Bayesian exploration for approximate dynamic programming

Ilya O. Ryzhov  Martijn R.K. Mes  Warren B. Powell  Gerald A. van den Berg

July 22, 2015

Abstract

Approximate dynamic programming (ADP) is a general methodological framework for multi-stage stochastic optimization problems in transportation, finance, energy, and other applications where scarce resources must be allocated optimally. We propose a new approach to the exploration/exploitation dilemma in ADP. First, we show how a Bayesian belief structure can be used to express uncertainty about the value function in ADP. Bayesian models can be easily integrated into both parametric and non-parametric value function approximations. Second, we propose a new exploration strategy, based on the concept of value of information from the optimal learning literature, for systematic exploration of the state and action space. Using several resource allocation problems, we demonstrate that the performance of our method is highly competitive against other exploration strategies, with the benefits being particularly valuable in online implementations where computation time is less restrictive.

1 Introduction

Approximate dynamic programming (ADP) provides a powerful and general framework for solving large-scale, complex stochastic optimization problems (Powell, 2011; Bertsekas, 2012). This methodology, also known as reinforcement learning (Sutton & Barto, 1998) or neuro-dynamic programming (Bertsekas & Tsitsiklis, 1996), combines statistical modeling, optimization, and Monte Carlo simulation to find approximate solutions to Bellman’s equation (Bellman, 1957) when the curse of dimensionality makes exact solutions computationally intractable. ADP thus allows us to create good practical strategies for multi-stage decision making under uncertainty, while overcoming the well-known computational hurdles of dynamic programming.

A major area of application for ADP is the broad class of resource allocation problems, spanning energy, finance, transportation and other fields. Examples include multicommodity flows (Topaloglu & Powell, 2006), asset acquisition (Nascimento & Powell, 2009), and sensor management (Castanon, 1997). ADP has been especially successful in large-scale dynamic fleet management (Simão et al., 2009, 2010) where it is possible to exploit the additional structure of convexity. Numerous other applications are relatively low-dimensional, but challenging to solve. These include:
1. **Commodity storage.** A firm stores a commodity such as electricity or natural gas (Lai et al., 2010; Löhndorf & Minner, 2010; Secomandi, 2010) in a storage device, and seeks to optimally trade the commodity on the spot market subject to a stochastic price or supply process.

2. **Water management.** In a water reservoir network, it is necessary to determine how much water to release from each basin to minimize long-term service and shortage costs (Cervellera et al., 2006).

3. **Cash management.** A mutual fund decides how much cash to keep on hand to meet shareholder redemptions (Nascimento & Powell, 2010). Holding too much cash incurs an opportunity cost due to lost investment opportunities.

4. **Service operations.** A call center assigns different classes of agents to jobs with stochastic arrival and service times (Koole & Pot, 2005).

5. **Health care.** A health care provider has to design a treatment plan that will dynamically adapt to the patient’s health condition (He et al., 2012).

6. **Robotics.** An artificial intelligence agent learns to interact with its environment, governed by nonlinear physical dynamics (Lagoudakis et al., 2002).

Many problems present a fundamental challenge to the successful implementation of ADP (or other approximate methods), known as the “exploration vs. exploitation” dilemma (see ch. 12 of Powell, 2011). Any approximate solution strategy involves some degree of experimentation; a decision that seems to be suboptimal based on past observations may actually be much better than we believe (and vice versa). Each individual decision carries immediate economic benefits (e.g., discharging a battery to generate revenue), but it also provides information that can be used to update our approximations, allowing us to make better decisions in the future. We must strike a balance between making decisions based solely on existing approximations, and making decisions with uncertain outcomes to learn about the problem. The importance of this issue has long been recognized by the research community: see, e.g., Kaelbling et al. (1996) and Bertsekas (2012). The literature contains many general exploration heuristics, ranging from simple methods such as Boltzmann exploration and epsilon-greedy (Sutton & Barto, 1998) to more sophisticated techniques such as interval estimation (Kaelbling, 1993), $E^3$ (Kearns & Singh, 2002) and R-max (Brafman & Tennenholtz, 2003). These methods can perform well on small, discrete problems, but may require tuning or run into scaling issues in higher dimensions.
The exploration/exploitation dilemma has been extensively studied in what we collectively refer to as optimal learning, a set of problems specifically emphasizing the efficient collection of information, usually with a simple, streamlined underlying optimization framework. In ranking and selection (Kim & Nelson, 2006; Hong & Nelson, 2009) and multi-armed bandit (Berry & Fristedt, 1985) problems, we have finitely many alternatives (e.g., supply chain designs, experimental medical treatments, or settings for a simulator) with unknown values, and we must use a limited number of experiments to efficiently learn the best value. Experiments can occur inside a simulator (Goldsman, 1983; Kim & Nelson, 2007), or in real time with an immediate reward, as in market pricing (Rothschild, 1974; Chhabra & Das, 2011) or Internet advertising (Babaioff et al., 2009). In the setting of optimizing over a finite set, there are numerous sophisticated algorithms for information collection, including Gittins indices (Gittins et al., 2011), upper confidence bound methods (Auer et al., 2002), Thompson sampling (Russo & Van Roy, 2014), and optimal computing budget allocation (He et al., 2007).

Despite the depth of the optimal learning literature, it is difficult to directly apply it to general stochastic optimization problems, due to the added complication of a physical state. In addition to providing rewards and information, a decision also changes the set of decisions available to us in the future, possibly many time periods later. While there have been some heuristic adaptations of multi-armed bandit techniques for ADP (Duff & Barto, 1996; Szepesvári, 2010), bandit theory usually does not engage with the physical state directly. To give an example of the difficulties that arise, a common assumption in optimal learning is that the values of alternatives are independent, whereas decisions in ADP are heavily correlated (a storage device that is 98% full is likely to be similar in value to one that is 99% full). It is possible to address such issues theoretically, but this often requires assumptions such as finite state spaces (Kearns & Singh, 2002), deterministic transitions (Wen & Van Roy, 2013), or known correlations (Pazis & Parr, 2013), whereas practical implementations of ADP almost always require the ability to generalize to a variety of algorithmic representations of the state space.

We propose a new framework that bridges the gap between ADP and optimal learning with the concept of value of information (Chick, 2006), also known by the name “knowledge gradient” (Powell & Ryzhov, 2012). First, we use a Bayesian belief structure to express our uncertainty about the unknown value function in ADP. While Bayesian thinking has previously been applied in reinforcement learning (Dearden et al., 1998), our model includes the crucial dimension of correlated
beliefs in order to quantify similarities and differences between various decisions; as a result, a single decision may now provide information about a much larger portion of the state space, greatly increasing our ability to explore. Furthermore, our Bayesian framework can be incorporated into lower-dimensional representations, such as parametric models, and thus can scale to much larger problems than existing methods. We then use the Bayesian model to make probabilistic forecasts about the outcomes of different decisions, allowing us to compare them based on their potential to provide useful information. The best decision can be identified via a tractable algorithm that also extends to sophisticated approximation architectures.

This paper makes the following contributions: 1) We propose a correlated Bayesian model for learning the value function in a dynamic program, and show how it can be used in conjunction with sophisticated approximation structures such as basis functions and hierarchical aggregation. 2) We explicitly compute the tradeoff between exploration and exploitation, deriving knowledge gradient algorithms for offline and online ADP. While our approach is based on a concept from optimal learning, we adapt the concept and create a completely new technique designed specifically for ADP. 3) We prove that the offline method systematically explores most (in some cases, all) of the state space. Proofs of convergence tend to be rare for algorithms based on approximate value iteration. 4) We show that this method is highly competitive against other exploration strategies on several test problems based on some of the motivating applications discussed previously.

We proceed as follows. Section 2 presents the key concepts of the paper, the correlated Bayesian model and the knowledge gradient policy, using a basic dynamic program with discrete states and actions. Section 3 analyzes the asymptotic behaviour of this policy in the discrete setting. In Section 4, we show how these concepts can be used together with more powerful value function approximations that can handle continuous, multi-dimensional state spaces. Section 5 demonstrates the performance of our method on several example problems. Our presentation emphasizes the flexibility of the knowledge gradient approach, and provides insights into implementation.

2 Main concepts: model and algorithm

We first give a brief description of a generic ADP algorithm along the lines of Powell (2011), ch. 4. We then present our correlated Bayesian model for learning the value function in ADP, followed by the knowledge gradient policy for making decisions. To make the main concepts as clear as
possible, we derive the algorithm in a classic Markov decision process (Puterman, 1994) setting, with discrete state and action spaces. Later, in Section 4, we explain how the same ideas can be used in multi-dimensional and continuous ADP problems.

2.1 Overview of ADP

Consider a dynamic program with state space \( \mathcal{S} \) and decision space \( \mathcal{X} \). Let \( C \) be a function mapping a state \( S \in \mathcal{S} \) and an action \( x \in \mathcal{X} \) to a deterministic reward \( C(S, x) \). Our objective is to maximize the infinite-horizon reward

\[
\sup_{\pi} \sum_{n=0}^{\infty} \gamma^n C(S^n, X^{\pi,n}(S^n))
\]

with a discount factor \( \gamma \in (0, 1) \). The notation \( X^{\pi,n} \) represents a decision rule (a function mapping states to actions) associated with some policy \( \pi \). The optimal value of the problem is described by Bellman’s equation,

\[
V(S) = \max_x C(S, x) + \gamma \mathbb{E}(V(S') | S, x) \quad \text{for all } S \in \mathcal{S},
\]

and the optimal policy makes the decision that maximizes the right-hand side of (1). The expectation in (1) is over the random transition from \( S \) to \( S' \), given action \( x \).

In most practical problems, we cannot solve (1) exactly due to the curse of dimensionality. A popular strategy in ADP is to step forward in time and compute an approximate observation

\[
\hat{v}^n = \max_x C(S^n, x) + \gamma \bar{V}^{n-1}(S^{M,x}(S^n, x)),
\]

where \( S^n \) is the state of the dynamic program in the \( n \)th time step. The function \( \bar{V}^{n-1} \) is an approximation, whereas \( S^{M,x} \) is a deterministic function that maps \( S^n \) and \( x \) to the post-decision state \( S^{x,n} \) (see chapters 4-5 of Powell, 2011), which represents the change in the state of the system after the decision \( x \) has been made, but before any random information (denoted by \( W^{n+1} \)) is observed. The next pre-decision state thus becomes \( S^{n+1} = S^{M,W}(S^{x,n}, W^{n+1}) \).

We can recast the optimal value function around the post-decision state,

\[
V(S^{x,n}) = \mathbb{E}(V(S^{n+1}) | S^n, x^n),
\]

and use \( \bar{V}^{n-1} \) to approximate the expectation. The quantity \( \hat{v}^n \) is then interpreted as an approximate observation of the value \( V(S^{x,n-1}) \) at the previous post-decision state. We use some
appropriate statistical technique (traditionally a form of stochastic approximation, see Kushner & Yin, 2003) to update $\bar{V}^{n-1} (S^{x,n-1})$ with $\hat{v}^n$, creating a new approximation $\bar{V}^n$.\footnote{In classical methods such as Q-learning (Watkins & Dayan, 1992), each post-decision state is updated independently and $\hat{v}^n$ is only used to update $\bar{V}^{n-1} (S^{x,n-1})$. Ribeiro & Szepesvári (1996) and Szepesvári & Littman (1999) introduce a heuristic form of dependence between states into Q-learning, but this scheme requires tuning of a stepsize.} After this update is complete, we choose and implement some action $x^n$. We then transition from $S^n$ to $S^{M,x} (S^n, x^n)$, and from there to $S^{n+1}$. If we have a stochastic simulation model of the underlying system, we can generate transitions by Monte Carlo simulation of $W^{n+1}$.

Most practical implementations of ADP use off-policy learning (Sutton et al., 2009), meaning that they treat the calculation of $\hat{v}^n$ as being completely independent from the subsequent action $x^n$. We always use (2) to calculate the approximate observation $\hat{v}^n$ and update $\bar{V}^{n-1}$, but we can then use a completely different algorithm to calculate $x^n$. As long as $\hat{v}^n$ is calculated using (2), any other policy can be used to explore the state space. The problem of designing efficient exploration policies is the main focus of this paper.

We also note that there are two different ways to evaluate an ADP algorithm. We refer to these as online and offline, which should not be confused with on- and off-policy. Online evaluation (which can be applied with either on- or off-policy algorithms) means that the value function approximation is trained in real time; we are actually solving the problem, making decisions, and collecting rewards while $\bar{V}^{n-1}$ is being updated. By contrast, offline evaluation means that we have access to a simulation model of the underlying process, and train ADP inside the simulator. Then, after a sufficient amount of time (say, $N$ iterations), we fix the policy

$$X^{fixed} (S) = \arg \max_x C (S, x) + \gamma \bar{V}^N (S^x),$$

and evaluate the objective value $\sum_{n=0}^{\infty} \gamma^n C (S^n, X^{fixed} (S^n))$ either inside the simulator or in real time. Online learning requires us to make decisions more conservatively, because a poor decision can severely impact immediate and future rewards. By contrast, offline learning allows for more experimentation, since the only real penalty for bad decisions is lost simulation time.

Generally, the ADP literature does not make the distinction between online and offline learning, but many applications (e.g., Simão et al., 2010) are offline (even though the algorithms may be off-policy). However, this distinction is quite important for understanding the value of exploration and comparing different algorithms.
2.2 ADP with correlated beliefs

In ADP, the decision \( x^\mathfrak{n} \) obviously has an impact on our next state \( S^{n+1} \), but it also affects the next observation \( \hat{v}^{n+1} \), and through that, the future approximation \( \bar{V}^{n+1} \). Thus, if we have some probabilistic model or forecast of the value of \( \hat{v}^{n+1} \), we could explicitly take this into account when making decision \( x^n \). In this section we describe such a methodology using a Bayesian belief structure.

In the Bayesian philosophy, any unknown quantity is a random variable whose distribution reflects our prior knowledge or belief about the quantity. In ADP, the most important unknown quantity is the value function itself. We place a multivariate Gaussian prior, with mean \( \bar{V}^0 \) and covariance matrix \( \Sigma^0 \), on the value function \( V \). Letting \( S^x \) be the space of possible post-decision states, we have \( E[V(S^x)] = \bar{V}^0(S^x) \) and \( Cov(V(S^x), V(S^y)) = \Sigma^0(S^x, S^y) \) for all possible \( S^x, S^y \in S^x \).

The prior mean \( \bar{V}^0 \) is the usual initial value function approximation required by ADP algorithms. The covariances represent our beliefs about the similarity of different states. For example, if \( S^x \) is a continuous Euclidean space, we can use the power-exponential correlation structure

\[
\Sigma^0(S^x, S^y) = \beta \exp^{-\alpha \|S^x - S^y\|^2_2},
\]

where \( \beta > 0 \) is an initial variance, and \( \alpha > 0 \) represents the spread of correlation over distance. This is a very heuristic covariance structure, and \( \alpha, \beta \) may require tuning. Nonetheless, it captures our intuition that the values of two states should be more heavily correlated if those states are closer together. For now, we assume that we have some initial guess of the correlations in order to illustrate our main concept. In Section 4, we will discuss other models that learn the correlation structure automatically. We now introduce a modeling assumption that will simplify the computation of our exploration algorithm.

**Assumption 2.1.** The ADP observation \( \hat{v}^{n+1} \) has the distribution \( \mathcal{N}(V(S^{x,n}), \sigma^2) \) and is independent of past observations.

A version of this assumption originally appeared in Dearden et al. (1998), where normality is justified with a central limit argument. A similar assumption is used by Engel et al. (2003, 2005) to learn the value of a fixed policy. Separately from normality, however, there is also the fact that, from (2), \( \hat{v}^{n+1} \) depends on \( \bar{V}^n \), introducing an obvious bias that is not considered by the assumption. This issue is common to ADP algorithms in general, because they almost always rely
on biased estimates of future values to construct new observations of the value of being in a state. The problem of estimating this bias is discussed to some extent in George & Powell (2006); see also Frazier et al. (2009b) for an algorithmic technique that uses estimates of the projected growth of the value function to reduce the bias. In practice, one may also conduct a set of prior runs to obtain a set of prior estimates where the bias was reduced. Finally, we note that Assumption 2.1 treats the variance $\sigma^2_\varepsilon$ as a known, state-independent constant; in practice, we would view $\sigma^2_\varepsilon$ as a tunable parameter of the model.

Assumption 2.1 has two very useful consequences. First, it provides a simple, computationally efficient way to update our entire approximation $\bar{V}^n$ using a single scalar observation $\hat{v}^{n+1}$, which is not the case for the independent normal-gamma priors used in Dearden et al. (1998). With Assumption 2.1, we can apply standard Bayesian updating equations (Powell & Ryzhov, 2012) to obtain a new set of beliefs

$$\bar{V}^{n+1}(S^x) = \bar{V}^n(S^x) + \frac{\hat{v}^{n+1} - \bar{V}^n(S^x,S^{x,n})}{\sigma^2_\varepsilon + \Sigma^n(S^x,n, S^{x,n})} \Sigma^n(S^x, S^{x,n}),$$

(6)

$$\Sigma^{n+1}(S^x, S^y) = \Sigma^n(S^x, S^y) - \frac{\Sigma^n(S^x, S^{x,n}) \Sigma^n(S^{x,n}, S^y)}{\sigma^2_\varepsilon + \Sigma^n(S^{x,n}, S^{x,n})},$$

(7)

for each post-decision state $S^x \in S^x$. As we shall see in Section 4, this update can be viewed as a form of recursive least squares (a widely used strategy for modeling value functions in ADP), which may help to put Assumption 2.1 in context.

Second, Assumption 2.1 allows us to make probabilistic forecasts about the future approximation $\bar{V}^{n+1}$ that will be obtained as a consequence of our next action $x^n$. Equation (7) gives an explicit expression for the uncertainty reduction that will be achieved by visiting the post-decision state $S^{x,n}$. We can use such forecasts to compare different post-decision states before deciding to visit one of them. Our exploration strategy in Section 2.3 uses this approach to derive a computationally tractable expression for the value of information that can be calculated and efficiently implemented in a wide variety of problems, where Assumption 2.1 would not be expected to hold. The convergence analysis in Section 3 works with the updating procedure in (6)-(7) directly, but does not explicitly rely on normality.

Note that $\bar{V}^n$ and $\Sigma^n$ completely characterize our Bayesian distribution of belief about the value function after we arrive in state $S^n$, but before the next action $x^n$ is implemented. The quantity $K^n = (\bar{V}^n, \Sigma^n)$ is variously called the belief state in reinforcement learning (Kaelbling et al., 1996).
or the knowledge state in optimal learning (Ryzhov et al., 2012). This is distinct from the physical state $S^n$ of the system at time $n$, which affects the set of actions available to us and the total reward that we can collect. However, both $S^n$ and $K^n$ are used to make decisions, and thus $K^n$ can be considered part of the state variable. The work by Bellman & Kalaba (1959) combines these concepts into a “hyperstate” $H^n = (S^n, K^n)$.

We briefly note that (6) and (7) are computationally expensive if the state space is large, since we are required to store values $\bar{V}^n(S^x)$ and $\Sigma^n(S^x, S^y)$ for all post-decision states $S^x$ and $S^y$. We refer to this as a “lookup-table” approximation. If $S^x$ is continuous, we cannot store a value for every post-decision state, and some form of discretization is required. Again, we defer this issue until Section 4, where we explain how the correlated Bayesian approach can be combined with more sophisticated value function approximations.

### 2.3 The knowledge gradient policy for exploration

The crux of our paper is how the correlated Bayesian model can be used to value information in ADP, leading to a new policy for balancing exploration with exploitation. We use the simple lookup-table approximation from Section 2.2 to illustrate the concept, but we will later extend it to parametric and non-parametric belief models in Section 4. For now, let $P(S'|S, x)$ denote the probability of transitioning to $S'$ after taking action $x$ out of state $S$, and let

$$Q(S, x) = C(S, x) + \gamma \sum_{S'} P(S'|S, x) V(S')$$

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

be the true and estimated values, respectively, of the state-action pair $(S, x)$. Recall that the true value function satisfies $V(S) = \max_x Q(S, x)$.

We now study the problem of choosing an action out of state $S^n$ at time $n$. Recalling the sequence of events from Section 2.1, the choice of action occurs after we have calculated $\hat{v}^n$ and run the update (6)-(7). Thus, the action $x^n$ can depend on the most recent value function approximation $\bar{V}^n$. Consider the following one-period look-ahead policy: we assume that this next action $x^n$ will be our last chance to learn, and optimize based on this assumption. That is, if we are in state $S^n$, we assume that we will update our beliefs one more time, after the next transition to $S^{n+1}$, but then no further learning will occur and $\bar{V}^{n'} = \bar{V}^{n+1}$ for all $n' \geq n + 1$. For any action $y$ that we
may consider out of the future state $S^{n+1}$, our best estimate of the value of $S^{y,n+1}$ will be fixed at $\tilde{V}^{n+1} \left(S^{M,x} \left(S^{n+1}, y \right)\right)$.

Under this assumption, the optimal choice of action out of the current state $S^n$ is given by

$$X^{KG,n} (S^n, K^n) = \arg \max_x C (S^n, x) + \gamma \mathbb{E}^n_x \sum_{S^{n+1}} P \left(S^{n+1} | S^n, x \right) \max_y \tilde{Q}^{n+1} \left(S^{n+1}, y \right).$$

(8)

On the right-hand side of (8), we assume that $\tilde{V}^{n+1}$ will be used to approximate the downstream value of the future action $y$ out of $S^{n+1}$. However, $\tilde{V}^{n+1}$ will become known only after the next transition. Under Assumption 2.1, $\hat{v}^{n+1}$ is random, so we take an expectation over its conditional distribution, given the choice of action $x$ and given the Bayesian modeling assumption $V \sim \mathcal{N} \left(\bar{V}^n, \Sigma^n\right)$. This expectation is denoted by $\mathbb{E}^n_x$.

We argue that the policy in (8) is a natural extension of the dynamic programming principle. Bellman’s equation is based on the idea of looking ahead to the next state. We simply extend this idea: when we take action $x^n$ out of state $S^n$, we look forward to the next physical state $S^{n+1}$, as well as the next knowledge state $K^{n+1}$. After the transition to $S^{n+1}$ is completed, we approximate $V \left(S^{n+1}\right) \approx \max_y \tilde{Q}^{n+1} \left(S^{n+1}, y \right)$, as we believe that no further improvements to the approximation will be possible. Then, the sum over $S^{n+1}$ in (8) is an expectation over future physical states, and $\mathbb{E}^n_x$ is an expectation over future knowledge states. Unlike (2), equation (8) allows the possibility that the value function approximation will change as a result of the next decision.

The change from $\bar{V}^n$ to $\tilde{V}^{n+1}$ will only occur after we observe the random transition from $S^{x,n} = S^{M,x} \left(S^n, X^{KG,n} \left(S^n, K^n\right)\right)$ to the next pre-decision state $S^{n+1}$, where we compute $\hat{v}^{n+1}$ and update using (6). It can be shown (Powell & Ryzhov, 2012) that, under Assumption 2.1, the conditional distribution of $\tilde{V}^{n+1} \left(S^x\right)$, given $S^n, K^n$ and action $x^n$ at time $n$, can be written as

$$\tilde{V}^{n+1} \left(S^x\right) \sim \bar{V}^n \left(S^x\right) + \frac{\sum_n \left(S^x, S^{x,n}\right)}{\sqrt{\sigma^2 + \sum_n \left(S^{x,n}, S^{x,n}\right)}} Z$$

for $Z \sim \mathcal{N} \left(0, 1\right)$. Note that, in (9), $S^{x,n}$ is the specific post-decision state that we visited at time $n$, while $S^x$ is an arbitrary post-decision state. Furthermore, the random variable $Z$ is not indexed by state, and is common to the conditional distributions of all $V \left(S^x\right)$ for $S^x \in S^x$. This reflects the correlation in our beliefs: we update our beliefs about every state, but the update is driven by a scalar observation. Consequently,

$$\mathbb{E}^n_x \max_y \tilde{Q}^{n+1} \left(S^{n+1}, y \right) = \mathbb{E}^n_x \max_y \left(a^n \left(S^{n+1}, y \right) + b^n \left(S^{n+1}, y \right) Z\right)$$

(10)
with
\[ a^n (S^{n+1}, y) = C(S^{n+1}, y) + \gamma V^n(S^{M,x}(S^{n+1}, y)), \] (11)
\[ b^n (S^{n+1}, y) = \gamma \frac{\sum_n (S^{M,x}(S^{n+1}, y), S^{x,n})}{\sqrt{\sigma^2 + \sum_n (S^{x,n}, S^{x,n})}}. \] (12)

Observe that the vector \( b^n (S^{n+1}, \cdot) \) could have all non-zero values even if \( S^{M,x}(S^{n+1}, y) \neq S^{x,n} \) for all \( y \), due to the correlations between states.

We now apply the computational results of Frazier et al. (2009a) to rewrite (10) as
\[ \mathbb{E}_x \max_y Q^{n+1} (S^{n+1}, y) = \left( \max_y a^n(S^{n+1}, y) \right) + \sum_{y_i \in A(S^{n+1})} [b^n(S^{n+1}, y_{i+1}) - b^n(S^{n+1}, y_i)] f(-|c_i|), \] (13)

where \( A(S^{n+1}) \) is the set of all \( y_i \) for which there exist \( c_{i-1} < c_i \) satisfying \( y_i = \arg \max_{y'} a^n(S^{n+1}, y') + b^n(S^{n+1}, y') z \) for \( z \in (c_{i-1}, c_i) \), with the largest-index rule used to break ties. These \( c_i \) also appear on the right-hand side of (13). The points \( y_i \in A(S^{n+1}) \) are numbered in order of increasing \( b^n(S^{n+1}, y) \), and \( f(z) = z \Phi(z) + \phi(z) \), with \( \Phi, \phi \) being the standard Gaussian cdf and pdf.

The knowledge gradient from \( S^{x,n} \) to \( S^{n+1} \) is defined as the difference
\[ \nu^{KG,n} (S^{x,n}, S^{n+1}) = \mathbb{E}_x \max_y (a^n(S^{n+1}, y) + b^n(S^{n+1}, y) Z) - \max_y a^n(S^{n+1}, y), \]

which can be viewed as the expected improvement achieved in our estimate of \( \max_y Q^n(S^{n+1}, y) = \max_y a^n(S^{n+1}, y) \) as a result of taking action \( x^n \) out of state \( S^n \) with information \( K^n \). We interpret the value of information in terms of this expected improvement. It follows that
\[ \sum_{S^{n+1}} P(S^{n+1} | S^n, x) \mathbb{E}_x \max_y Q^{n+1} (S^{n+1}, y) = \sum_{S^{n+1}} P(S^{n+1} | S^n, x) \max_y Q^n(S^{n+1}, y) + \sum_{S^{n+1}} P(S^{n+1} | S^n, x) \nu^{KG,n} (S^{x,n}, S^{n+1}). \]

We then write, by analogy with (3),
\[ \sum_{S^{n+1}} P(S^{n+1} | S^n, x) \max_y Q^n(S^{n+1}, y) \approx \bar{V}^n(S^{x,n}). \]
because $\hat{V}^n$ is meant to approximate the expected optimal value out of the next pre-decision state, given the current state and action. Thus, (8) becomes

$$X^{KG,n} (S^n, K^n) = \arg \max_x C(S^n, x) + \gamma \hat{V}^n (S^{x,n}) + \gamma \sum_{S^{n+1}} P (S^{n+1} | S^n, x) \nu^{KG,n} (S^{x,n}, S^{n+1}) . \quad (14)$$

The decision $X^{KG,n} (S^n, K^n)$ is similar to the greedy action, except that an additional uncertainty bonus, in the form of a weighted sum of knowledge gradients, is added to each $\hat{Q}^n (S^n, x)$. The balance between exploration and exploitation is evident from (14): the policy rewards actions with higher $\hat{Q}^n (S^n, x)$, but also rewards actions with higher values of $\nu^{KG,n}$. Knowledge gradients tend to be larger when there is more uncertainty (the diagonal entries of $\Sigma^n$ are larger). In the end, the KG policy may choose a decision with reasonably high $\hat{Q}^n (S^n, x)$ and high uncertainty, instead of the greedy action.

For a particular $S^{n+1}$, the KG formula (13) can be computed exactly using a procedure in Frazier et al. (2009a). The cost of the procedure for a single KG factor is $O(M^2 \log M)$, where $M$ is the size of $a^n (S^{n+1}, \cdot)$ and $b^n (S^{n+1}, \cdot)$. Observe that the size of these vectors depends on the number of actions available at $S^{n+1}$, not on the size of the state space. The state space comes into play when we take the weighted sum over $S^{n+1}$ in (14). In most practical problems, the transition probabilities $P (S^{n+1} | S^{x,n})$ are impossible to compute. However, it is often fairly easy to simulate a few transitions from $S^{x,n}$ to $S^{n+1}$. For example, in commodity storage, it is much easier to simulate from a model of the price process than to compute the transition function explicitly. We can approximate

$$\sum_{S^{n+1}} P (S^{n+1} | S^n, x) \nu^{KG,n} (S^{x,n}, S^{n+1}) \approx \frac{1}{K} \sum_{k=1}^K \nu^{KG,n} (S^{x,n}, \hat{S}^{n+1}_k) . \quad (15)$$

for $K$ different pre-decision states $\hat{S}^{n+1}_k$ simulated from the transition probabilities $P (\cdot | S^n, x)$. We used this technique in all of our experiments in Section 5, and obtained good performance for a relatively small sample size such as $K = 30$. Aside from the minor issue of the sample size, the KG algorithm has no new tunable parameters beyond those already in the Bayesian model, such as $\sigma_v^2$.

The KG decision rule in (14) balances the value of information against our current estimates of the rewards obtainable by choosing a particular action. Thus, this policy is most suitable for online learning, where we collect information and improve our approximation in real time, while
we are collecting rewards. Another version of the KG policy, given by

\[ X^{\text{Off,n}}(S^n, K^n) = \arg \max_x \sum_{S^{n+1}} P(S^{n+1} | S^n, x) \nu^{KG,n}(S^{x,n}, S^{n+1}), \]

encourages experimentation by making decisions based purely on the value of information they provide. This version is more suitable for offline learning, where we train our value function approximation inside a simulator.

3 Asymptotic analysis of offline KG

We consider the exploratory behaviour of the KG policy in the limit as the learning budget becomes large. We restrict our analysis to the offline version of the KG policy given in (16). In the online setting, it is known in the optimal learning literature that even optimal policies may converge to suboptimal decisions (Brezzi & Lai, 2000).

The main result of this section is that, by following offline KG, we are guaranteed to explore every pre-decision state (and, in some cases, every post-decision state) infinitely often. By itself, this fact is not equivalent to statistical consistency of the approximation \( \bar{V}^n \), which is generally quite difficult to show for approximate Bayesian models. However, it does provide insight into the behaviour of the algorithm; we see that the policy is driven to explore a large part of the state space and does not get stuck. In showing this result, we do not require Assumption 2.1. Our asymptotic analysis is non-Bayesian in nature, and thus we cannot use, e.g., the martingale analysis of Frazier et al. (2009a) to establish the asymptotic behaviour of the knowledge state \( K^n \). The Bayesian updating equations (6) and (7) can be viewed as a heuristic learning mechanism, which is motivated by Bayesian modeling assumptions, but can still be applied when they do not hold.

**Assumption 3.1.** Under the prior, all post-decision states are correlated. That is, \( \Sigma^0(S^x, S^y) \neq 0 \) for all \( S^x, S^y \in S^x \). Furthermore, the covariance matrix \( \Sigma^0 \) is of full rank.

Our first assumption requires a sufficiently high degree of correlation in our belief structure. Although this assumption may seem strong, it is reasonable in applications where the state variable represents a physical quantity, and there is an intuitive way to judge the degree to which two states are similar. For example, the power-exponential covariance in (5) assumes that all states are correlated. The requirement for \( \Sigma^0 \) to be of full rank allows us to avoid situations where the values
of two states are perfectly correlated. Note that Assumption 3.1 only concerns $\Sigma^0$, which we view as a design parameter of the model, rather than the underlying problem.

**Assumption 3.2.** The sets $\mathcal{S}$ and $\mathcal{X}$ are finite, and post-decision states are defined to be state-action pairs, that is,

$$S^{M,x}(S, x) = (S, x).$$

Furthermore, this Markov decision process is ergodic in the following sense. If we start in an arbitrary state $S$, then for any $S' \in S$, there exists a policy $\pi$ such that we will eventually visit $S'$ with probability (w.p.) 1 by following $\pi$.

Assumption 3.2 is standard in DP theory (Tsitsiklis, 1994; Jaakkola et al., 1994). Most of the work on non-finite state spaces requires stronger assumptions on the problem, or deals with learning the value of a fixed policy (Sutton et al., 2009), or optimizes the policy through approximate policy iteration. Our goal here, however, is to optimize the policy through the computationally efficient mechanism of approximate value iteration. For clarity, we treat the transition probabilities of the Markov decision process as known, though they can be replaced with Monte Carlo estimates as in (15).

**Assumption 3.3.** In every state $S \in \mathcal{S}$, we can choose a special action $\Delta$ for which $C(S, \Delta) = 0$ and $V(S^\Delta) = 0$. That is, $S^\Delta$ is a terminal (absorbing) state.

Essentially, $\Delta$ is an option to retire. In an offline setting, we will never choose this action while training a policy. However, once the training phase is over, and a policy has been implemented, it is reasonable to suppose that we may quit if we reach a hopelessly unfavourable state. Terminal states are also used in stochastic shortest-path models (Neu et al., 2012).

Because state transitions occur randomly, we use $\omega \in \Omega$ to denote a sample realization of all state transitions over an infinite horizon. Because our beliefs are also updated randomly depending on the state transitions, we view the belief parameters $(\bar{V}^n, \Sigma^n)$ as random variables on a probability space $(\Omega, \mathcal{F}, P)$. An event $A$ occurs almost surely (a.s.) if $P(A) = 1$, where $P$ is the probability measure induced by the joint distribution of all state transitions.

Our first result shows the almost sure componentwise convergence of $\Sigma^n$. If Assumption 2.1 were given, this result would follow straightforwardly from a martingale analysis on $K^n$. However,
we emphasize that we do not require Assumption 2.1. In other words, we show that our model, which is motivated by a normality assumption, still possesses certain asymptotic behaviours even when this assumption is removed.

**Proposition 3.1.** Let \( \pi \) be a policy, and let \( E^{\pi,x} \subseteq S^x \) be the set of post-decision states observed infinitely often by following \( \pi \). Under Assumption 3.1, the following holds:

1. For \( S^x \in E^{\pi,x} (\omega) \), we have \( \Sigma^n (S^x, S^y, \omega) \to 0 \) for all \( S^y \in S^x \).
2. If \( S^x, S^y \notin E^{\pi,x} (\omega) \), we have \( \Sigma^n (S^x, S^y, \omega) \to \Sigma^{\infty} (S^x, S^y, \omega) \) where \( \Sigma^{\infty} (S^x, S^y, \omega) \neq 0 \).

**Proof:** From (7), it is clear that \( \Sigma^n (S^x, S^x) \) is decreasing in \( n \), and therefore must have a limit. To show the first statement, fix \( \omega \) and let \( S^x \in E^{\pi,x} (\omega) \). Let \( (n_k) \) be a subsequence, converging to infinity, such that under policy \( \pi \), \( S^{x,n_k} (\omega) = S^x \) for all \( k \). Then, we rewrite (7) as

\[
\Sigma^{n_k+1} (S^x, S^x, \omega) = \left( 1 - \frac{\Sigma^{n_k} (S^x, S^x, \omega)}{\sigma^2 + \Sigma^{n_k} (S^x, S^x, \omega)} \right) \Sigma^{n_k} (S^x, S^x, \omega). \tag{17}
\]

Suppose that \( \lim_{n \to \infty} \Sigma^n (S^x, S^x, \omega) > 0 \). Then, it follows from (17) that

\[
\lim_{k \to \infty} \Sigma^{n_k+1} (S^x, S^x, \omega) < \lim_{n \to \infty} \Sigma^n (S^x, S^x, \omega),
\]

contradicting the uniqueness of limits. Thus, \( \Sigma^n (S^x, S^x, \omega) \to 0 \). By the Cauchy-Schwarz inequality, it follows that \( \Sigma^n (S^x, S^y, \omega) \to 0 \) as well for all \( S^y \in S^x \).

We will now show the second statement. If \( E^{\pi,x} (\omega) \neq S^x \), we can partition \( \Sigma^n (\omega) \) as

\[
\Sigma^n (\omega) = \begin{pmatrix} \Sigma^{E,n} (\omega) & \Sigma^{cross,n} (\omega) \\ \Sigma^{cross,n} (\omega)^T & \Sigma^{E,c,n} (\omega) \end{pmatrix}, \tag{18}
\]

where \( \Sigma^{E,n} (\omega) \) contains variances and covariances for only those states in \( E^{\pi,x} (\omega) \), while \( \Sigma^{E,c,n} (\omega) \) contains covariance information only for states in the complement of \( E^{\pi,x} (\omega) \). We define \( \tau \) to be the last time the policy \( \pi \) visits a state \( S^x \notin E^{\pi,x} (\omega) \) on \( \omega \). Without loss of generality, we can assume \( \tau = -1 \), since we can just take \( \Sigma^0 = \Sigma^{\tau+1} \).

By Assumption 3.1, \( \Sigma^0 \) is invertible. As in (18), we can partition \( (\Sigma^0)^{-1} \) to obtain

\[
(\Sigma^0)^{-1} (\omega) = \begin{pmatrix} \Sigma^{E,0} (\omega) & \Sigma^{cross,0} (\omega) \\ \Sigma^{cross,0} (\omega)^T & \Sigma^{E,c,0} (\omega) \end{pmatrix}.
\]

If \( \tau = 0 \), we can apply the Sherman-Morrison formula to (7) to write

\[
\lim_{n \to \infty} (\Sigma^n)^{-1} (\omega) = (\Sigma^0)^{-1} (\omega) + \sigma^2 \lim_{n \to \infty} n D (\omega),
\]
where $D$ is a diagonal matrix with

$$
D(S^x, S^x, \omega) = \begin{cases} 
1 & S^x \in E^{\pi,x}(\omega) \\
0 & \text{otherwise}.
\end{cases}
$$

From the preceding discussion, we know that $\Sigma^{E,n}(\omega) \to 0$ and $\Sigma^{cross,n}(\omega) \to 0$. We now apply the matrix inversion lemma, and the continuity of the matrix inverse over invertible matrices, to obtain

$$
\lim_{n \to \infty} \Sigma^{E,n}(\omega) = (\Sigma^{E,0})^{-1}(\omega).
$$

We thus conclude that $\Sigma^n(\omega)$ converges componentwise to a limit $\Sigma^\infty(\omega)$.

We argue that $\Sigma^\infty(S^x, S^y, \omega) \neq 0$ for $S^x, S^y \notin E^{\pi,x}(\omega)$. By Assumption 3.1, $\Sigma^0$ has full rank. Let $M = |S^x|$. We can view $\Sigma^0$ as the covariance matrix in a ranking and selection problem (Chau et al., 2014) with $M$ alternatives. In this problem, let $\mu$ denote the vector of true values of these alternatives (analogous to $V$ in the DP), and suppose that we have a multivariate Gaussian prior with covariance matrix $\Sigma^0$ on their values.

Suppose that we can collect unbiased Gaussian observations of the unknown values (as in Assumption 2.1). Suppose, furthermore, that we sequentially collect these observations according to a deterministic policy $\rho_\omega$, which measures the alternatives in the same order in which the policy $\pi$ visits post-decision states in the DP for the sample path $\omega$. The update (7) does not depend on the value of the observation. In fact, if we know which alternative was observed at time $n$, the covariance matrix is updated deterministically. Thus, the sequence of posterior covariance matrices $\Sigma^n$ in the ranking and selection problem will be identical to the sequence observed in the DP for the sample path $\omega$. The limiting behaviour of these two sequences will also be identical.

Now suppose that $S^x, S^y \notin E^{\pi,x}(\omega)$, and consider the two corresponding alternatives in the ranking and selection problem. For notational simplicity, we still label these as $S^x$ and $S^y$. By Assumption 3.1, our prior beliefs about these two alternatives are correlated. We can then express the true values of $S^x$ and $S^y$ as

$$
\mu(S^x) = a_x \cdot C + b_x \cdot Z_x + c \cdot Z_{x,y},
$$

$$
\mu(S^y) = a_y \cdot C + b_y \cdot Z_y + d \cdot Z_{x,y},
$$

where $C$, $Z_x$, $Z_y$ and $Z_{x,y}$ are mutually independent Gaussian random variables, each with strictly positive variance. Suppose that we never collect any observations for the alternatives $S^x$ and $S^y$.\]
analogously to our earlier assumption that $\Sigma^0 = \Sigma^{r+1}$. Then, the conditional variance of $Z_x, Z_y$ and $Z_{x,y}$ remains unchanged, and the resulting correlation between $S^x$ and $S^y$ remains non-zero even in the limit. Since the limiting behaviour of $\Sigma^n$ is identical for this problem and for the DP, we conclude that $\Sigma^\infty (S^x, S^y, \omega) \neq 0$. \qed

We now present a technical result needed to establish limiting behaviour for the KG factor. Essentially this is the same type of result as Proposition 3.1, but for the KG formula.

**Proposition 3.2.** The KG factor $\nu^{KG} (S^x, S)$ is a continuous function of the belief parameters $(\bar{V}, \Sigma)$.

**Proof:** Recall from (13) that

$$
\nu^{KG} (S^x, S) = \sum_{y_i \in A(S)} [b(S, y_{i+1}) - b(S, y_i)] f(-|c_i|),
$$

where $a$ and $b$ are computed using (11) and (12), and the values $c_{y_i}$ are the breakpoints between non-dominated lines of the form $a_i + b_i \cdot z$. These breakpoints have the form

$$
c_i = \frac{a(S, y_i) - a(S, y_{i+1})}{b(S, y_{i+1}) - b(S, y_i)},
$$

which is a rational function of the components of $a$ and $b$. The denominator in (19) is non-zero because the set $A(S)$ has already removed all dominated actions. The function $f$ is continuous. Thus, to show the continuity of $\nu^{KG} (S^x, S)$, it is enough to consider only those parameters $(\bar{V}, \Sigma)$ such that, for either $y = \arg \max_{y_i \in A(S)}$ or $y = \arg \min_{y_i \in A(S)}$, there exists some action $z$ such that $a(S, z) < a(S, y)$ and $b(S, z) = b(S, y)$. This means that there is an action $z$ that is dominated, but that the line corresponding to this action has the same slope as the line for action $y$. A slight change in the line corresponding to action $z$ will add another action to the set $A(S)$.

Let $\varepsilon > 0$ and choose $\delta$ to satisfy

$$
\delta f(0) < \varepsilon.
$$

In addition, $\delta$ should be small enough so that changing the slope and intercept of the line corresponding to action $z$ will only add another action to the beginning or end of the list $A(S)$. Now, let $(\bar{V}', \Sigma')$ be such that $|a(S, z) - a'(S, z)| < \delta$ and $|b(S, z) - b'(S, z)| < \delta$. The only change in
\( \nu^{KG,n} (S^x, S) \) will be a new breakpoint, so that

\[
\left| \nu^{KG,n} \left( S^x, S; \bar{V}, \Sigma \right) - \nu^{KG,n} \left( S^x, S; \bar{V}', \Sigma' \right) \right| = \left| b(S, z) - b(S, y) \right| \cdot f \left( \frac{a(S, y) - a(S, z)}{b(S, z) - b(S, y)} \right) \\
\leq \left| b(S, z) - b(S, y) \right| f(0) \\
< \delta f(0) \\
< \varepsilon.
\]

The second line is due to the fact that \( f \) is increasing. We conclude that the KG factor is continuous in the belief parameters. \( \square \)

We now present our main result showing that the offline KG policy visits every pre-decision state infinitely often w.p. 1. To summarize the argument, we use the preceding results to show that \( \nu^{KG,n} (S^x, S) \to 0 \) when we have visited \( S \) infinitely often, but that the KG factor converges to a strictly positive limit otherwise. We then use a proof by contradiction to show that, if the KG policy visits \( S \) only finitely often, eventually \( S \) becomes preferable to any other pre-decision state.

**Theorem 3.1.** Let \( E^{KG} \) be the set of pre-decision states observed infinitely often by following the offline KG policy. Under Assumptions 3.1, 3.2 and 3.3, we have

\[
P \left( E^{KG} = S \right) = 1.
\]

**Proof:** We assume in these proofs that a suitable set of measure zero has been removed from the outcome space. Suppose that \( E^{KG}(\omega) \neq S \). As in Proposition 3.1, we partition \( \Sigma^n(\omega) \) as

\[
\Sigma^n(\omega) = \begin{pmatrix}
\Sigma^{E,n}(\omega) \\
(\Sigma^{cross,n}(\omega))^T \\
\Sigma^{cross,n}(\omega)
\end{pmatrix}.
\]

By Proposition 3.1, we have

\[
\Sigma^n(\omega) \to \begin{pmatrix}
0 \\
0 \\
\Sigma^{\infty}(\omega)
\end{pmatrix}, \tag{20}
\]

where all components of \( \Sigma^{\infty}(\omega) \) are non-zero.

By Assumption 3.2, there must exist states \( S \in E^{KG}(\omega) \) and \( \bar{S} \not\in E^{KG}(\omega) \) such that, for at least one action \( x \in \mathcal{X} \), we have \( P \left( \bar{S} | S, x \right) > 0 \). Furthermore, by assumption, the offline KG policy must take action \( x \) out of state \( S \) only finitely many times. If this were not the case, it would follow that \( \bar{S} \in E^{KG}(\omega) \). Thus, we have \((S, x) \not\in E^{KG,x}(\omega)\). Furthermore, \((\bar{S}, \bar{x}) \not\in E^{KG,x}(\omega)\) for all \( \bar{x} \), since \( \bar{S} \) is visited only finitely many times.
From (20), we have
\[ \Sigma^\infty (S^M, (\bar{S}, \bar{x}), S^x) \neq 0, \tag{21} \]
\[ \Sigma^\infty (S^x, S^x) \neq 0. \tag{22} \]

By Proposition 3.2, we have
\[ \nu^{KG,n} (S^x, \bar{S}, \omega) \rightarrow \nu^{KG,\infty} (S^x, \bar{S}, \omega), \]
where \( \nu^{KG,\infty} (S^x, \bar{S}, \omega) > 0. \) The fact that the limit is strictly positive is ensured by Assumption 3.3. In (13), one component of the vector \( b^n \) will always be zero for all \( n \), corresponding to the action \( \Delta \). In the limit, \( b^n (\omega) \rightarrow b^\infty (\omega) \) where \( b^\infty (\omega) \) has at least one zero component (due to \( \Delta \)) and at least one non-zero component due to (21). There must therefore be at least one breakpoint.

The function \( f \) in (13) has no zeros on the real line, so we conclude that \( \nu^{KG,\infty} (S^x, \bar{S}, \omega) > 0. \)

Since \( P (\bar{S} | S, x) > 0 \), it follows that
\[ \lim_{n \rightarrow \infty} \sum_{S'} P (S' | S, x) \nu^{KG,n} (S^x, S', \omega) \geq P (\bar{S} | S, x) \nu^{KG,\infty} (S^x, \bar{S}, \omega) > 0. \tag{23} \]

Now, let \( y \) be an action taken infinitely often by the offline KG policy out of state \( S \). Such an action must exist because \( S \) is visited infinitely often. From the preceding discussion, it follows that \( P (S' | S, y) = 0 \) for all \( S' \notin E^{KG} (\omega) \). Furthermore, for any \( S' \in E^{KG} (\omega) \), we have
\[ \Sigma^n (S^M, (S', x'), S^y) \rightarrow 0 \]
due to (20). By Proposition 3.2, it follows that \( \nu^{KG,n} (S^y, S', \omega) \rightarrow 0 \), whence
\[ \sum_{S'} P (S' | S, y) \nu^{KG,n} (S^y, S', \omega) \rightarrow 0. \tag{24} \]

We now put together (23) and (24). Let \( \varepsilon = \nu^{KG,\infty} (S^x, S', \omega) \). Then, there exists an integer \( K_{\omega} \) such that, for all \( n \geq K_{\omega} \), we have
\[ \left| \sum_{S'} P (S' | S, x) \nu^{KG,n} (S^x, S', \omega) - \lim_{n \rightarrow \infty} \sum_{S'} P (S' | S, x) \nu^{KG,n} (S^x, S', \omega) \right| < \frac{\varepsilon}{2}, \]
\[ \sum_{S'} P (S' | S, y) \nu^{KG,n} (S^y, S', \omega) < \frac{\varepsilon}{2}. \]

Consequently, at all times after \( K_{\omega} \), the offline KG policy will prefer action \( x \) to action \( y \) out of state \( S \). This contradicts the assumption that \( \bar{S} \) is visited finitely many times. We conclude that \( E^{KG} (\omega) = S. \)
The next results provide further insight into the behaviour of the offline KG policy. First, we show that all KG factors go to zero, which enables us to examine the policy’s exploration of post-decision states.

**Proposition 3.3.** Under Assumptions 3.1, 3.2 and 3.3,

\[ \nu^{KG,n}(S^x, \tilde{S}) \to 0, \quad \forall S^x \in S^x, \tilde{S} \in S \]

with probability 1.

**Proof:** By Proposition 3.2, every KG factor converges to a limit. Suppose that \( \nu^{KG,n}(S^x, \tilde{S}, \omega) \to \nu^{KG,\infty}(S^x, \tilde{S}, \omega) \) for some \( S^x \in S^x \) and \( \tilde{S} \in S \). It follows that \( S^x \not\in E^{KG,x}(\omega) \), otherwise we would have \( \Sigma^n(S^x, S^x, \omega) \to 0 \), which would imply that \( \nu^{KG,n}(S^x, \tilde{S}, \omega) \to 0 \) by continuity.

Because \( S \) is visited infinitely often by Theorem 3.1, there must be at least one action \( y \) such that \( S^y \in E^{KG,x}(\omega) \). For this action, \( \nu^{KG,n}(S^y, S', \omega) \to 0 \). We can then repeat the argument concluding the proof of Theorem 3.1 to find that, after some time \( K_\omega \), the offline KG policy will prefer action \( x \) to action \( y \), which implies that \( S^x \in E^{KG,x}(\omega) \) and therefore \( \nu^{KG,n}(S^x, \tilde{S}, \omega) \to 0 \).

We conclude that every KG factor must converge to zero under the offline KG policy. \( \square \)

**Proposition 3.4.** Suppose that Assumptions 3.1, 3.2 and 3.3 hold. Let \( S^x \in S^x \) and take \( \tilde{S} \in S \) such that \( P(\tilde{S} \mid S, x) > 0 \). Then, for any action \( \bar{x} \) out of \( \tilde{S} \), the event that at least one of the post-decision states \( S^x, S^{M,x}(\tilde{S}, \bar{x}) \) is an element of \( E^{KG,x} \) occurs w.p. 1.

**Proof:** Suppose that \( S^x, S^{M,x}(\tilde{S}, \bar{x}) \not\in E^{KG,x}(\omega) \). By Proposition 3.1, it must be the case that \( \Sigma^n(S^{M,x}(\tilde{S}, \bar{x}), S^x, \omega), \Sigma^n(S^{M,x}(\tilde{S}, \bar{x}), S^{M,x}(\tilde{S}, \bar{x}), \omega) \), and \( \Sigma^n(S^x, S^x, \omega) \) converge to non-zero limits, as in (21) and (22). Applying Assumption 3.3 as in the proof of Theorem 3.1, we find that \( \nu^{KG,n}(S^x, \tilde{S}) \) converges to a strictly positive limit. This contradicts Proposition 3.3, which states that all KG factors must converge to zero. \( \square \)

From Proposition 3.4, we can conclude the following. For each state-action pair \( (S, x) \), the offline KG policy will either visit \( (S, x) \) infinitely often, or it will visit all state-action pairs downstream of \( (S, x) \) infinitely often, or both. For a particular class of problems, relevant in the study of Markov decision processes where means bound variances (Arlotto et al., 2014), this automatically implies full exploration of every post-decision state.

**Corollary 3.1.** Suppose that Assumptions 3.1, 3.2 and 3.3 hold. Suppose also that there exists
a “do-nothing” action $x_0$ such that $S^{M,W}(S^{M,x}(S,x_0)) = S$ for every $S$. Then, the offline KG policy will visit every state-action pair infinitely often.

The intuition behind Proposition 3.4 is as follows. Suppose, for discussion purposes, that Assumption 2.1 holds, and unbiased observations of $V(S^x)$ are available. Then, for some fixed $S^x$, there are two ways to learn the value $V(S^x)$. We can visit $S^x$ infinitely often and average the observations to obtain the value by the strong law of large numbers. Or, we can visit $S^x$ finitely often, in which case Proposition 3.4 ensures that we will visit all state-action pairs $(\bar{S}, \bar{x})$ with $P(\bar{S} | S, x) > 0$ infinitely often. By the strong law of large numbers, Assumption 2.1 implies that we will learn the values of all post-decision states $S^{M,x}(\bar{S}, \bar{x})$ exactly. However, that gives us the exact value of $V(\bar{S})$, whence (3) also gives us $V(S^x)$. In this way, if we stop visiting a state, we are forced to thoroughly explore all possible downstream states.

4 Extensions to parametric and non-parametric VFAs

In Section 2, we discussed Bayesian exploration for a simplistic value function approximation where values were estimated separately for each post-decision state. In particular, our original Bayesian model required us to store a large matrix containing covariances for each pair of states. Even if we discretize the problem, updating this matrix using (7) quickly becomes computationally intractable as the discretization becomes finer. Furthermore, one rarely has a good sense of the covariances in practice; we would prefer to learn them adaptively using some method for generalizing across states. Fortunately, the KG concept is flexible, and Bayesian learning can easily be integrated with more sophisticated value function approximations (VFAs). In this section, we show how this can be done for two broad classes of approximations: parametric VFAs with basis functions, and non-parametric VFAs with hierarchical aggregation of the state space.

4.1 Basis functions

Suppose that

$$V(S^x) = \sum_{f=1}^{F} \theta_f \phi_f(S^x) = \theta^T \phi(S^x),$$

where the basis functions $\phi_i : S^x \to \mathbb{R}$ are given to us in advance. For example, in the commodity storage problem with $S^x = (R^x, P^x)$, we might define $\phi(S^x) = \left(1, R^x, (R^x)^2, P^x, (P^x)^2, P^x R^x\right)^T$. 

21
Choosing the right basis functions is an art in itself, but once they have been found, the problem of approximating a multi-dimensional and continuous value function reduces to the much simpler problem of estimating a vector \( \theta \) of linear parameters. The intuitive simplicity of basis functions has made them extremely popular as an approximation technique, particularly in reinforcement learning; early treatments can be found in Tesauro (1992) and Sabes (1993), with a more theoretical approach in Tsitsiklis & Van Roy (1997), whereas recent work includes Sutton et al. (2009) and Busoniu et al. (2010).

In this setting, our Bayesian beliefs are placed on the parameter vector \( \theta \), rather than directly on the value function. Initially, we assume \( \theta \sim \mathcal{N}(\theta^0, C^0) \). This automatically induces a prior on the value function, with \( \mathbb{E} V(S^x) = (\theta^0)^T \phi(S^x) \) and \( \text{Cov}(V(S^x), V(S^y)) = \phi(S^x)^T C^0 \phi(S^y) \).

Repeating Assumption 2.1, we obtain the updating equations

\[
\theta^{n+1} = \theta^n + \frac{\hat{V}^{n+1} - (\theta^n)^T \phi(S_{x,n})}{\sigma^2 + \phi(S_{x,n})^T C^n \phi(S_{x,n})} C^n \phi(S_{x,n}), \tag{25}
\]

\[
C^{n+1} = C^n - \frac{C^n \phi(S_{x,n}) \phi(S_{x,n})^T C^n}{\sigma^2 + \phi(S_{x,n})^T C^n \phi(S_{x,n})}, \tag{26}
\]

The Bayesian updating equations (25) and (26) are identical to analogous equations for recursive least squares (Negoescu et al., 2011). They also resemble our original updating equations (6) and (7), but now the covariances are derived from the basis functions. This allows us to avoid having to guess initial covariances: we can choose a diagonal \( C^0 \), whereupon (26) will automatically fill in the off-diagonal entries with empirical covariances. Choi & Van Roy (2006) suggests using \( C^0 = \varepsilon I \) for some small \( \varepsilon \).

The KG algorithm remains virtually unchanged. We only need some additional logic to compute (11) and (12), namely \( \bar{V}^n(S^x) = (\theta^n)^T \phi(S^x) \) and \( \Sigma^n(S^x, S^y) = \phi(S^x)^T C^n \phi(S^y) \). This slightly increases the computational cost of making decisions relative to the original algorithm in Section 2.3, because we now require a small matrix-vector multiplication to compute a posterior covariance. At the same time, we also greatly reduce the computational cost of the updating equations, because the number \( F \) of basis functions is typically very small relative to the size of the state space.

Unfortunately, parametric VFAs may encounter unpredictable convergence issues. These issues affect basis functions in general, regardless of which policy is used to make decisions. Pathological examples of diverging basis functions have been known since Baird (1995). However, many practitioners prefer to use basis functions anyway (see the discussion in Sutton et al., 2008), be-
cause they are easy to use and may produce excellent approximations when they do work well. We showed here how KG can be used with basis functions, but we also offer a discussion of a different, non-parametric class of VFAs.

4.2 Hierarchical aggregation

Hierarchical aggregation of the state space can be useful for large ADP problems (George et al., 2008). We show how this VFA can be combined with the Bayesian ranking and selection model of Mes et al. (2011). We refer to this work for the full derivation of the Bayesian model; here, we show how the computation should be performed in the ADP setting.

An aggregation function $A^g : S^x \rightarrow S^{(g)}$ maps a post-decision state to an “aggregated state” in the space $S^{(g)}$, for $g = 0, ..., G$. The aggregation structure is said to be hierarchical if $|S^{(g)}| \leq |S^{(g-1)}|$ and

$$A^{g-1}(S^x) = A^{g-1}(S^y) \Rightarrow A^g(S^x) = A^g(S^y).$$

If two states belong to the same aggregated state at level $g - 1$, they must also belong to the same aggregated state at level $g$. Our eventual goal in this section is to approximate the value of being in a state as a weighted sum

$$\bar{V}^n(S^x) = \sum_{g=0}^G w^{g,n}(S^x) \bar{V}^{g,n}(S^x),$$

(27)

where $\bar{V}^{g,n}(S^x)$ is our time-$n$ value for the aggregated state $A^g(S^x)$, and $w^{g,n}$ is a weight assigned to level $g$. Below, we describe how this is achieved using Bayesian learning.

We create some additional notation. Let $\mathcal{X}^g(S^x)$ be the set of all $S^y \in S^x$ such that $A^g(S^y) = A^g(S^x)$ for all $S^y \in \mathcal{X}^g(S^x)$. In words, $\mathcal{X}^g(S^x)$ is the set of all states that belong to the same aggregated state at level $g$. Define also $\mathcal{G}(S^x,S^y)$ to be the set of all $g$ with $A^g(S^x) = A^g(S^y)$, that is, all levels of aggregation shared by the two states under consideration.

We retain Assumption 2.1, and also assume that we receive additional observations $\hat{v}^{g,n+1} \sim \mathcal{N}(V^g(S^{x,n}), \lambda^{g,n}_\epsilon(S^{x,n}))$, for all $g = 0, ..., G$, after we make a decision at time $n$. The definition of separate observations for each level of aggregation is an artefact of the hierarchical model; in practice, we simply use $\hat{v}^{g,n+1} = \hat{v}^{n+1}$ for all $g$. The mean $V^g(S^x)$ is interpreted as the unknown value of the aggregated state $A^g(S^x)$. The variance $\lambda^{g,n}_\epsilon(S^{x,n})$ is viewed as a fixed, known parameter by the model, but is automatically inferred from data in our implementation.
We begin with a Bayesian prior \( V(S^x) \sim \mathcal{N}(\bar{V}^0(S^x), \lambda^0(S^x)) \). Our initial assumption is that the values of all the states are independent, but as soon as we start collecting information, we will infer the correlations in the value function through the aggregation structure. To minimize the consequences of this initial independence assumption, we can set \( \lambda^0 \equiv \infty \) to make the prior non-informative, rendering the initial estimate \( \bar{V}^0 \) irrelevant. For the value at level \( g \), we use the model \( V^g(S^x) \sim \mathcal{N}(V^g(S^x), \eta^g(S^x)) \). The variance \( \eta^g(S^x) = \text{Var}(V^g(S^x) - V(S^x)) \) is conditional given our prior distribution.

Given the observations \( \hat{v}^{g,m} \) for \( m \leq n \), the posterior distribution of \( V^g(S^x) \) is Gaussian with mean \( \bar{V}^g_{n}(S^x) \) and variance \( \lambda^g_{n}(S^x) \). The posterior parameters can be obtained (Mes et al., 2011) via the recursive equations

\[
\bar{V}^{g,n+1}(S^x) = \begin{cases} 
\frac{\lambda^g_{n}(S^x)^{-1}\bar{V}^{g,n}(S^x)+\lambda^g_{n}(S^x)^{-1} \hat{v}^{g,n+1}}{\lambda^g_{n}(S^x)^{-1}+\lambda^g_{n}(S^x)^{-1}} & \text{if } A^g(S^x,n) = A^g(S^x) \\
\bar{V}^{g,n}(S^x) & \text{otherwise,}
\end{cases}
\]

and

\[
\lambda^{g,n+1}(S^x) = \begin{cases} 
\left( \lambda^g_{n}(S^x)^{-1} + \lambda^g_{n}(S^x)^{-1} \right)^{-1} & \text{if } A^g(S^x,n) = A^g(S^x) \\
\lambda^g_{n}(S^x) & \text{otherwise.}
\end{cases}
\]

Thus, when we receive an observation of \( V(S^x,n) \), we learn about all states that share at least one aggregated state with \( S^x,n \). Our posterior beliefs about \( V^g(S^x) \) induce a Gaussian posterior on the true value \( V(S^x) \) of that state, with mean \( \bar{V}^n(S^x) \) and variance \( \lambda^n(S^x) \). If our initial prior on \( V \) is non-informative, the mean at time \( n \) is simply given by the weighted sum in (27), with weights

\[
w^{g,n}(S^x) = \frac{(\lambda^g_{n}(S^x) + \eta^g(S^x))^{-1}}{\sum_{g'=0}^G (\lambda^{g',n}(S^x) + \eta^{g'}(S^x))^{-1}}.
\]

It is particularly important to emphasize that the weights are a function of the state. Regions of states that are visited more often will put more weight on the more disaggregate levels, giving us greater precision in our estimates of the values of the most important states. If we have never visited \( A^g(S^x) \), we let \( w^{g,n}(S^x) = 0 \).

In practice, the variance \( \eta^g(S^x) \) is unknown. At time \( n \), we can approximate \( \eta^g(S^x) \approx \delta^{g,n}(S^x)^2 \), where

\[
\delta^{g,n}(S^x) = \begin{cases} 
0 & \text{if } g = 0 \\
|\bar{V}^{g,n}(S^x) - \bar{V}^{0,n}(S^x)| & \text{if } g > 0
\end{cases}
\]
estimates the aggregation bias. Finally, the variance $\lambda_{x}^{g,n} (S^{x,n})$ of the aggregate observations is also unknown, and we estimate it using

$$\lambda_{x}^{g,n} (S^{x}) = \sigma_{x}^{2} + \frac{1}{|A^{g} (S^{x})|} \sum_{S^{y} \in A^{g} (S^{x})} \delta_{x}^{g,n} (S^{y}),$$

a scheme where we put equal weight on the aggregation bias of $S^{y}$ for each $S^{y} \in A^{g} (S^{x})$.

The aggregated model has many parameters, but we can estimate all of them from the knowledge state $K^{n} = \{ \tilde{V}^{g,n}, \lambda^{g,n} \mid g = 0, \ldots, G \}$, updated via (28) and (29). In the end, we only have a single tunable parameter, the variance $\sigma_{x}^{2}$ of the ADP observations. A non-informative prior completely avoids the issue of choosing initial means and variances, which was still present in Section 4.1. Correlations between states are obtained through the aggregation structure, because at time $n$ we update our beliefs about all states $S^{x}$ with $A^{g} (S^{x}) = A^{g} (S^{x,n})$, so we do not have to specify an initial covariance matrix as in Section 2.3.

The KG concept remains the same as in Section 2.3, but the vectors in (11) and (12) are defined in a different way. In the hierarchical model, we have

$$a^{n} (S^{n+1}, y) = C (S^{n+1}, y) + \gamma \tilde{V}^{n} (S^{M,x} (S^{n+1}, y)),
$$
$$b^{n} (S^{n+1}, y) = \gamma \tilde{\sigma}^{n} (S^{M,x} (S^{n+1}, y)),
$$

where, noting the notational difference between $\tilde{V}^{n}$ and $\bar{V}^{n}$, we have

$$\tilde{V}^{n} (S^{y}) = \sum_{g=0}^{G} w^{g,n+1} (S^{y}) \bar{V}^{g,n} (S^{y}) + \sum_{g \in G (S^{y}, S^{x,n})} w^{g,n+1} (S^{y}) \frac{\lambda_{x}^{g,n} (S^{x,n})^{-1}}{\lambda^{g,n} (S^{x,n})^{-1} + \lambda_{x}^{g,n} (S^{x,n})^{-1}} (\bar{V}^{n} (S^{x,n}) - \bar{V}^{g,n} (S^{x,n})) \tag{30}$$

and

$$\tilde{\sigma}^{n} (S^{y}) = \sum_{g \in G (S^{y}, S^{x,n})} w^{g,n+1} (S^{y}) \frac{\lambda_{x}^{g,n} (S^{x,n})^{-1} \sqrt{\lambda^{n} (S^{x,n}) + \sigma_{x}^{2}}}{\lambda^{g,n} (S^{x,n})^{-1} + \lambda_{x}^{g,n} (S^{x,n})^{-1}}. \tag{31}$$

The predictive distribution of $\tilde{V}^{n+1} (S^{y})$ given $S^{n}, K^{n}, x^{n}$ now depends on the old beliefs $\bar{V}^{g,n} (S^{y})$ as well as the bias of our aggregated beliefs about the state $S^{x,n}$ that we chose to visit at time $n$. The weights of these quantities across different levels of aggregation also change from time $n$ to $n+1$. If we knew the variances $\eta^{g} (S^{y})$ for $g = 0, \ldots, G$, we could compute the time-$(n+1)$
weights deterministically given only the time-$n$ beliefs. However, these variances are unknown, and our future approximations $\delta^{g,n+1}(S^y)$ depend on $\bar{V}^{g,n+1}$, still unknown at time $n$. We thus replace $w^{g,n+1}$ in (30) and (31) by the predictive weights

$$w^{g,n}(S^y) = \frac{\left(\left(\lambda^{g,n}(S^y)^{-1} + 1_{g \in G(S^y,S^x,n)})\lambda^{g,n}(S^y)^{-1}\right)^{-1} + \delta^{g,n}(S^y)^2\right)^{-1}}{\sum_{g'=0}^{G}\left(\left(\lambda^{g',n}(S^y)^{-1} + 1_{g' \in G(S^y,S^x,n)})\lambda^{g',n}(S^y)^{-1}\right)^{-1} + \delta^{g',n}(S^y)^2\right)^{-1}},$$

which are equal to $w^{g,n+1}$, except that $\delta^{g,n+1}$ is replaced by $\delta^{g,n}$.

There is a significant computational difference between the original KG algorithm in Section 2.3 and the version presented here. In the hierarchical model, any new information will affect the weights assigned to different levels of aggregation, and the new weights may depend on updated estimates of the values for all states. Thus, even if our new observation happened to be exactly equal to our estimate, we would still change our beliefs, because we would place more or less weight on certain levels. At the same time, updating the knowledge state $K^n$ is extremely fast, depending only on the number $G$ of levels of aggregation, and requiring no matrix operations.

5 Experimental study

Section 5.1 describes the general setup of the experiments and the policies that were implemented. Section 5.2 describes the test problems and presents the results.

5.1 Setup and description of policies

We first discuss the statistical models used by KG and various benchmark policies, as well as the performance metrics used to evaluate these policies. We then discuss the policies themselves. Note that the Bayesian learning model described in Section 2.2 can be used with any exploration policy, not just KG. In fact, all the policies that were implemented in our study use some version of Bayesian learning with correlated beliefs. This allows us to isolate the value of the KG algorithm relative to other exploration strategies. We also found that correlated beliefs were necessary for any policy to learn about the state space within a reasonable amount of time.

Learning models. We used the hierarchical VFA from Section 4.2 for most policies, finding that it consistently performed well and improved computational times. In one of our test problems,
we also implemented the basic discretized belief model from Section 2.2 to better illustrate the benefits of the hierarchical model. The lookup-table model did not scale well to the other examples we considered. Aggregation of the state space was carried out in the same way for any policy that could exploit this approximation structure.

All of our Bayesian learning models require tuning of the noise parameter $\sigma^2$. Note that this parameter belongs to the learning model, not to the exploration policy. Thus, it is implicitly used by every policy, and is completely separate from any additional tuning that a policy might require.

**Policy evaluation.** We evaluated both the offline and online performance of each policy. Recall from Section 2.1 that online performance is evaluated in real time, by adding up the rewards collected by the policy while learning. That is, if $\pi$ is a policy, we calculate

$$C^{\text{online},\pi} = \mathbb{E} \sum_{n=0}^{N} \gamma^n C \left( S^n, X^{\pi,n} (S^n, K^n) \right),$$

where $N$ is the time horizon of interest, and the knowledge state changes in every iteration. By contrast, in the offline setting, we first run the exploration strategy for $N'$ iterations, which yields the final VFA $\bar{V}^{N'}$. We then define a fixed policy, as in (4), in terms of this final VFA, and implement this fixed policy inside our simulator. The offline performance

$$C^{\text{offline},\pi} = \mathbb{E} \sum_{n=0}^{N} \gamma^n C \left( S^n, X^{\text{fixed}} (S^n) \right)$$

measures the quality of the policy that we learned using the exploration strategy. In our simulations, we calculated $C^{\text{offline},\pi}$ for different values of $N'$ (e.g., $N' = 10, 20, ...$) to show how the offline value improves with more time in the simulator.

**Description of learning policies.** Five different types of policies were implemented. Some of these policies, unlike KG, were designed for problems with discrete state spaces, and do not carry over easily to multi-dimensional and continuous problems. These were only implemented in one problem, which was small enough for discretization to be feasible. The descriptions of the policies are as follows.

**Knowledge gradient (KG).** We tested both the online and offline versions of the KG policy, from (14) and (16), with sample size $K = 30$.

**Value of perfect information (VPI).** To our knowledge, the VPI policy by Dearden et al. (1998) was the first exploration strategy to be used together with a Bayesian prior on the value function.
The original definition of VPI is designed for discrete state spaces with normal-gamma priors. However, the policy easily carries over to the VFA structures from Section 4. The decision rule is given by

\[ X^{VPI,n}(S^n, K^n) = \arg \max_x C(S^n, x) + \gamma \bar{V}^n(S^{x,n}) + \gamma \nu^{VPI,n}(S^{x,n}) \]

where

\[ \nu^{VPI,n}(S^{x,n}) = \sqrt{\Sigma^n(S^{x,n}, S^{x,n})} f \left( -\frac{\bar{V}^n(S^{x,n}) - \max_{y \neq x} \bar{V}^n(S^{y,n})}{\sqrt{\Sigma^n(S^{x,n}, S^{x,n})}} \right), \]

with \( f \) remaining the same as in Section 2.3. If we use a VFA from Section 4, we replace \( \Sigma^n(S^{x,n}, S^{x,n}) \) by the corresponding expression for the prior variance of \( V(S^{x,n}) \). For example, in Section 4.1, this is \( \phi(S^{x,n})^T C^n \phi(S^{x,n}) \), and in Section 4.2, this is \( \left( \sum_g \lambda_g^n(S^{x,n}) + \delta_g^n(S^{x,n}) \right)^{-1} \).

VPI can be viewed as a version of the expected improvement policy (Jones et al., 1998; Gramacy & Lee, 2011) from the global optimization literature.

**Epsilon-greedy.** The \( \epsilon \)-greedy policy chooses the action \( \arg \max_x C(S^n, x) + \gamma \bar{V}^n(S^{x,n}) \) with probability \( 1 - \epsilon \), and a random action with probability \( \epsilon \). We tuned the parameter \( \epsilon \) in our experiments. Clearly, this policy can be used with any representation of \( \bar{V}^n \).

**R-max.** The R-max policy of Brafman & Tennenholtz (2003) has attracted considerable attention in the reinforcement learning literature. Essentially, the policy classifies the states based on whether or not we have “enough” knowledge of their values. The decision rule is given by

\[ X^{Rmax,n}(S^n, K^n) = \arg \max_x C(S^n, x) + \gamma F^n(S^{x,n}) \]

where

\[ F^n(S^{x,n}) = \begin{cases} R_{\text{max}} & \text{if } S^{x,n} \text{ has been visited fewer than } m \text{ times}, \\ \bar{V}^n(S^{x,n}) & \text{if } S^{x,n} \text{ has been visited at least } m \text{ times}. \end{cases} \]

The integer \( m \) is a tunable parameter that represents the number of times we need to visit a post-decision state to gain enough knowledge about it. The value \( R_{\text{max}} \) is an arbitrarily large number. Thus, we are encouraged to explore actions with which we are unfamiliar. Because this policy does not easily extend to hierarchical aggregation (the number of times we have visited a state is ambiguous in the hierarchical model), we implemented it with a lookup-table approximation in one of our test problems. We did not implement R-max in problems where the lookup-table approximation did not scale.
The $E^3$ policy of Kearns & Singh (2002) is somewhat similar to R-max. If we visit a state that we have never visited before, we choose a random action. If we have visited the state at least once, but fewer than $m$ times, we choose the action that has been tried the fewest number of times out of all the times we have previously visited the state. Lastly, if we have visited the state more than $m$ times, we take the greedy action $\arg\max_x C(S^n, x) + \gamma \hat{V}^n(S^{x,n})$. We implemented $E^3$ in our first test problem, but not in subsequent problems, for the same reasons as R-max.

5.2 Experimental results

We considered three test problems based on different applications of resource allocation. In the following, we omit the details of the models (e.g., the precise transition functions used), as every competing policy was implemented on the same model, and our focus is on comparing policies. We briefly discuss the size and structure of each problem and present the results.

**Commodity storage with stochastic price.** We consider a simple storage problem with a two-dimensional state space $S = (R, P)$, where $R$ is the resource level and $P$ is the price. The price evolves according to a geometric Ornstein-Uhlenbeck process, and is subject to a sufficiently high level of noise. The decision $x$ is a scalar quantity representing how much to buy or sell, appropriately constrained by how much of the commodity we currently have in storage. The single-period reward represents the revenue earned from selling, or the cost incurred by buying; the objective is to maximize long-term discounted revenue.

For the hierarchical VFA of Section 4.2, each level of aggregation partitioned the state space into rectangles. At the finest levels of aggregation, we added more “bins” to the price variable, while the coarser levels focused primarily on the resource variable. As the discretization became finer, it was more important to distinguish between similar prices than similar storage quantities. This VFA was used by the KG, $\varepsilon$-greedy, and VPI policies. The R-max and $E^3$ policies were implemented with a basic lookup-table VFA. For comparison, we also implemented the KG policy with the lookup table.

Figure 1(a) shows online performance, expressed in terms of the revenue collected in real time by each learning policy. Figure 1(b) shows offline performance, expressed in terms of the infinite-horizon value of the VFA trained by running each learning policy. Performance values are averaged over sufficiently many simulations to render the standard errors negligible. The key results are as
The hierarchal VFA of Section 4.2 adds value over the basic lookup-table model in Section 2.2. We implemented KG and $\varepsilon$-greedy with both types of models, and found that the hierarchal VFA improved the performance of both policies.

With the hierarchal VFA, the KG policy, described by (14) or (16) depending on the type of problem, achieved the best overall performance. The targeted exploration provided by the KG logic clearly outperformed policies that chose actions at random, including R-max and $E^3$.

The $E^3$ policy worked well offline, though KG obtained similar results in fewer iterations. The R-max policy conducted less exploration (it essentially makes a greedy choice between seldom-visited states), and underperformed the other policies both online and offline.

Among all the policies, KG required the most computational effort to make a single decision. The KG logic required approximately 3.1 milliseconds (ms) per iteration with the hierarchal VFA and 0.7 ms with the lookup VFA. Simpler policies such as $\varepsilon$-greedy, R-max and $E^3$ required on the order of $10^{-3}$ ms to make a decision. At the same time, the computational effort required to update the beliefs was on the order of $10^{-3}$ ms per iteration for the hierarchal VFA, and approximately 1.3 ms for the lookup table. Furthermore, the time and storage space required to update the lookup table grows quadratically in the size of the discretization (since we have to store a covariance matrix), creating a bottleneck for larger problems. Thus, the fastest policies overall were $\varepsilon$-greedy and VPI.
which can be combined with the hierarchical VFA.

**Storage with stochastic supply.** Similar to the previous problem, this version also has a two-dimensional state variable \( S = (R, W) \), where \( W \) is the (mean-reverting) stochastic supply process. The demand and spot price of the commodity were assumed to be fixed constants in every time period, and the objective is to minimize the long-term cost of meeting demand. However, the decision space now has five dimensions (up from one dimension in the previous example), because the commodity can be moved 1) from supply to demand; 2) from supply to storage; 3) from the spot market to storage; 4) from the spot market to demand; 5) from storage to demand. Three of these dimensions are controllable by the decision-maker. Because of the higher dimensionality, the lookup-table VFA does not scale to this problem, and the R-max and \( E^3 \) policies are omitted from the results.

Figures 2(a) and 2(b) show online and offline performance, respectively. The results are largely consistent with Figure 1. KG retains a considerable lead over the other policies in the online setting. Offline, it performs competitively with \( \varepsilon \)-greedy, but takes fewer iterations to find a good solution.

**Nomadic trucker.** In the nomadic trucker problem (Powell, 2011), a single truck observes demands that arise randomly in different locations. The state \( S = (S^d, S^l, S^k) \) has three dimensions representing the current location of the trucker, the current day of the week, and the trailer type. The set of possible decisions depends on the current state: for example, we can choose to accept a currently available load of a particular type in some location, or we can choose to move empty. Such decisions can conveniently be modeled as binary vectors whose components sum up to 1, and
whose $i$th component equals 1 if the $i$th possible decision is chosen. Moving empty incurs a cost, while transporting a load gives a reward (larger trailer types result in higher rewards). The arrival intensity of loads is randomized to ensure sufficient variability between locations.

Table 1 gives an overview of the aggregation structure used by the VFA, with ‘*’ corresponding to a dimension that was included in the aggregation level, and ‘-’ corresponding to a dimension that was aggregated out. Trailer type and day-of-week are either included or left out, while location is represented with an increasingly fine grid at the more disaggregate levels.

Figure 3 shows the online and offline performance of the policies (hierarchical KG, VPI, and $\varepsilon$-greedy). The $\varepsilon$-greedy policy is implemented both with the hierarchical VFA and with a simple lookup-table approximation. We see that hierarchical KG achieves the best performance in both cases, although both VPI and hierarchical $\varepsilon$-greedy are competitive in the offline setting. Note that, in Figure 3(b), $\varepsilon$-greedy exhibits a decline in performance after the first 50 iterations, which occurs because the policy tends to get stuck visiting locations that it has already seen, leading to a large number of empty moves.

<table>
<thead>
<tr>
<th>Level</th>
<th>Location</th>
<th>Trailer type</th>
<th>Day-of-week</th>
<th>Size of state space</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16 × 16</td>
<td>*</td>
<td>*</td>
<td>$256 \times 3 \times 7 = 5376$</td>
</tr>
<tr>
<td>1</td>
<td>8 × 8</td>
<td>*</td>
<td>*</td>
<td>$64 \cdot 3 \cdot 7 = 1344$</td>
</tr>
<tr>
<td>2</td>
<td>4 × 4</td>
<td>*</td>
<td>*</td>
<td>$16 \cdot 3 \cdot 7 = 336$</td>
</tr>
<tr>
<td>3</td>
<td>4 × 4</td>
<td>-</td>
<td>*</td>
<td>$16 \cdot 7 = 112$</td>
</tr>
<tr>
<td>4</td>
<td>2 × 2</td>
<td>-</td>
<td>*</td>
<td>$4 \cdot 7 = 28$</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>*</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1: Aggregation structure for the nomadic trucker problem.
We briefly discuss some implementation issues that arose in this problem. First, every policy that used the hierarchical VFA required tuning of $\sigma^2\varepsilon$, which has a large impact on performance. For hierarchical KG and VPI, this quantity has an impact on the amount of exploration, since it governs the decline in the posterior variance of the beliefs. The hierarchical $\varepsilon$-greedy policy requires tuning of both $\sigma^2\varepsilon$ and $\varepsilon$, and can perform competitively when these values are finely tuned, though this process is more time-consuming than for KG. We found that KG worked the best with large initial values of the VFA (also recommended in Sec. 4.9.1 of Powell, 2011 for ADP in general) and moderate values of $\sigma^2\varepsilon$, to balance the need for exploration with the decreasing value of information in the later stages.

In the offline setting, computation times were higher for KG than for any other policy. However, in the online setting, the additional computational effort may be beneficial for avoiding unnecessary costs with each time step. Because we make decisions in real time in the online setting, the time between decisions may be hours, a day or even longer, justifying the extra effort involved in using the KG policy.

6 Conclusion

We have proposed a new exploration strategy for approximate dynamic programming. Our approach uses a Bayesian belief model to quantify our uncertainty about the value function, and applies optimal learning concepts to calculate the value of new observations in improving our approximation and reducing our uncertainty. The value of information is traded off against our current beliefs about the value function using a generalized form of Bellman’s equation. We first prove that this approach conducts sufficient exploration under a simple lookup-table belief model. The real advantage of the approach, however, is that it can easily be incorporated into two powerful classes of value function approximations. In our experiments, the new method outperformed the other benchmarks on all test problems after an initial number of replications. This performance comes at the cost of increased computation time, but this is less important in the online setting, leading to much greater practical benefits.

By using more sophisticated VFAs, our approach can scale to multi-dimensional, continuous state variables, but it generally requires discrete decision variables. It may be possible to handle low-dimensional, continuous decision variables by adapting the procedure in Scott et al. (2011).
For parametric VFAs, the approach of Han et al. (2013) may help to adaptively learn $\sigma^2$, reducing the burden of tuning this parameter. In problems where both the state and decision spaces are large, our ability to use ADP effectively has more to do with the ability of the VFA to exploit the underlying problem structure. However, our results suggest that Bayesian exploration can help to obtain good results more quickly when it is possible to integrate Bayesian learning into the VFA.

**Acknowledgments**

This research was supported in part by AFOSR contract FA9550-08-1-0195.

**References**


