, namely inventory management and energy storage. The experiments provide encouraging results in support of the KG policy.

II. BAYESIAN MODEL WITH CORRELATED BELIEFS

Consider an infinite-horizon dynamic programming problem with state space $\mathcal{S}$, action space $\mathcal{X}$, and discount factor $\gamma \in (0,1)$. For notational convenience, we will assume that the reward $C(S,x)$ is a deterministic function of state and action. The objective function is assumed to be the usual MDP objective,

$$\sup_{\pi} \sum_{n=0}^{\infty} \gamma^n C\left(S^n, X^{\pi,n}\left(S^n\right)\right),$$

where $X^{\pi,n}$ is a decision rule associated with a policy π. The post-decision state $S^{x,n} = S^{M,x}(S^n, x)$ is also obtained deterministically from S^n and x, whereas the next pre-decision state $S^{n+1} = S^{M,W}\left(S^{x,n}, W^{n+1}\right)$ may be determined randomly. Let $\mathcal{S}^x$ denote the set of possible post-decision states. For now, we assume that both $\mathcal{S}$ and $\mathcal{S}^x$ are finite sets; we will show how this assumption can be relaxed in Section IV-B.

Let $R(S^x)$ be the total infinite-horizon discounted reward that we would receive if we were to start in the post-decision state $S^x \in \mathcal{S}^x$, then follow an optimal policy. Because transitions between pre-decision states are random, $R(S^x)$ is a random variable. The value of the state is obviously the mean of this random variable, that is, $V(S^x) = \mathbb{E}R(S^x)$. For the remainder of this study, V will be a function of the post-decision state rather than the pre-decision state.

Assumption 1: $R(S^x)$ has a normal distribution with unknown mean $V(S^x)$ and known variance $\lambda(S^x)$.

The work by [19] also assumes normality, justifying it with a central limit argument. We make the stronger assumption that $\lambda(S^x)$ is known. In practice, this is not the case, and the parameter $\lambda(S^x)$ would be set to a user-defined constant. The issue of choosing this constant is discussed below.

We now place a multivariate Gaussian prior with mean vector V^0 and covariance matrix Σ^0 on the value function. That is, we assume that $\mathbb{E}V(s) = V^0(s)$, and $Cov\left(V(s), V(s')\right) = \Sigma^0(s,s')$ for possible post-decision states $s, s' \in \mathcal{S}^x$. The prior mean $V^0(s)$ is entirely analogous to the initialization of the value function approximation in the approximate dynamic programming literature. The prior variances reflect our beliefs about the uncertainty of V^0. The covariances can be thought of as a measure of the similarity of two states; our examples in Section IV give one possible way to initialize them. We do not require prior independence in the beliefs about different states, an assumption that is made in [19].

This belief structure will evolve as we make decisions. Let $\mathcal{F}^n$ be the sigma-algebra generated by the first n states we visited and the first n decisions we made. We say that we are "at time n" when this occurs. The notation $\mathbb{E}^n$ denotes the expectation given $\mathcal{F}^n$. Our next assumption will give us a simple way to update our beliefs.

Assumption 2: It is possible to obtain a sample $\hat{R}(S^{x,n}) \sim \mathcal{N}\left(V(S^{x,n}), \lambda(S^{x,n})\right)$ independently of all past samples after making a decision at time n.

This assumption is again carried over from [19]. It is a standard assumption in the optimal learning literature, but does not hold in approximate dynamic programming. We cannot obtain unbiased samples of the true value of a state; the best we can do to update our approximation of $V(S^{x,n})$ is a biased sample of the form

$$\hat{v}^{n+1} = \max_x C\left(S^{n+1}, x\right) + \gamma V^n\left(S^{M,x}\left(S^{n+1}, x\right)\right). \quad (3)$$

In practice, we simply use $\hat{v}^{n+1}$ in lieu of $\hat{R}$, keeping in mind that this observation may not satisfy the assumptions of the Bayesian model. We can choose a value for the variance parameter $\lambda(S^x)$ that reflects our notion of the possible range or spread of values for the observations $\hat{v}$. Since the observations depend on previous value function approximations (constants from our point of view), this quantity will typically be smaller than the prior variances in Σ^0.

Under Assumption 2, we can use standard Bayesian updating equations [20] to obtain a new set of beliefs

$$V^{n+1}(s) = V^n(s) + \frac{\hat{R}(S^{x,n}) - V^n(S^{x,n})}{\lambda(S^{x,n}) + \Sigma^n(S^{x,n}, S^{x,n})} \Sigma^n(s, S^{x,n}) \quad (4)$$

with covariances given by

$$\Sigma^{n+1}(s,s') = \Sigma^n(s,s') - \frac{\Sigma^n(s, S^{x,n})\,\Sigma^n(S^{x,n}, s')}{\lambda(S^{x,n}) + \Sigma^n(S^{x^n}, S^{x,n})} \quad (5)$$

for all $s, s' \in \mathcal{S}^x$. In this manner, we can change our beliefs about any post-decision states that were previously believed to be correlated with $S^{x,n}$. Consequently, we do assume

posterior independence, as [19] does. In the correlated model, a single observation becomes much more powerful than in the independent model.

III. THE KNOWLEDGE GRADIENT POLICY FOR EXPLORATION

In the ranking and selection literature, a measurement decision fixes the type of observation we will get (by specifying a particular reward process), and thus determines the way that we will update our beliefs once we see the observation. In approximate dynamic programming, once we have reached the pre-decision state S^{n+1}, we know that the next observation will be viewed as a sample of the value of $S^{x,n}$, the previous post-decision state. The choice of which post-decision state to observe was made at the previous pre-decision state S^n. Thus, when we make a decision at state S^n, we are implicitly deciding the type of observation that we will receive.

One possible way to make decisions at time n is

$$x^n = \arg\max_x Q^n\left(S^n, x\right) \tag{6}$$

where

$$Q^n\left(S^n, x\right) = C\left(S^n, x\right) + \gamma V^n\left(S^{M,x}\left(S^n, x\right)\right).$$

This is the usual "pure exploitation" policy, which assumes that our current value function approximation is accurate and optimizes based on it. We now present a new policy, based on the method of knowledge gradients, that incorporates exploration into the decision rule by considering the value of information obtained from a single decision.

In [23], the knowledge gradient (KG) concept is described as "choosing the measurement that would be optimal if it were our last chance to learn." Suppose that we are at state S^n at time n, and the next decision will be the last to impact our beliefs, that is, $V^{n'} = V^{n+1}$ for all $n' > n+1$. Under this assumption, the optimal action can be found by computing

$$x^{*,n} = \arg\max_x C\left(S^n, x\right) + \gamma \mathbb{E}_x^n V^{n+1}\left(S^{M,x}\left(S^n, x\right)\right) \tag{7}$$

where $\mathbb{E}_x^n$ is given $\mathcal{F}^n$ and the decision x at time n. Equation (7) is similar to the usual approximate Bellman's equation from (2), but incorporates the fact that the value function approximation will change after action x is taken into the decision-making. Bellman's equation was first used in this learning context by [26]. The KG idea assumes that V^{n+1} will be our final approximation starting at time $n+1$, and allocates the time-n decision to maximize the improvement made by the resulting change from V^n to V^{n+1}.

The change will only occur after we transition from $S^{x,n}$ to S^{n+1} and use (3) to update our approximation. With this in mind, we expand the Q-factor of action x in (7) as

$$\begin{aligned} Q^{*,n}(S^n, x) &= C\left(S^n, x\right) + \gamma \mathbb{E}_x^n V^{n+1}\left(S^{M,x}\left(S^n, x\right)\right) \\ &= C\left(S^n, x\right) \\ &\quad + \gamma \sum_{S^{n+1}} P\left(S^{n+1} | S^{x,n}\right) \mathbb{E}_x^n \max_{x'} Q^{n+1}\left(S^{n+1}, x'\right) \end{aligned}$$

The only reason to follow an exploration strategy is to collect information that may affect our value function approximation. If the approximation becomes fixed at time $n+1$, it is impossible to collect any new information, and we should follow a pure exploitation policy beginning with state S^{n+1}.

The quantity $Q^{n+1}\left(S^{n+1}, x'\right)$, however, depends on $V^{n+1}\left(S^{M,x}\left(S^{n+1}, x'\right)\right)$, which was updated using (4) between time n and time $n+1$. It can be shown [20] that the conditional distribution of $V^{n+1}\left(s\right)$ given $\mathcal{F}^n$ and the decision x at time n, can be written as

$$V^{n+1}\left(s\right) \sim V^n\left(s\right) + \frac{\Sigma^n\left(s, S^{x,n}\right)}{\sqrt{\lambda\left(S^{x,n}\right) + \Sigma^n\left(S^{x,n}, S^{x,n}\right)}} \cdot Z$$

where $Z \sim \mathcal{N}\left(0, 1\right)$. Note that the random variable Z is not indexed by s. In fact, it is common to all the conditional distributions of $V\left(s\right)$ for $s \in \mathcal{S}^x$, reflecting the correlation in our beliefs. As a result, we can write

$$\mathbb{E}^n \max_{x'} Q^{n+1}\left(S^{n+1}, x'\right) = \mathbb{E} \max_{x'}\left(a_{x'}^n + b_{x'}^n Z\right) \tag{8}$$

where

$$a_{x'}^n = C\left(S^{n+1}, x'\right) + \gamma V^n\left(S^{M,x}\left(S^{n+1}, x'\right)\right), \tag{9}$$

$$b_{x'}^n = \gamma \frac{\Sigma^n\left(S^{M,x}\left(S^{n+1}, x'\right), S^{x,n}\right)}{\sqrt{\lambda\left(S^{x,n}\right) + \Sigma^n\left(S^{x,n}, S^{x,n}\right)}}. \tag{10}$$

Due to the correlations, the vector b^n could have all non-zero values even if $S^{M,x}\left(S^{n+1}, x'\right) \neq S^{x,n}$ for all x'.

From the work by [20], it is known that

$$\begin{aligned} \mathbb{E} \max_{x'}\left(a_{x'}^n + b_{x'}^n Z\right) &= \left(\max_{x'} a_{x'}^n\right) \\ &\quad + \sum_{y \in A}\left(b_{y+1}^n - b_y^n\right) f\left(-\left|c_y\right|\right) \end{aligned} \tag{11}$$

where A is the set of all y for which we can find numbers $c_{y-1} < c_y$ for which $y = \arg\max_{x'} a_{x'}^n + b_{x'}^n z$ for $z \in \left(c_{y-1}, c_y\right)$, with ties broken by the largest-index rule. These c_y are the same as the quantities c_y used in (11). The points in the set A are renumbered in order of increasing $b_{x'}^n$. The function f is defined to be $f\left(z\right) = z\Phi\left(z\right) + \phi\left(z\right)$, where ϕ and Φ are the standard Gaussian pdf and cdf, respectively.

Define the *knowledge gradient* to be the difference

$$\begin{aligned} \nu^{KG,n}\left(S^{x,n}, S^{n+1}\right) &= \mathbb{E} \max_{x'}\left(a_{x'}^n + b_{x'}^n Z\right) - \max_{x'} a_{x'}^n \\ &= \sum_{y \in A}\left(b_{y+1}^n - b_y^n\right) f\left(-\left|c_y\right|\right). \end{aligned}$$

This quantity is the expected improvement achieved in our estimate of $\max_{x'} Q^{n+1}\left(S^{n+1}, x'\right)$ by taking action x out of state S^n. Since $a_{x'}^n = Q^n\left(S^{n+1}, x'\right)$, we can now write

$$\begin{aligned} &\sum_{S^{n+1}} P\left(S^{n+1} | S^{x,n}\right) \mathbb{E}_x^n \max_{x'} Q^{n+1}\left(S^{n+1}, x'\right) \\ = &\sum_{S^{n+1}} P\left(S^{n+1} | S^{x,n}\right) \max_{x'} Q^n\left(S^{n+1}, x'\right) \\ + &\sum_{S^{n+1}} P\left(S^{n+1} | S^{x,n}\right) \nu^{KG,n}\left(S^{x,n}, S^{n+1}\right) \end{aligned}$$

We can then write

$$\sum_{S^{n+1}} P\left(S^{n+1}|S^{x,n}\right) \max_{x'} Q^n\left(S^{n+1}, x'\right) = V^n\left(S^{x,n}\right)$$

because V^n is meant to approximate the value of a post-decision state. Consequently, (7) becomes

$$x^{*,n} = \arg\max_x C\left(S^n, x\right) + \gamma V^n\left(S^{x,n}\right) + \gamma \sum_{S^{n+1}} P\left(S^{n+1}|S^{x,n}\right) \nu^{KG,n}\left(S^{x,n}, S^{n+1}\right). \quad (12)$$

This decision rule is very similar to the pure exploitation decision rule in (6), except that an uncertainty bonus in the form of a weighted sum of KG factors is added to $Q^n\left(S^n, x\right)$. This bonus will tend to be larger if the components of Σ^n are larger, indicating more uncertainty, but it will also tend to reward decisions with higher $Q^n\left(S^n, x\right)$. Thus, the KG factor attempts to strike a balance between exploration and exploitation. Decisions with reasonably high Q-factors and high uncertainty may be preferred by the KG policy over the pure exploitation decision.

An algorithm for computing the KG factors exactly can be found in [20]. It is necessary to note that the computational cost of computing the right-hand side of (8) is $\mathcal{O}\left(M^2 \log M\right)$, where M is the size of a^n and b^n. In our setting, the length of a^n and b^n depends on the size of the action space, but not on the size of the state space. Thus, although the KG computation can be fairly costly, our algorithm will scale to problems with large state spaces, as long as the action space is not too large.

In order to compute (12) exactly, we require the transition probabilities $P\left(S^{n+1}|S^{x,n}\right)$. If these probabilities are unknown or difficult to compute, we can compute (12) approximately by simulating K transitions out of $S^{x,n}$. Then,

$$\sum_{S^{n+1}} P\left(S^{n+1}|S^{x,n}\right) \nu^{KG,n}\left(S^{x,n}, S^{n+1}\right) \approx \frac{1}{K} \sum_{k=1}^{K} \nu^{KG,n}\left(S^{x,n}, S_k^{n+1}\right) \quad (13)$$

where $S_k^{n+1} = S^{M,W}\left(S^{x,n}, W^{n+1}\left(\omega_k\right)\right)$ for K different sample paths ω_k. We use this technique in our experiments in Section IV-B.

Finally, the KG rule given in (12) is designed for online problems, where we are solving the dynamic program in real time. In this setting, we are collecting the rewards $C\left(S^n, x^n\right)$ as we make decisions and train the value function approximation. However, many applications take place in an offline setting, where we first use ADP to train our value function approximation offline, and generate the random transitions from $S^{x,n}$ to S^{n+1} in a simulator. The final approximation obtained from this procedure can then be evaluated by first fixing it in place of the true value function in Bellman's equation, then running the resulting policy in real time and observing the result, or estimating the average performance of the policy through simulation. In this offline setting, it is more logical to use an offline version of (12) given by

$$\tilde{x}^{*,n} = \arg\max_x \sum_{S^{n+1}} P\left(S^{n+1}|S^{x,n}\right) \nu^{KG,n}\left(S^{x,n}, S^{n+1}\right), \quad (14)$$

reflecting the distinction between the offline KG rule of [22] and the online KG rule of [24].

IV. NUMERICAL EXAMPLES

We analyzed the performance of the KG policy on an inventory management problem and an energy storage problem. Two types of competing policies were considered:

Bayesian Q-learning (BQL). The Bayesian Q-learning policy of [19] computes the expected gain from learning the true value of the Q-factor $Q\left(S, x\right) = C\left(S, x\right) + \gamma V\left(S^x\right)$. The gain function used for this computation assumes independence of Q-factors. We tested this policy under our correlated Bayesian model, that is, our beliefs were updated using (4) and (5), but the decisions were made under the original independence assumption.

Epsilon-greedy (EG). The ε-greedy policy described in [4] chooses the pure exploitation decision (6) with probability $1-\varepsilon$ and picks an action at random the rest of the time. The parameter ε was tuned separately for the online and offline settings. The policy was tested both under our correlated Bayesian model, and in a traditional ADP setting where stochastic approximation with a stepsize rule $\alpha_n = \frac{k}{k+n}$ with $k = 25$ is used to smooth the observation $\hat{v}^n$ with the previous approximation $V^{n-1}\left(S^{x,n-1}\right)$.

A. Inventory Management

Suppose that the demand D^n for a product on the nth day follows a Poisson distribution with parameter $\mu = 25.0$, and that daily demands are independent. Suppose that the price P^n on day n takes a value in the set $\{7.5, 11.0, 15.0\}$ and evolves according to a Markov chain with transition probability matrix

$$P = \begin{pmatrix} 0.3 & 0.6 & 0.1 \\ 0.25 & 0.5 & 0.25 \\ 0.1 & 0.7 & 0.2 \end{pmatrix}.$$

Let R^n be the amount of inventory held at time n. We assume that R^n can be any integer from 0 to 99. The state of the inventory problem can be written as $S^n = (R^n, P^n, D^n)$. After choosing an order quantity x^n, we move to a post-decision state $S^{x,n} = (R^{x,n}, P^{x,n})$ where

$$\begin{aligned} R^{x,n} &= R^n + x^n, \\ P^{x,n} &= P^n. \end{aligned}$$

We assume that x^n can be any integer between 0 and 49. Let $c = 10.0$ be the cost of ordering one unit of product, and let $h = 5.0$ be the holding cost. The single-period reward can be written as

$$\begin{aligned} C\left(R^{x,n-1}, P^n, D^n\right) &= P^n \cdot \min\left(D^n, R^{x,n-1}\right) - cx^n \\ &- h \cdot \max\left(R^{x,n-1} - D^n, 0\right). \end{aligned}$$

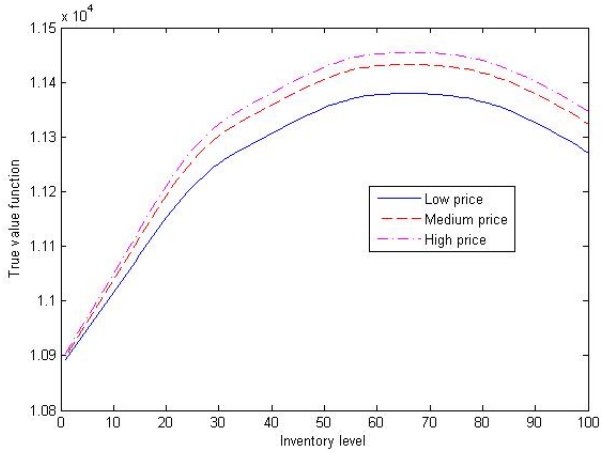

Fig. 1. True value functions for the inventory problem.

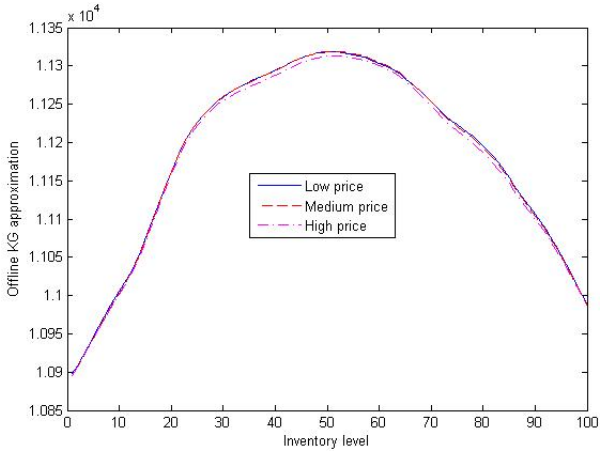

Fig. 2. Value function approximations obtained by offline KG.

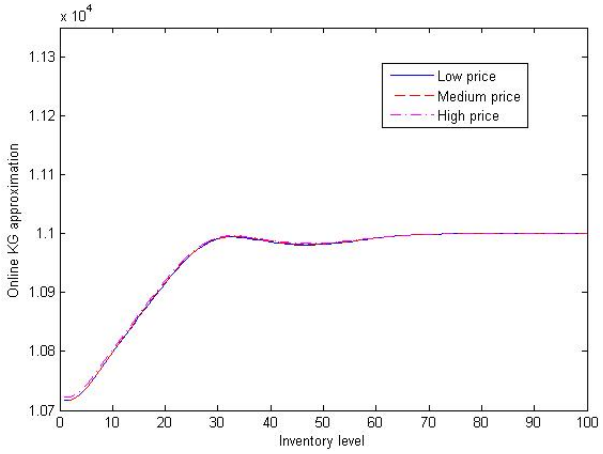

Fig. 3. Value function approximations obtained by online KG.

The transition to the next pre-decision state is made via the equation $R^{n+1} = \max\left(R^{x,n} - D^{n+1}, 0\right)$. The variables D^{n+1} and P^{n+1} evolve as described above. Our objective is to maximize the infinite-horizon discounted reward with a discount factor of $\gamma = 0.99$. In this problem, the size of the post-decision state space $\mathcal{S}^x$ is $3 \cdot 100 = 300$, and the size of the action space is 50.

In this problem, it is necessary to learn the demand and price distributions without ordering too much inventory, due to the high holding cost. It is possible to compute the true value function exactly using (1), because the state space is small and the transition probabilities can be computed exactly using the Poisson cdf. Fig. 1 shows the true value functions for all three price levels.

We initialize our Bayesian model by letting $V^0(s) = 11000$ for every s. The initial covariances are set using a power-exponential function:

$$\Sigma^0(s, s') = 500^2 e^{-0.01\left[(r-r')^2 + (\tilde{p}-\tilde{p}')^2\right]}. \qquad (15)$$

For a state $s = (r, p)$, the quantity $\tilde{p}$ is set to 0, 1 or 2 depending on whether p is low, medium or high. The coefficient 500^2 signifies that we believe $V(s)$ to be in the interval $V^0(s) \pm 2 \cdot 500$. The covariance between two states is larger if those states are close together in terms of price and inventory levels. The sample variance was set to $\lambda(s) = 150^2$, and the starting state was chosen such that $R^0 = 0$ and P^0 was the medium price.

Each policy was run for 150 iterations. This was repeated with 10^3 different sample paths, which were divided into lots of 50 to obtain approximately normal samples of the average online objective value achieved by each policy. The policies were then evaluated in an offline setting by simulating the average final value function approximations obtained from the online runs on 10^5 sample paths, which were divided into lots of 1000. Table I reports the means and average standard errors obtained for both the online and offline values of the different policies.

TABLE I
MEANS AND STANDARD ERRORS FOR THE INVENTORY PROBLEM.

	Offline objective		Online objective	
	Mean	Avg. SE	Mean	Avg. SE
Offline KG	10639.28	3.45	-1644.99	59.90
Online KG	10188.41	3.16	3961.38	31.44
BQL	10632.58	3.10	3819.69	46.15
Bayes EG	10586.56	3.29	3328.39	35.28
ADP EG	9626.17	3.22	2448.67	23.98

The KG policies achieve the best performance in their respective settings. Offline KG performs poorly in the online setting, but obtains the best value function approximation. Online KG yields the best online performance, but loses to Bayesian Q-learning and ϵ-greedy policies offline. The Bayesian Q-learning policy of [19] performs competitively against KG, though KG wins by a relatively small margin.

Figs. 2-6 show the final value function approximations obtained from each policy, used to evaluate the policies offline. Offline KG is the only policy that captures the general shape of the value function as having a clear global maximum, and the only one to discover that certain states have values greater than 11000. The other policies obtain approximations with small local peaks, and do most of

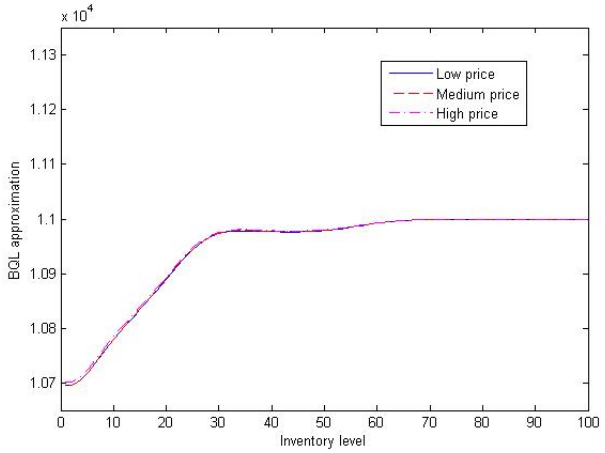

Fig. 4. Value function approximations obtained by Bayesian Q-learning.

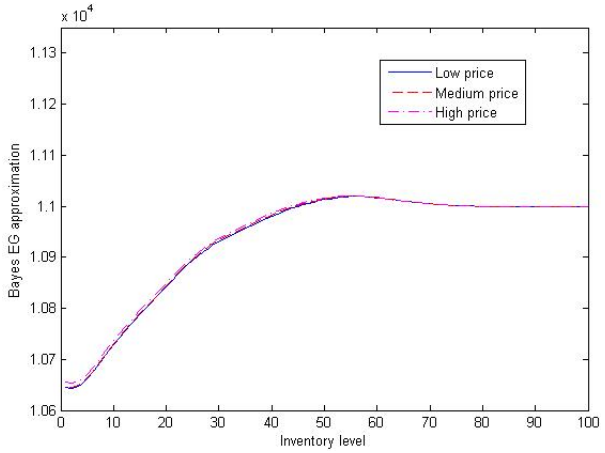

Fig. 5. Value function approximations obtained by Bayesian ε-greedy.

their exploration on states corresponding to lower inventory levels (< 50). The approximations for higher inventory levels mostly remain at their starting values of 11000. Fig. 6 shows the approximations obtained by the ADP ε-greedy policy for the value of ε with the best offline performance. This policy is able to capture some of the difference between price levels (which the other policies were unable to do), but still achieves poor performance in both online and offline cases.

B. Energy Storage

We also considered an energy storage problem with a continuous state variable. Suppose that we have five sodium-sulfur batteries, each with an energy capacity of 7 MWh and a power of 1 MW, meaning that 1 MWh per battery can be discharged in one hour. The spot price P^n is assumed to follow a geometric Ornstein-Uhlenbeck process [27]. When time is discretized into hours, this means that

$$\log \frac{P^{n+1}}{P^0} - \log \frac{P^n}{P^0} = -\alpha \log \frac{P^n}{P^0} + \sigma Z^{n+1} \qquad (16)$$

for $Z^{n+1} \sim \mathcal{N}(0,1)$. We used values of $\alpha = 0.0633$ and $\sigma = 0.2$ in our experiments, as well as an initial price $P^0 = 30$. Let R^n denote the current charge level (in MWh) of the batteries; we assume that R^n is an integer between 0 and 35. In the value function approximation, the scaled log-price $\log \frac{P^n}{P^0}$ is discretized into 18 intervals of width 0.25 between -2 and 2. The state of the problem is $S^n = (R^n, P^n)$ and the post-decision state is given by

$$\begin{aligned} R^{x,n} &= R^n + x^n, \\ P^{x,n} &= P^n. \end{aligned}$$

The random transition to the next pre-decision state consists of updating the price using (16). Thus, the size of $\mathcal{S}^x$ is $18 \cdot 36 = 648$. We assume that demand is infinite; thus, our revenue depends only on R^n and the amount x^n that we wish to charge or discharge. The decision x^n is integer-valued and can be as low as -5 and as high as $+5$. The single-period reward is given by

$$C(P^n, x^n) = -P^n x^n.$$

A positive decision corresponds to charging the battery, which incurs a cost, whereas discharging the battery produces revenue. We assume that there are no other costs in the problem; thus, the main challenge of the problem is to learn the price distribution and understand when the price is low enough to buy or high enough to sell.

While the value function approximation uses a discretized state space, we keep track of the continuous price as we step forward through time. We simulate P^n using the continuous transition function given in (16), and discretize the price when it is necessary to call the value function approximation. This issue is discussed in [6]. For our purposes, the significance of the continuous price is that we can no longer solve

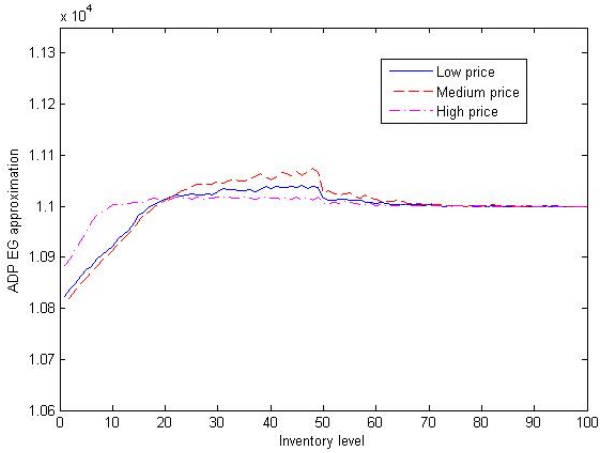

Fig. 6. Value function approximations obtained by ADP ε-greedy.

TABLE II
MEANS AND STANDARD ERRORS FOR THE STORAGE PROBLEM.

	Offline objective		Online objective	
	Mean	Avg. SE	Mean	Avg. SE
Offline KG	208.39	0.33	-260.59	16.86
Online KG	68.10	0.24	155.30	6.01
BQL	133.65	0.31	76.03	1.87
Bayes EG	85.65	0.31	67.10	3.00
ADP EG	154.47	0.35	7.09	2.57

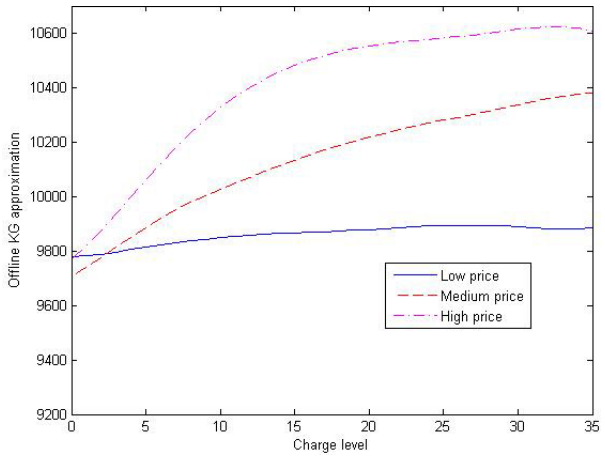

Fig. 7. Value function approximations obtained by offline KG.

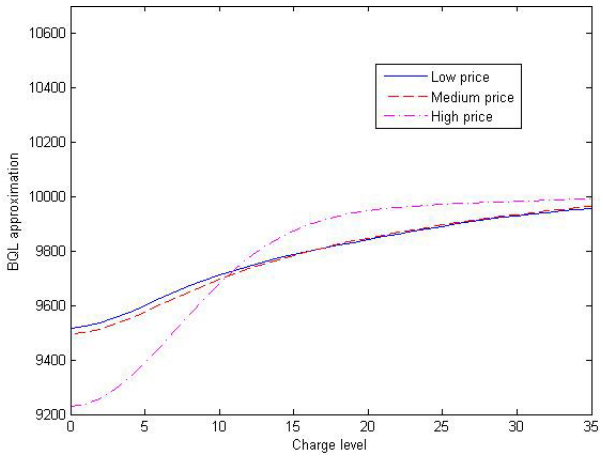

Fig. 9. Value function approximations obtained by Bayesian Q-learning.

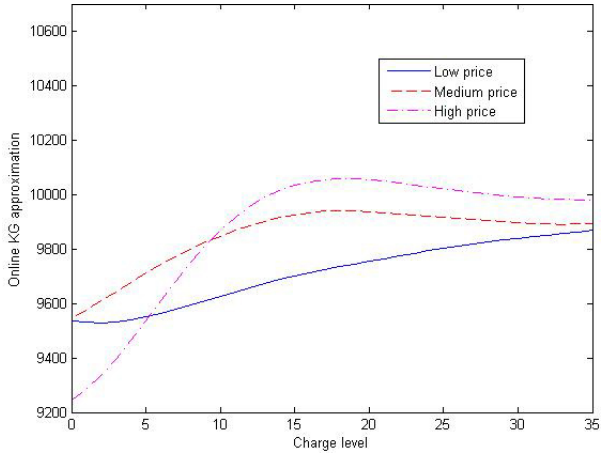

Fig. 8. Value function approximations obtained by online KG.

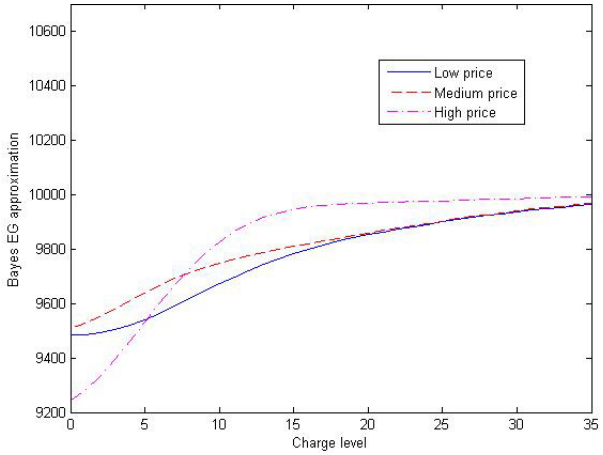

Fig. 10. Value function approximations obtained by Bayesian ε-greedy.

the problem exactly using (1), nor can we compute the exact KG factors in (12). We use the approximate KG factor from (13) with $K = 20$. The value function approximation was initialized to a high value with $V^0(s) = 10^4$. The actual performance values obtained from the policies were much smaller; Sec. 4.7 of [6] discusses the benefits of optimistic initial estimates. The initial covariances were set using (15) with a coefficient of 1000^2 in front. The sample noise was taken to be $\lambda(s) = 200^2$.

Table II reports the results obtained for the different policies in both offline and online settings. In the storage problem, the KG approach appears to have a clear advantage. Offline KG outperforms all other policies by a convincing margin in the offline case, while performing poorly online; similarly, online KG achieves the worst offline result, but significantly outperforms the competition online. The Bayesian Q-learning policy does not maintain the competitive performance that we observed for the inventory problem.

Figs. 7-11 show the value function approximations obtained by each policy for the lowest, middle and highest prices (out of 18 price levels in all). Offline KG makes the biggest distinction between price levels. The value appears to increase with the charge level for each price. This is expected, because the problem has no holding cost, and it is possible to hold on to energy instead of selling it, so it is always better to have more in storage. However, the value of a low charge level will be lower if the price is higher. If the batteries are empty and the price is high, it will take several time steps to charge them, resulting in multiple missed opportunities to sell. By the time the batteries are full, the price process is likely to have reverted to the mean.

As before, the other policies do most of their exploration on states with low charge levels. The lack of exploration is particularly noticeable in the case of the ADP policy. The lowest and highest price levels occur with low probability; the price process spends most of its time around the middle. Thus, states with these extreme price levels are rarely visited. As a result, the ADP policy never changes its estimate of the values of such states. The Bayesian policies, however, use correlated learning to update these estimates.

Our experiments on the storage problem provide reassuring evidence that the KG policy still performs well when the KG factors cannot be computed exactly, requiring us to use the approximation from (13). The results suggest that KG

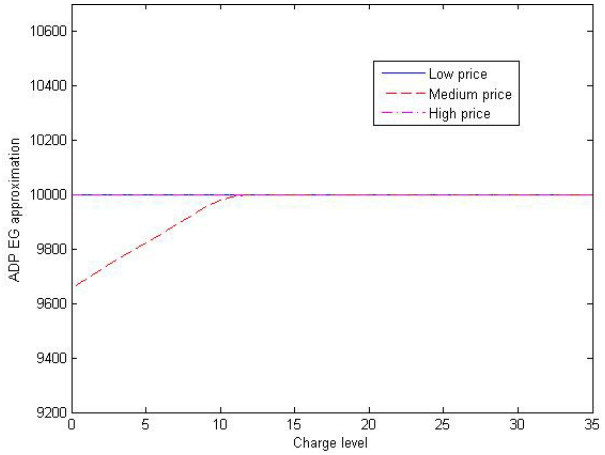

Fig. 11. Value function approximations obtained by ADP ε-greedy.

can retain its usefulness in situations where the distribution function of the random component W^{n+1} of the transition function $S^{M,W}$ is difficult to compute, but random transitions can still be simulated.

V. CONCLUSION

We have proposed a new Bayesian model for approximate dynamic programming in which our beliefs about the values of different states are allowed to be correlated. The model builds on work by [19], but makes fewer assumptions and allows us to obtain much more information out of a single decision. We have also proposed a policy for making decisions in this model based on the knowledge gradient approach from the optimal learning literature. Variations of the KG idea allow us to handle both offline and online problems. We have conducted experiments on two important problem classes. The results indicate that the KG policy yields better performance given the same number of iterations as other techniques, even when the KG factors are computed approximately. We believe that the KG approach, used in conjunction with a correlated learning model, is a promising approach to the issue of exploration vs. exploitation in approximate dynamic programming.

REFERENCES

[1] R. Bellman, *Dynamic Programming*. Princeton: Princeton University Press, 1957.

[2] M. L. Puterman, *Markov Decision Processes*. New York: John Wiley & Sons, 1994.

[3] H. Geman and A. Roncoroni, "Understanding the Fine Structure of Electricity Prices," *The Journal of Business*, vol. 79, no. 3, 2006.

[4] R. Sutton and A. Barto, *Reinforcement Learning*. Cambridge, Massachusetts: The MIT Press, 1998.

[5] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[6] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. New York: John Wiley and Sons, 2007.

[7] B. Van Roy, D. Bertsekas, Y. Lee, and J. Tsitsiklis, "A neuro-dynamic programming approach to retailer inventory management," in *Proceedings of the 36th IEEE Conference on Decision and Control*, vol. 4, 1997, pp. 4052–4057.

[8] H. Kushner and G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*, 2nd ed. Springer, 2003.

[9] L. P. Kaelbling, *Learning in Embedded Systems*. Cambridge, MA: MIT Press, 1993.

[10] R. Bechhofer, T. Santner, and D. Goldsman, *Design and Analysis of Experiments for Statistical Selection, Screening and Multiple Comparisons*. New York: John Wiley and Sons, 1995.

[11] J. Gittins, *Multi-Armed Bandit Allocation Indices*. New York: John Wiley and Sons, 1989.

[12] E. Silver, "Markovian decision processes with uncertain transition probabilities or rewards," Operations Research Center, MIT, Tech. Rep. 1, 1963.

[13] M. Duff and A. Barto, "Local bandit approximation for optimal learning problems," *Advances in Neural Information Processing Systems*, vol. 9, pp. 1019–1025, 1996.

[14] R. Dearden, N. Friedman, and D. Andre, "Model-based Bayesian Exploration," in *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, 1999, pp. 150–159.

[15] P. Poupart, N. Vlassis, J. Hoey, and K. Regan, "An analytic solution to discrete Bayesian reinforcement learning," in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 697–704.

[16] M. Strens, "A Bayesian framework for reinforcement learning," in *Proceedings of the 17th International Conference on Machine Learning*, 2000, pp. 943–950.

[17] M. Duff, "Design for an optimal probe," in *Proceedings of the 20th International Conference on Machine Learning*, 2003, pp. 131–138.

[18] S. Mannor, D. Simester, P. Sun, and J. Tsitsiklis, "Bias and variance approximation in value function estimates," *Management Science*, vol. 53, no. 2, pp. 308–322, 2007.

[19] R. Dearden, N. Friedman, and S. Russell, "Bayesian Q-learning," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 1998, pp. 761–768.

[20] P. I. Frazier, W. B. Powell, and S. Dayanik, "The knowledge-gradient policy for correlated normal rewards," *INFORMS J. on Computing*, vol. 21, no. 4, pp. 599–613, 2009.

[21] S. Gupta and K. Miescke, "Bayesian look ahead one-stage sampling allocations for selection of the best population," *Journal of Statistical Planning and Inference*, vol. 54, no. 2, pp. 229–244, 1996.

[22] P. I. Frazier, W. B. Powell, and S. Dayanik, "A knowledge gradient policy for sequential information collection," *SIAM Journal on Control and Optimization*, vol. 47, no. 5, pp. 2410–2439, 2008.

[23] I. O. Ryzhov and W. B. Powell, "The knowledge gradient algorithm for online subset selection," in *Proceedings of the 2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, Nashville, TN*, 2009, pp. 137–144.

[24] I. O. Ryzhov, W. B. Powell, and P. I. Frazier, "The knowledge gradient algorithm for a general class of online learning problems," *Submitted for publication*, 2009.

[25] I. O. Ryzhov and W. B. Powell, "Information collection on a graph," *Operations Research (to appear)*, 2010.

[26] M. H. DeGroot, *Optimal Statistical Decisions*. Hoboken, NJ: John Wiley and Sons, 1970.

[27] E. Schwartz, "The stochastic behavior of commodity prices: Implications for valuation and hedging," *Journal of Finance*, vol. 52, no. 3, pp. 923–973, 1997.