Bayesian Exploration for Approximate Dynamic Programming

Ilya O. Ryzhov
Robert H. Smith School of Business & Institute for Systems Research, University of Maryland, College Park, MD 20742, iryzhov@rhsmith.umd.edu, http://scholar.rhsmith.umd.edu/iryzhov

Martijn R.K. Mes
Industrial Engineering and Business Information Systems, University of Twente, PO Box 217, 7500 AE Enschede, The Netherlands, m.r.k.mes@utwente.nl

Warren B. Powell, Gerald van den Berg

Approximate dynamic programming (ADP) is a general methodological framework for multi-stage stochastic optimization problems in transportation, finance, energy, and other domains. We propose a new approach to the exploration/exploitation dilemma in ADP that leverages two important concepts from the optimal learning literature: first, we show how a Bayesian belief structure can be used to express uncertainty about the value function in ADP; second, we develop a new exploration strategy based on the concept of value of information, and prove that it systematically explores the state space. An important advantage of our framework is that it can be integrated into both parametric and non-parametric value function approximations, which are widely used in practical implementations of ADP. We evaluate this strategy on a variety of distinct resource allocation problems and demonstrate that, while more computationally intensive, it is highly competitive against other exploration strategies.

Key words: approximate dynamic programming; optimal learning; Bayesian learning; correlated beliefs; value of information

History: Received July 2015; revisions received March 2017, December 2017, April 2018; accepted May 2018.
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 and Barto 1998) or neuro-dynamic programming (Bertsekas and 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 and Powell 2006), asset acquisition (Nascimento and Powell 2009), and sensor management (Cas-tanon 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 and 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 requires cash on hand to meet shareholder redemptions (Nascimento and Powell 2010). Holding too much cash leads to lost investment opportunities, while a deficit of cash leads to high shortfall costs from liquidating part of the portfolio.

4. **Service operations.** A call center assigns different classes of agents to jobs with stochastic arrival and service times (Koole and 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 more 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 ADP literature contains many general exploration heuristics, ranging from simple methods such as Boltzmann exploration and epsilon-greedy (Sutton and Barto 1998) to more sophisticated techniques, such as $E^3$ (Kearns and Singh 2002) and R-max (Brafman and Tennenholtz 2003). These methods can perform well on small, discrete-state problems, but may require tuning or run into scaling issues in higher dimensions (both $E^3$ and R-max essentially enumerate the state space).

In this paper, we present a new framework for exploration in ADP that leverages two concepts from the literature on optimal learning (Powell and Ryzhov 2012), which considers exploration/exploitation in simple, streamlined optimization models. In these problems, the state variable is a state of knowledge (a Bayesian distribution of belief) about a set of unknown values. This knowledge state is updated by conducting experiments, either within a simulator (the ranking and selection problem; Kim and Nelson 2006, Hong and Nelson 2009), or in real time with an immediate reward (the multi-armed bandit problem; Gittins et al. 2011). By adopting this Bayesian approach
in ADP, using the observed state transitions as the “experiments,” we can express our uncertainty about the unknown value function. While Bayesian thinking has been previously applied in reinforcement learning (Dearden et al. 1998), our model makes full use of correlated beliefs (see Ryzhov and Chen 2017 for an introduction) to quantify similarities and differences between various decisions; as a result, a single decision can now provide information about a much larger portion of the state space, greatly increasing our ability to explore. Second, the concept of value of information (Chick 2006), also known as knowledge gradient or KG (Frazier et al. 2009), may be brought into ADP to evaluate the potential of different decisions to provide useful information. The result is a computationally tractable algorithm that adaptively identifies states that are likely to yield high rewards themselves or to lead us to such high-reward states later on.

The optimal learning literature has developed numerous sophisticated algorithms for information collection, including Gittins indices (Gittins et al. 2011), upper confidence bound methods (Auer et al. 2002), Thompson sampling (Russo and Van Roy 2014), and optimal computing budget allocation (He et al. 2007). However, despite the depth of this literature, extending these ideas to ADP is far from straightforward, due to the added complication of a physical state (or, more generally, a controllable state). In ADP, a decision not only provides rewards and information, it 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 and Barto 1996, Szepesvári 2010), bandit theory does not engage with the physical state directly unless the latter exhibits some special structure (such as the network structure in Brown and Smith 2013). Unlike the aforementioned methods, the knowledge gradient approach adopted in this paper can be adapted to handle both physical and knowledge states in a principled way.

Moreover, practical implementations of ADP almost always require the ability to generalize to a variety of algorithmic representations of the state space, whereas most efforts to tackle exploration rely on strong assumptions such as finite state spaces (Kearns and Singh 2002), deterministic transitions (Wen and Van Roy 2013), or fixed correlations (Pazis and Parr 2013). Our framework not
only handles physical states in exploration, it can also be efficiently integrated into both parametric (Sutton et al. 2009) and non-parametric (George et al. 2008) approximation architectures, which scale much better to larger problems than the traditional lookup-table representation. The optimal learning literature has observed that Bayesian beliefs can be rigorously modeled in these settings (Negoescu et al. 2011, Mes et al. 2011); the implication for our ADP setting is that our approach gains from switching to one of these representations, because correlations in the Bayesian structure can now be stored much more efficiently or calculated on demand. Our paradigm is also directly applicable to both online (learning in real time) and offline (learning inside a simulator) implementations of ADP. This flexibility is perhaps the most useful aspect of our proposed exploration strategy: although it incurs additional computational cost, it leads to improved performance on a variety of problems that are too large to solve using lookup tables.

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. In Section 3, we show how these concepts can be used together with more powerful value function approximations (VFAs) that can handle continuous, multi-dimensional state spaces. Section 4 analyzes the asymptotic behaviour of the policy in the discrete setting. Section 5 demonstrates the performance of our method on several example problems. Section 6 concludes. The online Appendix includes full proofs, additional technical discussions and extensions, and details of the computational study.

2. Main concepts: models and algorithms

To make the main concepts of our paper as clear as possible, we first present them in a classic Markov decision process (Puterman 1994) setting, with discrete state and action spaces (Section 3 will explain how the same ideas can be applied in multi-dimensional and/or continuous problems). Section 2.1 presents an overview of ADP (largely following ch. 4 of Powell 2011) and describes the basic structure of approximate value iteration, a fundamental algorithmic strategy for learning the value function. Section 2.2 introduces Bayesian statistical learning into this setting. Section 2.3 then shows how Bayesian learning can be integrated into the approximate value iteration algorithm using the knowledge gradient concept.
2.1. Overview of ADP

Consider a Markov decision process with state space $S$ and decision space $X$. Let $C$ be a function mapping a state $S \in S$ and an action $x \in X$ to a deterministic reward $C(S, x)$. Our objective is to maximize the total discounted expected infinite-horizon reward

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

for all $S \in S$, where $\gamma \in (0, 1)$ is a discount factor and $(S^n)$ is the sequence of visited states. The notation $\mathbb{E}_\pi$ represents an expectation over all random transitions (from $S^n$ to $S^{n+1}$ for all $n$) under some policy $\pi$, which is characterized by the decision rule $X^{\pi,n}$ mapping states to actions. The optimal value function $V$ is the unique solution to Bellman’s equation,

$$V(S) = \max_{x \in X} C(S, x) + \gamma \mathbb{E} (V(S') \mid S, x) \quad \text{for all } S \in S,$$

and the optimal policy makes the decision that maximizes the right-hand side of (1).

In most practical problems, we cannot solve (1) exactly due to the curses of dimensionality (Powell 2011). A popular strategy in ADP is to step forward in time and compute an approximate observation

$$\hat{v}^n = \max_{x \in X} C(S^n, x) + \gamma \hat{V}^{n-1} \left( S^{M,x}(S^n, x) \right),$$

where $S^n$ is the state of the dynamic program in the $n$th time step. The function $\hat{V}^{n-1}$ is an approximation of $V$, 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 becomes $S^{n+1} = S^{M,W}(S^{x,n}, W^{n+1})$. Figure 1 illustrates the distinction between these two types of states; the transition from pre- to post-decision state is purely deterministic and depends only on the chosen action, while the transition from post- to pre-decision state is purely stochastic and cannot be controlled by the decision-maker.

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

$$V(S^{x,n}) \equiv \mathbb{E} \left( V(S^{n+1}) \mid S^n, x^n \right),$$

(3)
Figure 1  Diagram illustrating pre-decision states (circles) and post-decision states (squares). Solid lines indicate deterministic transitions, while dashed lines represent stochastic transitions.

and use $\hat{v}^{n-1}$ 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 and Yin 2003) to update $\hat{V}^{n-1}(S^{x,n-1})$ with $\hat{v}^n$, creating a new approximation $\tilde{V}^n$. In classical methods such as Q-learning (Watkins and Dayan 1992), each post-decision state is updated independently and $\hat{v}^n$ is only used to update $\hat{V}^{n-1}(S^{x,n-1})$. 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 $x^n$ (by contrast, in on-policy learning, $x^n$ is always the action that maximizes (2)). We always use (2) to calculate the approximate observation $\hat{v}^n$ and update $\hat{V}^{n-1}$, but the action $x^n$ that we choose is not required to be the maximizer of (2), and can be calculated using some other algorithm. In this paper, the terms “off-policy learning” and “exploration” are interchangeable, since we will choose $x^n$ to explore the state space.
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 VFA is trained in real time; we are actually solving the problem, making decisions, and collecting rewards while $\bar{V}^{n-1}$ is being updated. The online performance of a policy $\pi$ can be written as

$$C_{\text{online},\pi} = \mathbb{E}_\pi \sum_{n=0}^{\infty} \gamma^n C(S^n, X^{\pi,n}(S^n)).$$ (4)

By contrast, offline evaluation means that we have access to a simulation model of the underlying system, and train the VFA inside the simulator under some policy $\pi$. Then, after running $\pi$ for a sufficient amount of time (say, $N$ iterations) and obtaining the final VFA $\bar{V}^N$, we define a fixed policy $\rho(\pi)$ via the decision rule

$$X^{\rho(\pi)}(S) = \arg \max_x C(S, x) + \gamma \bar{V}^N(S, M, x(\pi), x).$$ (5)

and implement this fixed policy inside our simulator. The notation $\rho(\pi)$ indicates that the fixed policy $\rho$ is induced by a VFA that was derived from the policy $\pi$. Then, the offline performance of $\pi$ is calculated as

$$C_{\text{offline},\pi} = \mathbb{E}_{\rho(\pi)} \sum_{n=0}^{\infty} \gamma^n C(S^n, X^{\rho(\pi)}(S^n)).$$ (6)

Offline evaluation allows for more experimentation: the policy $\pi$ is allowed to make poor decisions while it is running, as long as we can obtain a good VFA in $N$ iterations. On the other hand, online evaluation requires us to make decisions more conservatively, because a poor decision can severely impact immediate and future rewards.

In the past, the ADP literature has generally not made 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 $x^n$ affects the next state $S^{n+1}$, but it also affects the next observation $\hat{v}^{n+1}$, and through that, the future approximation $\tilde{V}^{n+1}$. Thus, if we have some probabilistic model or forecast of the value of $\hat{v}^{n+1}$, we could explicitly take this feedback into account when making decision $x^n$. In this section we describe such a methodology using a Bayesian belief structure.

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 $\mathbb{E}V(S^x) = \bar{V}^0(S^x)$ and $\text{Cov}(V(S^x), V(S^y)) = \Sigma^0(S^x, S^y)$ 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

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

(7)

where $\| \cdot \|_2$ is the Euclidean norm, $\beta > 0$ is an initial variance, and $\alpha > 0$ represents the spread of correlation over distance. This is a heuristic covariance structure (the parameters $\alpha$ and $\beta$ may require tuning), but 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. We introduce a modeling assumption that will simplify the execution of our exploration algorithm.

**Assumption 1.** The ADP observation $\hat{v}^{n+1}$ has the distribution $\mathcal{N}(V(S^{x,n}), \sigma^2)$, with $\sigma^2 > 0$ known, 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 theorem 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 $\tilde{V}^n$, 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 (see George and Powell 2006 for a discussion of the problem of estimating this bias). In practice, one may reduce this bias by conducting a set of prior runs to obtain prior estimates $\bar{V}^0$. Finally, a further limitation of Assumption 1 is that $\sigma_\varepsilon$ is assumed to be known; however, it is possible to introduce additional logic into the Bayesian model that allows the variance parameter to be tuned automatically as new information is acquired. To simplify the presentation, we treat $\sigma_\varepsilon$ as known in the main body of this paper, and treat the extension to unknown variance in the online Appendix (Section EC.2).

Assumption 1 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 1, we can apply standard Bayesian updating equations (Powell and Ryzhov 2012) to obtain a new set of beliefs

$$\bar{V}^{n+1}(S_x) = \bar{V}^n(S_x) + \frac{\hat{v}^{n+1} - \bar{V}^n(S_x^n)}{\sigma_\varepsilon^2 + \Sigma^n(S_x^n, S_x^n)} \Sigma^n(S_x, S_x, S_x^n),$$

(8)

$$\Sigma^{n+1}(S_x, S_y) = \Sigma^n(S_x, S_y) - \frac{\Sigma^n(S_x^n, S_x^n) \Sigma^n(S_x^n, S_y)}{\sigma_\varepsilon^2 + \Sigma^n(S_x^n, S_x^n)},$$

(9)

for each post-decision state $S_x^n \in S_x$. This update may be viewed as a form of Gaussian process (GP) regression (Rasmussen and Williams 2006) on the finite set $S^x$, using the biased observation $\hat{v}^{n+1}$ for lack of a better and more tractable estimate of the value function (see Deisenroth et al. 2009 for an approach to constructing more accurate observations, at the cost of much greater computation times).

The posterior parameters $\bar{V}^n$ 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 term “hyperstate” (dating back to Bellman and Kalaba 1959) is sometimes used in the literature to describe $(S^n, K^n)$.

So far, the learning model we have laid out requires us to store values $\tilde{V}^n(S^x)$ 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. In many practical settings, lookup tables are computationally prohibitive, but for now we use them to illustrate the main ideas of the paper.

### 2.3. The knowledge gradient policy for exploration

We now use the simple lookup-table learning model from Section 2.2 to show how Bayesian beliefs can be used to promote exploration in ADP. Let $P(S'|S, x)$ denote the probability of transitioning to state $S'$ after taking action $x$ out of state $S$, and let

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

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

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

Now suppose that we are in 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 (8)-(9). Thus, the action $x^n$ can depend on the most recent value function approximation $\tilde{V}^n$. 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 $\tilde{V}^{n'} = \tilde{V}^{n+1}$ 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 $\tilde{V}^{n+1}(S^{M,x}(S^{n+1}, y))$.

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

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

(10)
On the right-hand side of (10), we assume that, once we land in the downstream state \( S^{n+1} \), our future action \( y \) will be calculated using \( \bar{Q}^{n+1}(S^{n+1},\cdot) \), which depends on both the future VFA \( \bar{V}^{n+1} \) and on the downstream state itself. The future VFA \( \bar{V}^{n+1} \) will only become known after the transition to \( S^{n+1} \), and the choice of action \( x \) in (10) is made before this transition happens. Assumption 1 treats \( \hat{v}^{n+1} \) as a random variable and induces a conditional distribution on the future VFA \( \bar{V}^{n+1}(\cdot) \) (known as a “predictive distribution” in Bayesian statistics). The expected value \( \mathbb{E}_x^n \) in (10) is taken over this predictive distribution.

We view (10) as a natural extension of the dynamic programming principle. Bellman’s equation calculates the optimal action at time \( n \) by looking ahead to the next state \( S^{n+1} \). Our proposed approach likewise looks forward to the next physical state \( S^{n+1} \), but also to the next knowledge state \( K^{n+1} \). The sum in (10) is an expectation over future physical states, while \( \mathbb{E}_x^n \) is an expectation over future knowledge states. Unlike (2), equation (10) acknowledges that the VFA will change as a result of the next decision, rather than simply using the point estimate \( \bar{V}^n \) of the downstream value.

It can be shown (Powell and Ryzhov 2012) that, under Assumption 1, the Bayesian predictive distribution of \( \bar{V}^{n+1}(\cdot) \), given \( S^n, K^n \) and action \( x^n \) at time \( n \), can be written as

\[
\bar{V}^{n+1}(\cdot) \sim \bar{V}^n(\cdot) + \frac{\Sigma^n(S_x^n, S_{x,n}^n)}{\sqrt{\sigma_x^2 + \Sigma^n(S_{x,n}^n, S_{x,n}^n)}} Z
\]  

for \( Z \sim \mathcal{N}(0,1) \). In (11), \( S_{x,n}^n \) is the specific post-decision state that we visited at time \( n \), while \( S_x^n \) 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 \( V(S_x^n) \) for \( S_x^n \in S_x^n \). This reflects the correlation in our beliefs: we update our beliefs about every state, but the update is driven by a scalar observation. Consequently,

\[
\mathbb{E}_x^n \max_y \bar{Q}^{n+1}(\cdot, y) = \mathbb{E}_x^n \max_y (a^n(\cdot, y) + b^n(\cdot, y) Z)
\]

with

\[
a^n(\cdot, y) = C(\cdot, y) + \gamma \bar{V}^n(S_{M,x}(\cdot, y))
\]

\[
b^n(\cdot, y) = \gamma \frac{\Sigma^n(S_{M,x}(\cdot, y), S_{x,n}^n)}{\sqrt{\sigma_x^2 + \Sigma^n(S_{x,n}^n, S_{x,n}^n)}}.
\]
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 derivations from Frazier et al. (2009) to rewrite (12) as

\[
\mathbb{E}_n^{\max_y} \bar{Q}^{n+1}(S^{n+1}, y) = \left( \max_y a^n(S^{n+1}, y) \right) + \sum_{y_i \in \mathcal{A}(S^{n+1})} \left[ b^n(S^{n+1}, y_i + 1) - b^n(S^{n+1}, y_i) \right] f \left( -|c_i| \right).
\]

(15)

The function \( f \) is defined as

\[
f(z) = z\Phi(z) + \varphi(z),
\]

with \( \Phi \) and \( \varphi \) being the standard Gaussian cdf and pdf. The values \( c_i \) are the breakpoints of the piecewise linear function

\[
z \mapsto \max_y \left( a^n(S^{n+1}, y) + b^n(S^{n+1}, y) \cdot z \right)
\]

(16)

arranged in ascending order, i.e., \( c_{i-1} < c_i \). More formally, \( c_i \) can be characterized as follows. In (15), \( \mathcal{A}(S^{n+1}) \) denotes the set of all \( y_i \) that achieve the argmax in (16) for \( z \in (c_{i-1}, c_i) \). In other words, for each \( y_i \in \mathcal{A}(S^{n+1}) \), there exist \( c_{i-1} < c_i \) satisfying \( y_i = \arg\max_{y'} a^n(S^{n+1}, y') + b^n(S^{n+1}, y') \cdot z \) for \( z \in (c_{i-1}, c_i) \), with the largest-index rule used to break ties.

The knowledge gradient from \( S^{x,n} \) to \( S^{n+1} \) is defined as the difference

\[
\nu^{KG,n}(S^{x,n}, S^{n+1}) = \mathbb{E}_n^{\max_y} \left( a^n(S^{n+1}, y) + b^n(S^{n+1}, y) \cdot Z \right) - \max_y a^n(S^{n+1}, y),
\]

which can be viewed as the expected improvement achieved in our estimate of \( \max_y \bar{Q}^n(S^{n+1}, y) = \max_y a^n(S^{n+1}, y) \) 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

\[
\sum_{S^{n+1}} P(S^{n+1} | S^n, x) \mathbb{E}_n^{\max_y} \bar{Q}^{n+1}(S^{n+1}, y) = \sum_{S^{n+1}} P(S^{n+1} | S^n, x) \max_y \bar{Q}^n(S^{n+1}, y) + \sum_{S^{n+1}} P(S^{n+1} | S^n, x) \nu^{KG,n}(S^{x,n}, S^{n+1}).
\]

We then write, by analogy with (3),

\[
\sum_{S^{n+1}} P(S^{n+1} | S^n, x) \max_y \bar{Q}^n(S^{n+1}, y) \approx \bar{V}^n(S^{x,n})
\]
because $\bar{V}^n$ is meant to approximate the expected optimal value out of the next pre-decision state, given the current state and action. Thus, (10) becomes

$$X^{KG,n}(S^n, K^n) = \arg \max_x C(S^n, x) + \gamma \bar{V}^n(S^{x,n})$$

$$+ \gamma \sum_{S^{n+1}} P(S^{n+1} | S^n, x) \nu^{KG,n}(S^{x,n}, S^{n+1}). \quad (17)$$

The decision $X^{KG,n}(S^n, K^n)$ is similar to the greedy action $\arg \max_x C(S^n, x) + \gamma \bar{V}^n(S^{x,n})$, except that an additional uncertainty bonus, in the form of a weighted sum of knowledge gradients, is added to each $\bar{Q}^n(S^n, x)$. The balance between exploration and exploitation is evident from (17): the policy rewards actions with higher $\bar{Q}^n(S^n, x)$, 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). The KG policy may now choose a decision with high uncertainty, rather than taking the greedy action.

For a particular $S^{n+1}$, the KG formula (15) can be computed exactly using a procedure from Frazier et al. (2009). 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 $b^n(S^{n+1}, \cdot)$. 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 (17). 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

$$\sum_{S^{n+1}} P(S^{n+1} | S^n, x) \nu^{KG,n}(S^{x,n}, S^{n+1}) \approx \frac{1}{K} \sum_{k=1}^K \nu^{KG,n}(S^{x,n}, \hat{S}^{n+1}_k). \quad (18)$$

for $K$ different pre-decision states $\hat{S}^{n+1}_k$ simulated from the transition probabilities $P(\cdot | S^n, x)$. In our experience, good performance can be obtained with 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 present in the Bayesian model, such as $\sigma^2_z$. 
The KG decision rule in (17) 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 evaluation, as in (4), where we collect information and improve our approximation in real time, while we are collecting rewards. If, however, we are interested in offline evaluation, as in (6), a more suitable version of the KG policy is given by

$$X^{OIf,n}(S^n, K^n) = \arg\max_x \sum_{S^{n+1}} P\left(S^{n+1} | S^n, x\right) \nu^{KG,n}(S^{x,n}, S^{n+1}),$$  

which encourages experimentation by making decisions based purely on the value of information they provide.

3. Algorithmic extensions: parametric and non-parametric VFAs

The basic Bayesian model introduced in Section 2 requires us to store a large covariance matrix. Even if we discretize the problem, updating this matrix using (9) 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 be integrated with more sophisticated 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.

3.1. Basis functions

Suppose that

$$V(S^x) = \sum_{i=1}^I \theta_i \phi_i(S^x) = \theta^T \phi(S^x),$$

where the basis functions $\phi_i : S^x \to \mathbb{R}$ are given to us in advance. For example, in commodity storage, the state variable may include the price $P$ of the commodity; the basis functions might then include, e.g., $P$, $P^2$ or other polynomials. 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 and Van Roy (1997), whereas more 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 $\theta \sim \mathcal{N}(\theta^0, \Lambda^0)$. This assumption automatically induces a prior on the value function, with $E(V(S^x)) = (\theta^0)^T \phi(S^x)$ and $Cov(V(S^x), V(S^y)) = \phi(S^x)^T \Lambda^0 \phi(S^y)$. Under Assumption 1, we obtain the updating equations

\begin{align}
\theta^{n+1} &= \theta^n + \frac{\hat{v}^{n+1} - (\theta^n)^T \phi(S^n) + \Lambda_n \phi(S^n)}{\sigma^2 + \phi(S^n)^T \Lambda_n \phi(S^n)} \Lambda_n \phi(S^n), \\
\Lambda^{n+1} &= \Lambda^n - \frac{\Lambda_n \phi(S^n)^T \Lambda_n \phi(S^n)}{\sigma^2 + \phi(S^n)^T \Lambda_n \phi(S^n)}.
\end{align}

The Bayesian updating equations (20) and (21) are identical to analogous equations for recursive least squares (Negoescu et al. 2011). They also resemble our original updating equations (8) and (9), but now the covariances are derived from the basis functions. Thus, we are no longer required to guess initial covariances: we can choose a diagonal $\Lambda^0$, whereupon (21) will automatically fill in the off-diagonal entries with empirical covariances. See also the online Appendix (Section EC.2) for a discussion of how this model may be adapted to learn the residual variance $\sigma^2$. 

The KG algorithm remains virtually unchanged. We only need some additional logic to compute (13) and (14), namely $\tilde{V}^n(S^x) = (\theta^n)^T \phi(S^x)$ and $\Sigma^n(S^x, S^y) = \phi(S^x)^T \Lambda^n \phi(S^y)$. 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), because of their simplicity.

3.2. 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 aggregation level \( g = 0, ..., G \). The aggregation structure is said to be hierarchical if \( |S^{(g)}| \leq |S^{(g-1)}| \) and

\[
A^{g-1}(S^x) = A^{g-1}(S^y) \implies A^g(S^x) = A^g(S^y).
\]

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 the weighted sum

\[
\tilde{V}^n(S^x) = \sum_{g=0}^{G} w^{g,n}(S^x) \tilde{V}^{g,n}(S^x),
\]

where \( \tilde{V}^{g,n}(S^x) \) is our time-\( n \) estimated 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 introduce some additional notation. Let \( X^g(S^x) \) be the set of all \( S^y \in S^x \) such that \( A^g(S^y) = A^g(S^x) \) for all \( S^y \in X^g(S^x) \). In words, \( X^g(S^x) \) is the set of all states that belong to the same aggregated state at level \( g \). Let \( G(S^x, S^y) \) be the set of all levels of aggregation shared by \( S^x \) and \( S^y \), that is, all \( g \) with \( A^g(S^x) = A^g(S^y) \).

We retain Assumption 1, and also assume that we receive additional observations \( \hat{v}^{g,n+1} \sim \mathcal{N}(V^g(S^{x,n}), \lambda_{v}^{g,n}(S^{x,n})) \), 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 \( \lambda_{e,n}^{g,n}(S^{x,n}) \) 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 \( V(S^x) \sim N(\bar{V}_0(S^x), \lambda_0(S^x)) \). 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 \( V^g(S^x) \sim N(V(S^x), \eta^g(S^x)) \). 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 \( \bar{V}^g_{g,n}(S^x) \) and variance \( \lambda_{g,n}(S^x) \). The posterior parameters can be obtained (Mes et al. 2011) via the recursive equations

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

and

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

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 (22), with weights

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

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 $\lambda^{g,n}_\varepsilon(S^x,n)$ of the aggregate observations is also unknown, and we estimate it using

$$
\lambda^{g,n}_\varepsilon(S^x) = \sigma^2_\varepsilon + \frac{1}{|\mathcal{A}_g(S^x)|} \sum_{S^y \in \mathcal{X}_g(S^x)} \delta^{g,n}(S^y),
$$
a scheme where we put equal weight on the aggregation bias of $S^y$ for each $S^y \in \mathcal{X}_g(S^x)$.

The aggregated model has many parameters, but we can estimate all of them from the knowledge state $K^n = \{\bar{V}^{g,n}, \lambda^{g,n} | g = 0, \ldots, G\}$, updated via (23) and (24). In the end, we only have a single tunable parameter, the variance $\sigma^2_\varepsilon$ of the ADP observations. A non-informative prior completely avoids the issue of choosing initial means and variances, which was still present in Section 3.1. 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 (13) and (14) are defined in a different way. In the hierarchical model, we have

$$
a^n(S^{n+1}, y) = C(S^{n+1}, y) + \gamma \bar{V}^n(S^{n+1}, y),
$$

$$
b^n(S^{n+1}, y) = \gamma \bar{\sigma}^n(S^{n+1}, y),
$$

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

$$
\bar{V}^n(S^y) = \sum_{g=0}^G w^{g,n+1}(S^y) \bar{V}^{g,n}(S^y)
$$

$$
+ \sum_{g \in \mathcal{G}(S^y, S^x,n)} w^{g,n+1}(S^y) \frac{\lambda^{g,n}_\varepsilon(S^x,n)^{-1}}{\lambda^{g,n}(S^x,n)^{-1} + \lambda^{g,n}_\varepsilon(S^x,n)^{-1}} (\bar{V}^n(S^x,n) - \bar{V}^{g,n}(S^x,n)),
$$

(25)
and
\[
\tilde{\sigma}^n(S^y) = \sum_{g \in G(S^y, S^{x,n})} w_{g,n}^{g,n+1}(S^y) \frac{\lambda_{g,n}^{g,n}(S^{x,n})^{-1} \sqrt{\lambda_{g,n}^{g,n}(S^{x,n}) + \sigma_{g,n}^2}}{\lambda_{g,n}^{g,n}(S^{x,n})^{-1} + \lambda_{g,n}^{g,n}(S^{x,n})^{-1}}.
\]

The predictive distribution of \(\tilde{V}^{n+1}(S^y)\) given \(S^n, K^n, x^n\) now depends on the old beliefs \(\tilde{V}^{g,n}(S^y)\) 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 \(\tilde{V}^{g,n+1}(S^y)\), still unknown at time \(n\). We thus replace \(w_{g,n}^{g,n+1}\) in (25) and (26) by the predictive weights
\[
\tilde{w}_{g,n}^{g,n}(S^y) = \frac{\left(\lambda_{g,n}^{g,n}(S^y)^{-1} + 1\{g \in G(S^y, S^{x,n})\}\lambda_{g,n}^{g,n}(S^y)^{-1}\right)^{-1} + \delta^{g,n}(S^y)^2}{\sum_{g' = 0}^{G} \left(\lambda_{g',n}^{g',n}(S^y)^{-1} + 1\{g' \in G(S^y, S^{x,n})\}\lambda_{g',n}^{g',n}(S^y)^{-1}\right)^{-1} + \delta^{g',n}(S^y)^2}^{-1},
\]
which are equal to \(w_{g,n}^{g,n+1}\), except that \(\delta^{g,n+1}\) is replaced by \(\delta^{g,n}\).

There is a significant computational difference between the original KG algorithm from 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.

4. 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 (19) for the finite-state, finite-action setting. With regard to online learning, it is known in the literature that even optimal policies may converge to suboptimal decisions (Brezzi and 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 $\tilde{V}^n$, which is generally quite difficult to show for approximate Bayesian models (Ryzhov 2015). 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 1.

Our asymptotic analysis is non-Bayesian in nature, and thus we cannot use, e.g., the martingale analysis of Frazier et al. (2009) to establish the asymptotic behaviour of the knowledge state $K^n$. The Bayesian updating equations (8) and (9) 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.

We first present and discuss the assumptions used in our analysis, then state the results. All proofs in this section can be found in the online Appendix (Section EC.1).

**Assumption 2.** 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, which 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 (7) 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. Furthermore, this assumption only concerns $\Sigma^0$, which is a design parameter of the Bayesian model, not of the underlying Markov decision process.

**Assumption 3.** The sets $S$ and $X$ 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 1 by following $\pi$.

Assumption 3 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 (18).

**Assumption 4.** *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 \((\Omega, \mathcal{F}, P)\).

Our first result shows the almost sure componentwise convergence of \( \Sigma^n \). If Assumption 1 were given, this result would follow straightforwardly from a martingale analysis on \( K^n \). However, we emphasize that we do not require Assumption 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 1.** *Let \( \pi \) be a policy, and let \( E^{\pi,x} \subseteq \mathcal{S}^x \) be the set of post-decision states observed infinitely often by following \( \pi \). Under Assumption 2, 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 \( S^y \in \mathcal{S}^x \).*
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 \( \Sigma^\infty(S^x, S^y, \omega) \neq 0 \).*

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 1, but for the KG formula.
**Proposition 2.** The KG factor $\nu^{KG}(S^x, S)$ is a continuous function of the belief parameters $(\bar{V}, \Sigma)$.

We now present our main result showing that the offline KG policy visits every pre-decision state infinitely often with probability 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 1.** Let $E^{KG}$ be the set of pre-decision states observed infinitely often by following the offline KG policy. Under Assumptions 2, 3 and 4, we have

$$ P(E^{KG} = S) = 1. $$

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.** Under Assumptions 2, 3 and 4,

$$ \nu^{KG,n}(S^x, \bar{S}) \to 0, \quad \forall S^x \in S^x, \bar{S} \in S $$

with probability 1.

Building on Proposition 3, we now show that, 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.

**Proposition 4.** Suppose that Assumptions 2, 3 and 4 hold. Let $S^x \in S^x$ and take $\bar{S} \in S$ such that $P(\bar{S} | S, x) > 0$. Then, for any action $\bar{x}$ out of $\bar{S}$, the event that at least one of the post-decision states $S^x, S^{M,x}(\bar{S}, \bar{x})$ is an element of $E^{KG,x}$ occurs with probability 1.
The intuition behind Proposition 4 is as follows. Suppose, for discussion purposes, that Assumption 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 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, we will then learn the values of all post-decision states $S^{M,x}(\bar{S}, \bar{x})$. 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.

For a particular class of problems, relevant in the study of Markov decision processes where means bound variances (Arlotto et al. 2014), Proposition 4 automatically implies full exploration of every post-decision state, as stated in the following corollary.

**Corollary 1.** Suppose that Assumptions 2, 3 and 4 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.

## 5. Experimental study

We tested our approach on four distinct example problems, which present different types of practical challenges. Section 5.1 describes the general setup of the experiments and the policies that were implemented. Section 5.2 briefly describes the test problems and presents the results. Due to space considerations, the details of the test problems are given in the online Appendix (Section EC.3).

### 5.1. Experimental setup

We first discuss the statistical models used to construct VFAs in our test problems. We then briefly discuss the policies that were implemented (a more comprehensive description can be found in the online Appendix, Section EC.3) and the performance metrics used to evaluate their performance.

**Learning models.** We used the hierarchical VFA from Section 3.2 in three test problems, finding that it consistently performed well and improved computational times compared to the
lookup-table VFA. 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. Hierarchical aggregation of the state space was carried out in the same way for any policy that could exploit this approximation structure. Finally, in our last test problem, we used the method of basis functions (Section 3.1) as the state space proved too large to effectively use aggregation.

**Benchmark policies.** We tested both the online and offline versions of the KG policy, from (17) and (19), with sample size $K = 30$. Four benchmark policies were implemented: the value of perfect information (VPI) policy of Dearden et al. (1998); the standard $\varepsilon$-greedy policy, which takes the greedy action $\arg\max_x C(S^n,x) + \gamma \bar{V}^n(S^{x,n})$ with probability $1 - \varepsilon$, and a random action with probability $\varepsilon$; the R-max policy of Brafman and Tennenholtz (2003); and the $E^3$ policy of Kearns and Singh (2002).

Among these policies, R-max and $E^3$ were designed for problems with discrete state spaces, and do not carry over easily to multi-dimensional and continuous problems. Due to these scaling issues, we implemented these benchmarks in only one test problem, which was small enough for discretization to be feasible. The VPI and $\varepsilon$-greedy policies are compatible with correlated Bayesian beliefs and can take advantage of both basis functions and hierarchical aggregation, which speaks to the applicability of the Bayesian framework. The adaptation of VPI to ADP with correlated beliefs is completely new to this paper, as the original presentation in Dearden et al. (1998) used lookup tables.

**Policy evaluation.** We evaluated both the offline and online performance of each policy. Recall from (4) that online performance is evaluated in real time, by adding up the rewards collected by the policy while learning (we stop evaluation at some number $N$ of iterations). By contrast, in the offline setting, we first run the exploration policy $\pi$ for $N$ iterations, which yields the final VFA $\bar{V}^N$. We then define a fixed policy $\rho(\pi)$ induced by this VFA, as in (5), and use (6) to evaluate the performance of this fixed policy. 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.
5.2. Experimental results

We considered four 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; however, more details on each implementation can be found in the online Appendix (Section EC.3).

**Commodity storage problem with stochastic price.** We consider a stylized inventory problem motivated by commodity storage and trading (see, e.g., Secomandi 2010 or Salas and Powell 2018 for applications in energy). We define the state variable $S = (R, P)$, 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 3.2, 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 2(a) shows online performance, expressed in terms of the cumulative revenue collected in real time by each learning policy. Figure 2(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 follows:
1. The hierarchical VFA of Section 3.2 adds value over the basic lookup-table model from Section 2.2. We implemented KG and \( \varepsilon \)-greedy with both types of models, and found that the hierarchical VFA improved the performance of both policies.

2. With the hierarchical VFA, the KG policy, described by (17) or (19) 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 \( E^3 \).

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 hierarchical 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 hierarchical 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.
(a) Online performance. (b) Offline performance.

Figure 3 Experimental comparisons for commodity storage with stochastic supply.

Commodity storage problem 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. Stochastic supply arises in, e.g., renewable energy storage (Zhou et al. 2018). 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; and 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 3(a) and 3(b) show online and offline performance, respectively. The results are largely consistent with Figure 2. 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 problem. In the nomadic trucker problem (Powell 2011), a single truck observes demands that arise randomly in different locations. The state $S = (S^l, S^d, S^k)$ 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. The online Appendix (Section EC.3) provides more details on this problem and on the aggregation structure used by the VFA.

Figure 4 shows the online and offline performance of the KG, VPI, and $\varepsilon$-greedy policies. The $\varepsilon$-greedy policy is implemented both with the hierarchical VFA and with a simple lookup-table approximation. First, we see that exploration helps, since $\varepsilon$-greedy benefits from using $\varepsilon > 0$. Second, the Bayesian exploration policies (VPI and KG) perform considerably better, with KG consistently outperforming the other policies; however, the differences between the Bayesian exploration policies themselves are relatively small. In the online case, exploration results in real costs, and the Bayesian learning policies benefit from focusing on states with either high rewards or high uncertainty, which explains the larger differences with hierarchical $\varepsilon$-greedy. In the offline case, random exploration does not result in real costs, but still provides useful information to update the VFA, which explains the smaller differences between the three policies using the hierarchical
VFA. In Figure 4(b), $\varepsilon$-greedy exhibits a decline in performance after the first 50 iterations, which occurs because the policy gets stuck visiting locations that it has visited before, leading to many costly empty moves.

In our experiments, we saw that every policy that used the hierarchical VFA could be sensitive to the value of $\sigma^2_\varepsilon$. For 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 well with relatively large initial values of the VFA (also recommended in Sec. 4.9.1 of Powell 2011 for ADP in general) and moderate values of $\sigma^2_\varepsilon$.

**Freight consolidation problem.** Our last example also belongs to the transportation domain, but has broader applicability to, e.g., order fulfillment problems. The example is derived from Pérez Rivera and Mes (2017) and involves the periodic assignment of loads to a high-capacity vehicle. Loads are characterized by earliest and latest departure times, as well as a destination. Revenues and costs are determined by the combination of destinations for the assigned loads. Unfulfilled demand is backlogged and may have to be transported using other means, at a detriment to revenue. The decision-maker has probabilistic knowledge about arrival times and other characteristics of loads. This problem may be viewed as a variant of the delivery dispatching problem (Minkoff 1993).

The problem setup generally follows Pérez Rivera and Mes (2017), with some modifications to convert it to the infinite-horizon discounted setting (for more details, see the online Appendix, Section EC.3). Due to the large size of this problem, we found that basis functions (Sec. 3.1) led to a more computationally tractable implementation than hierarchical aggregation (Sec. 3.2). We considered two types of VFAs: one of these (VFA2) includes one feature for each destination and time-window combination, as well as other features describing the number of loads and different destinations of these loads, where we distinguish between urgent loads, less urgent loads, and
loads that have not yet been released. The other type (VFA1) omits the features for individual destinations and time windows, but keeps the descriptive features.

Figure 5 presents experimental results for the KG, VPI, and $\varepsilon$-greedy policies. In contrast with the previous problem, $\varepsilon$-greedy no longer benefits from exploration in the online case, since the information gained by exploration does not outweigh the real costs resulting from exploration. Furthermore, $\varepsilon$-greedy experiences numerical instability when using the more informative VFA2 (a known disadvantage of the use of basis functions), which leads to rapidly declining performance after 50 iterations. The Bayesian exploration policies clearly benefit from exploration and VPI does benefit from using VFA2. It should be noted that VFA2 greatly increases the computational cost of any policy, since the matrix calculations used to find posterior means and covariances become more expensive. For this reason, we found that it was not practical to implement KG with VFA2 at all; but, by the same token, we found that KG with VFA1 outperformed other policies with VFA2, suggesting that KG is able to realize greater benefits from a more parsimonious approximation architecture.

In all of our examples, KG required more computational effort than any other policy. However, in the online setting, the additional 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, days or even longer, justifying the extra effort involved in using KG.
6. Conclusion

We have proposed a new exploration strategy for approximate dynamic programming. Our approach uses a Bayesian belief model to quantify uncertainty about the value function, and calculates 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. The greatest practical advantage of the approach is that it can be incorporated into two powerful classes of value function approximations, namely basis functions and hierarchical aggregation. We evaluated this approach on four distinct resource allocation problems. In our experiments, the new method outperformed several benchmarks on all test problems after an initial number of replications. This performance comes at the cost of increased computation time, but may provide practical advantages especially in the online setting.

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 implementing Gaussian process regression and adapting the procedure in Scott et al. (2011). 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.

Although our presentation has focused on using the knowledge gradient together with Bayesian beliefs, in principle one could use the Bayesian VFAs with other criteria, such as the VPI policy implemented in Section 5, or the UCB method of Srinivas et al. (2010). While such criteria may be practical when computation time is a major concern, we note that they do not explicitly consider correlated beliefs during decision-making. By contrast, the KG method considers correlations when evaluating actions, leading to more systematic exploration of the state space.

Acknowledgments

The authors are grateful to the Area Editor, Associate Editor and two referees for their thorough reading of the paper and helpful comments. This research was supported in part by AFOSR contract FA9550-08-1-0195.
Endnotes

1. Preliminary presentations of ideas from Sections 2 and 3.1 appeared in conference proceedings (Ryzhov and Powell 2010, 2011) and were summarized again in the monograph by Powell (2011).

2. To avoid notational clutter, we will write max_x from here, under the implicit assumption that x ∈ X.

3. Ribeiro and Szepesvári (1996) and Szepesvári and Littman (1999) introduce a heuristic form of dependence between states into Q-learning, but this scheme requires tuning of a stepsize.

References


Appendices

Section EC.1 gives complete proofs for results stated in the main text of the paper. Section EC.2 discusses an extension of the Bayesian learning model to handle unknown variance. Section EC.3 provides the full implementation details for our numerical examples.

EC.1. Appendix: proofs

Below, we give the full technical proofs for results that were stated in the main text.

EC.1.1. Proof of Proposition 1

From (9), 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 $S^x \in \mathcal{E}_{\pi,x}(\omega)$. 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 (9) as

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

Suppose that $\lim_{n \to \infty} \Sigma^n(S^x, S^x, \omega) > 0$. Then, it follows from (EC.1) 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 $E^\pi,\pi(x) \neq S^x$, we can partition $\Sigma^n(\omega)$ as

$$\Sigma^n(\omega) = \begin{pmatrix} \Sigma^{E,n}(\omega) & \Sigma^{cross,n}(\omega) \\ \Sigma^{cross,n}(\omega)^T & \Sigma^{E^\pi,n}(\omega) \end{pmatrix}, \tag{EC.2}$$

where $\Sigma^{E,n}(\omega)$ contains variances and covariances for only those states in $E^\pi,\pi(x)$, while $\Sigma^{E^\pi,n}(\omega)$ contains covariance information only for states in the complement of $E^\pi,\pi(x)$. We define $\tau$ to be the last time the policy $\pi$ visits a state $S^x \notin E^\pi,\pi(x)$ on $\omega$. Without loss of generality, we can assume $\tau = -1$, since we can just take $\Sigma^0 = \Sigma^{\tau+1}$. 
By Assumption 2, \( \Sigma^0 \) is invertible. As in (EC.2), we can partition \((\Sigma^0)^{-1}\) to obtain

\[
(\Sigma^0)^{-1}(\omega) = \begin{pmatrix}
\bar{\Sigma}_E,0(\omega) & \bar{\Sigma}_{\text{cross},0}(\omega) \\
(\bar{\Sigma}_{\text{cross},0}(\omega))^T & \bar{\Sigma}_{E',0}(\omega)
\end{pmatrix}.
\]

If \( \tau = 0 \), we can apply the Sherman-Morrison formula to (9) to write

\[
\lim_{n \to \infty} (\Sigma^n)^{-1}(\omega) = (\Sigma^0)^{-1}(\omega) + \sigma^2 \varepsilon \lim_{n \to \infty} nD(\omega),
\]

where \( D \) is a diagonal matrix with

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

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

\[
\lim_{n \to \infty} \Sigma^{E',n}(\omega) = \left(\bar{\Sigma}_{E',0}(\omega)\right)^{-1}.
\]

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 \( S^x, S^y \notin E^{x,x}(\omega) \). By Assumption 2, \( \Sigma^0 \) 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 1). Assume, 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 (9) 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 \( S^x, S^y \notin E^{π,x}(ω) \), 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 2, our prior beliefs about these two alternatives are correlated. We can then express the true values of \( S^x \) and \( S^y \) as

\[
μ(S^x) = a_x \cdot C + b_x \cdot Z_x + c \cdot Z_{x,y},
\]
\[
μ(S^y) = a_y \cdot C + b_y \cdot Z_y + d \cdot Z_{x,y},
\]

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 \( Σ^0 = Σ^{τ+1} \). 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 \( Σ^ω \) is identical for this problem and for the DP, we conclude that \( Σ^∞(S^x, S^y, ω) \neq 0 \).

**EC.1.2. Proof of Proposition 2**

Recall from (15) that

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

where \( a \) and \( b \) are computed using (13) and (14), 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

\[
c_i = \frac{a(S, y_i) - a(S, y_{i+1})}{b(S, y_{i+1}) - b(S, y_i)}, \tag{EC.3}
\]

which is a rational function of the components of \( a \) and \( b \). The denominator in (EC.3) 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 \( ν^{KG}(S^x, S) \), it is enough to consider only those parameters \((\bar{V}, Σ)\) 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 \( (V', \Sigma') \) be such that \( |a(S, z) - a'(S, z)| < \delta \) and \( |b(S, z) - b'(S, z)| < \delta \). The only change in \( \nu^{KG,n}(S', S) \) will be a new breakpoint, so that

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

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

**EC.1.3. Proof of Theorem 1**

We assume that a suitable set of measure zero has been removed from the outcome space. Suppose that \( E^{KG}(\omega) \neq S \). As in Proposition 1, we partition \( \Sigma^n(\omega) \) as

\[
\Sigma^n(\omega) = \begin{pmatrix}
\Sigma^{E,n}(\omega) & \Sigma^{cross,n}(\omega) \\
(\Sigma^{cross,n}(\omega))^T & \Sigma^{E',n}(\omega)
\end{pmatrix}.
\]

By Proposition 1, we have

\[
\Sigma^n(\omega) \rightarrow \begin{pmatrix}
0 & 0 \\
0 & \Sigma^\infty(\omega)
\end{pmatrix},
\]  \hspace{1cm} (EC.4)

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

By Assumption 3, there must exist states \( S \in E^{KG}(\omega) \) and \( \tilde{S} \notin E^{KG}(\omega) \) such that, for at least one action \( x \in \mathcal{X} \), we have \( P(\tilde{S} | S, x) > 0 \). 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 $(S,x) \notin E^{KG,\bar{x}}(\omega)$. Furthermore, $(\bar{S},\bar{x}) \notin E^{KG,\bar{x}}(\omega)$ for all $\bar{x}$, since $\bar{S}$ is visited only finitely many times.

From (EC.4), we have

$$\Sigma^\infty (S^M,x(\bar{S},\bar{x}),S^\bar{x}) \neq 0,$$

(EC.5)

$$\Sigma^\infty (S^\bar{x},S^\bar{x}) \neq 0.$$  

(EC.6)

By Proposition 2, we have

$$\nu^{KG,n}(S^\bar{x},\bar{S},\omega) \rightarrow \nu^{KG,\infty}(S^\bar{x},\bar{S},\omega),$$

where $\nu^{KG,\infty}(S^\bar{x},\bar{S},\omega) > 0$. The fact that the limit is strictly positive is ensured by Assumption 4.

In (15), 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 (EC.5). There must therefore be at least one breakpoint. The function $f$ in (15) has no zeros on the real line, so we conclude that $\nu^{KG,\infty}(S^\bar{x},\bar{S},\omega) > 0$. Since $P(\bar{S} \mid S,x) > 0$, it follows that

$$\lim_{n \to \infty} \sum_{S'} P(S' \mid S,x) \nu^{KG,n}(S^\bar{x},S',\omega) \geq P(\bar{S} \mid S,x) \nu^{KG,\infty}(S^\bar{x},\bar{S},\omega) > 0.$$  

(EC.7)

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 $P(S' \mid S,y) = 0$ for all $S' \notin E^{KG}(\omega)$. Furthermore, for any $S' \in E^{KG}(\omega)$, we have

$$\Sigma^n (S^M,x(S',x'),S^y) \rightarrow 0$$

due to (EC.4). By Proposition 2, it follows that $\nu^{KG,n}(S^y,S',\omega) \rightarrow 0$, whence

$$\sum_{S'} P(S' \mid S,y) \nu^{KG,n}(S^y,S',\omega) \rightarrow 0.$$  

(EC.8)
We now put together (EC.7) and (EC.8). Let \( \varepsilon = \nu^{\text{KG},\infty} (S^x, S', \omega) \). Then, there exists an integer \( K_\omega \) such that, for all \( n \geq K_\omega \), we have

\[
\left| \sum_{S'} P(S' \mid S, x) \nu^{\text{KG},n} (S^x, S', \omega) - \lim_{n \to \infty} \sum_{S'} P(S' \mid S, x) \nu^{\text{KG},n} (S^x, S', \omega) \right| < \frac{\varepsilon}{2},
\]

\[
\sum_{S'} P(S' \mid S, y) \nu^{\text{KG},n} (S^y, S', \omega) < \frac{\varepsilon}{2}.
\]

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 \( E^{\text{KG}} (\omega) = S \).

**EC.1.4. Proof of Proposition 3**

By Proposition 2, every KG factor converges to a limit. Suppose that \( \nu^{\text{KG},n} (S^x, \bar{S}, \omega) \to \nu^{\text{KG},\infty} (S^x, \bar{S}, \omega) \) for some \( S^x \in S^x \) and \( \bar{S} \in S \). It follows that \( S^x \notin E^{\text{KG},x} (\omega) \), otherwise we would have \( \Sigma^n (S^x, S^x, \omega) \to 0 \), which would imply that \( \nu^{\text{KG},n} (S^x, \bar{S}, \omega) \to 0 \) by continuity.

Because \( S \) is visited infinitely often by Theorem 1, there must be at least one action \( y \) such that \( S^y \in E^{\text{KG},x} (\omega) \). For this action, \( \nu^{\text{KG},n} (S^y, S', \omega) \to 0 \). We can then repeat the argument concluding the proof of Theorem 1 to find that, after some time \( K_\omega \), the offline KG policy will prefer action \( x \) to action \( y \), which implies that \( S^x \in E^{\text{KG},x} (\omega) \) and therefore \( \nu^{\text{KG},n} (S^x, \bar{S}, \omega) \to 0 \). We conclude that every KG factor must converge to zero under the offline KG policy.

**EC.1.5. Proof of Proposition 4**

Suppose that \( S^x, S^{M,x} (\bar{S}, \bar{x}) \notin E^{\text{KG},x} (\omega) \). By Proposition 1, it must be the case that \( \Sigma^n (S^{M,x} (\bar{S}, \bar{x}), S^x, \omega) \), \( \Sigma^n (S^{M,x} (\bar{S}, \bar{x}), S^{M,x} (\bar{S}, \bar{x}), \omega) \), and \( \Sigma^n (S^x, S^x, \omega) \) converge to non-zero limits, as in (EC.5) and (EC.6). Applying Assumption 4 as in the proof of Theorem 1, we find that \( \nu^{\text{KG},n} (S^x, \bar{S}) \) converges to a strictly positive limit. This contradicts Proposition 3, which states that all KG factors must converge to zero.
EC.2. Appendix: learning the unknown noise variance

One limitation of the Bayesian model from Section 2.2 is that the observations $\hat{v}^{n+1}$ are assumed to have known variance $\sigma^2_v$. In practice, this creates a tunable parameter that requires additional computational effort to optimize. In the following, we present an extension that explicitly models the noise variance as a random variable, and updates a set of beliefs about this quantity over time.

Let $\rho = \sigma_v^{-2}$ be the precision of the observations. We assume that $\rho$ is unknown and impose the prior distribution $\rho \sim \text{Gamma}(\alpha^0, \beta^0)$, where $\alpha^0, \beta^0 > 0$ are fixed. We then suppose that the conditional prior distribution of $V$, given $\rho$, is multivariate normal with mean $\bar{V}^0$ and covariance matrix $\rho^{-1}\Sigma^0$. In Bayesian statistics, the joint distribution of $(V, \rho)$ is known as a “multivariate normal-gamma” prior. It can be shown (DeGroot 1970) that the marginal distribution of $V$ under this model is a multivariate Student’s $t$-distribution (Kotz and Nadarajah 2004), by analogy with classical statistics where this distribution is used to model observations with unknown variance. In this setting, Assumption 1 reads as follows.

**Assumption EC.1.** Given $V$ and $\rho$, the ADP observation $\hat{v}^{n+1}$ follows the conditional distribution $\mathcal{N}(V(S^x), \rho^{-1})$ and is conditionally independent of past observations.

As in Section 2.2, Assumption EC.1 allows us to update our entire approximation $\bar{V}^n$ using a single scalar observation $\hat{v}^{n+1}$. Equations (8)-(9) now become

$$\bar{V}^{n+1}(S^x) = \bar{V}^n(S^x) + \frac{\hat{v}^{n+1} - \bar{V}^n(S^x)}{1 + \Sigma^n(S^x, S^x)n}(S^x, S^x)n),$$
$$\Sigma^{n+1}(S^x, S^y) = \Sigma^n(S^x, S^y) - \frac{\Sigma^n(S^x, S^x)n)(S^x, S^x)n)}{1 + \Sigma^n(S^x, S^x)n),}$$
$$\alpha^{n+1} = \alpha^n + \frac{1}{2},$$
$$\beta^{n+1} = \beta^n + \frac{(\hat{v}^{n+1} - \bar{V}^n(S^x))^2}{2(1 + \Sigma^n(S^x, S^x)n))}.$$

This learning model thus has essentially the same complexity as the one in Section 2.2; the only addition consists of two scalar posterior parameters $\alpha^n, \beta^n$. 
The KG logic of Section 2.3 can now be applied. The only difference (Han et al. 2016) is that now (15) should be rewritten as
\[
\mathbb{E}^n_x \max_y \tilde{Q}^{n+1} (S^{n+1}, y) = \left( \max_y a^n (S^{n+1}, y) \right) + \sum_{y_i \in A(S^{n+1})} \left[ b^n (S^{n+1}, y_{i+1}) - b^n (S^{n+1}, y_i) \right] g_{2\alpha^n} (-|c_i|), \quad \text{(EC.9)}
\]
where
\[
g_s (t) = \frac{s + t^2}{s - 1} \psi_s (t) + t \Psi_s (t)
\]
and \(\psi_s, \Psi_s\) are the pdf and cdf of Student’s \(t\)-distribution with \(s\) degrees of freedom. Equations (13)-(14), which define the vectors \(a^n\) and \(b^n\) in (15), are now replaced by
\[
a^n (S^{n+1}, y) = C (S^{n+1}, y) + \gamma \tilde{V}^n (S^{M,x} (S^{n+1}, y)), \quad \text{(EC.10)}
\]
\[
b^n (S^{n+1}, y) = \gamma \Sigma^n (S^{M,x} (S^{n+1}, y), S^{x,n}) \sqrt{\frac{\beta^n}{\alpha^n (1 + \Sigma^n (S^{x,n}, S^{x,n}))}}, \quad \text{(EC.11)}
\]
After this, the computation of the KG policy proceeds as before (in particular, the breakpoints \(c_i\) in (EC.9) are computed from \(a^n\) and \(b^n\) in the same way). The only difference is that the knowledge gradient computations now use the tail properties of the \(t\)-distribution, rather than the standard normal.

In fact, the multivariate normal-gamma prior may also be used in conjunction with the linear VFA of Section 3.1. Once again, we assume that \(V (S^x) = \theta^T \phi (S^x)\) and impose the prior distribution \(\rho \sim \text{Gamma} (\alpha^0, \beta^0)\). The conditional prior distribution of \(\theta\), given \(\rho\), is \(\mathcal{N} (\theta^0, \rho^{-1} \Lambda^0)\). Under Assumption EC.1, this produces the update
\[
\theta^{n+1} = \theta^n + \frac{\hat{v}^{n+1} - (\theta^n)^T \phi (S^{x,n})}{1 + \phi (S^{x,n})^T \Lambda^n \phi (S^{x,n})} \Lambda^n \phi (S^{x,n}),
\]
\[
\Lambda^{n+1} = \Lambda^n - \frac{\Lambda^n \phi (S^{x,n}) \phi (S^{x,n})^T \Lambda^n}{1 + \phi (S^{x,n})^T \Lambda^n \phi (S^{x,n})},
\]
\[
\alpha^{n+1} = \alpha^n + \frac{1}{2},
\]
\[
\beta^{n+1} = \beta^n + \frac{\left( \hat{v}^{n+1} - (\theta^n)^T \phi (S^{x,n}) \right)^2}{2 \left( 1 + \phi (S^{x,n})^T \Lambda^n \phi (S^{x,n}) \right)}.
\]
The KG computation is again given by (EC.9), but (EC.10)-(EC.11) are replaced by

\[
a^n(S^{n+1}, y) = C(S^{n+1}, y) + \gamma (\theta^y)^T \phi(S^{M,y} (S^{n+1}, y)),
\]

\[
b^n(S^{n+1}, y) = \gamma \phi(S^{M,y} (S^{n+1}, y))^T \Lambda^n \phi(S^{x,n}) \left[ \frac{\beta^n}{\alpha^n (1 + \phi(S^{x,n})^T \Lambda^n \phi(S^{x,n}))} \right].
\]

Although the performance of this model may still be influenced by the initialization of \(\alpha^0, \beta^0\), these parameters will be adjusted over time and thus the model is less susceptible to misspecification of the noise variance. Furthermore, since Student’s t-distribution has heavier tails than the normal distribution, the value of information will tend to be higher under this framework and thus KG will conduct more exploration. Moreover, KG will incur virtually the same computational cost here as in the known-variance model, since the main computational bottleneck in (EC.9) is the calculation of the breakpoints, which remains unchanged.

**EC.3. Appendix: details of experimental settings**

In this section, we give more details on the benchmark policies and test problems from Section 5.2.

**EC.3.1. Description of learning policies**

Five different types of policies were implemented; their descriptions are as follows.

**Knowledge gradient (KG).** We tested both the online and offline versions of the KG policy, from (17) and (19), 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 3. 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 3, 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 3.1, this is \( \phi(S^{x,n})^T \Lambda^n \phi(S^{x,n}) \), and in Section 3.2, 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 and Lee 2011) from the global optimization literature. However, its adaptation to ADP with correlated beliefs is completely new to this paper. Like KG, the VPI policy can work well with both basis functions and hierarchical aggregation. We view this as an additional argument in favour of our Bayesian framework; although we mainly focus on the KG policy in this paper, our Bayesian models have even broader potential since they may be combined with other algorithms.

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

**R-max.** The R-max policy of Brafman and 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

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

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 and 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.

**EC.3.2. Commodity storage problem with stochastic price**

We considered two versions of a stylized inventory problem motivated by commodity storage and trading, an application that has seen recent attention, e.g., in Secomandi (2010). We have simplified this problem for easier benchmarking, to make some of the competing policies computationally tractable, and to allow us to run more simulations for statistically valid comparisons.

The first version of the problem considers an asset held in storage. At any point in time, we can buy from or sell to the spot market. Figure 1(a) illustrates the decision variable in this problem using the example of electricity stored in a battery. Our decision $x^n$ at time $n$ depends on the state variable $S^n = (R^n, P^n)$, where $R^n$ is the amount currently in storage and $P^n$ is the current spot price. If $x^n \geq 0$, we buy energy from the market, whereas we sell if $x^n < 0$. The single-period revenue or cost is then

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

since we pay a cost to buy, and receive revenue from selling. Our objective is to maximize the long-term profit. The spot price was assumed to follow a geometric Ornstein-Uhlenbeck process with mean reversion parameter 0.0633 and volatility parameter 0.2, ensuring a sufficient level of noise in the problem. The post-decision state, given a decision $x^n$, is computed by $R^{x,n} = R^n + x^n$ and $P^{x,n} = P^n$. To find the next pre-decision state, we take $R^{n+1} = R^{x,n}$ and simulate $P^{n+1}$ from our price process.

We discretized $R^n$ on the interval $[0, 1]$ in increments of 0.01, representing the percentage of the battery’s capacity that is being used. The decision $x^n$ was similarly discretized on $[-R^n, 1 - R^n]$. 

However, we did not discretize the price variable in the problem. When running a policy, we always kept track of the continuous value of $P^n$, and only discretized prices when calling the value function approximation. Thus, we are still solving a continuous problem; see Section 8.1.1 of Powell (2011) for a discussion of this point. The prior approximation $\bar{V}^0$ was uniformly set to a very large value, as discussed in Section 4.9 of Powell (2011), and (7) was used to set a prior covariance for the lookup table VFA.

For the hierarchical VFA of Section 3.2, 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. We found that, as the discretization became finer, it was more important to distinguish between similar prices than similar storage quantities. This VFA was used for the hierarchical KG method, as well as the $\epsilon$-greedy and VPI policies. Thus, all competing policies had access to the benefits of correlated beliefs.

EC.3.3. Commodity storage problem with stochastic supply

The second version of our commodity storage problem has a vector decision variable. Figure 1(b) illustrates using an example where electricity is obtained exogenously from a wind farm, stored in
a battery, and then used to satisfy demand. If there is not enough energy in storage to cover all the demand, we have to purchase the remainder from the spot market. This problem is similar to the one in Section EC.3.2, but now we can also receive supply from an exogenous process.

We considered a simplified version of the problem where the demand in each time period was set to a fixed number $D$. Our state variable was thus $S^n = (R^n, W^n)$, where $0 \leq R^n \leq \bar{R}$ is the amount in storage as before, and $W^n$ is the stochastic supply process. In our simulations, we designed $W^n$ by fitting a mean-reverting process to historical wind speed data, then converting these wind speeds to power output.

From Figure 1(b), we see that the decision is a vector $x^n = (x^{wr,n}, x^{wd,n}, x^{gr,n}, x^{gd,n}, x^{rd,n})$. We reduce this vector to three dimensions by observing that

$$E^n = x^{wr,n} + x^{wd,n}$$
$$D = x^{wd,n} + x^{gd,n} + x^{rd,n},$$

and setting $x^{wd,n} = E^n - x^{wr,n}$, and $x^{gd,n} = D - (E^n - x^{wr,n}) - x^{rd,n}$. We constrain $0 \leq x^{wr,n} \leq E^n$ to ensure the positivity of $x^{wd,n}$. We assume that oversupplying demand is not allowed, so we also constrain $0 \leq (E^n - x^{wr,n}) + x^{rd,n} \leq D$. Finally, we add the constraints

$$0 \leq \rho^c x^{gr,n} \leq \bar{R} - R^n$$
$$0 \leq \rho^c x^{wr,n} \leq \bar{R} - R^n$$
$$0 \leq x^{rd,n} \leq R^n + \rho^c (x^{wr,n} + x^{gd,n})$$
$$\rho^d x^{rd,n} \leq D,$$

where $\rho^c$ and $\rho^d$ represent loss rates for the storage device (e.g., energy dissipation from the battery). Once the decision has been made, the post-decision storage level is given by

$$R^{r,n} = \min \left( R^n + \rho^c (x^{gr,n} + x^{wr,n}) - x^{rd,n}, \bar{R} \right),$$

indicating that there is a cap on how much we can store.
The single-period contribution function depends on a selling price \( P_s \), a buying price \( P_g \) (if we buy from the spot market), and an extra penalty cost \( P_p \) imposed when our purchase \( x^{gr,n} + x^{gd,n} \) exceeds some level \( x^{max} \). To reduce the size of the state variable, we assumed constant prices \( P_s = 0.14, P_g = 0.12 \) and \( P_p = 0.5 \). We also let \( D = 100 \) be the constant demand, with \( x^{max} = 75 \). The contribution is then calculated as

\[
C(S^n, x^n) = P_s D - P_g (x^{gr,n} + x^{gd,n}) - P_p (x^{gr,n} + x^{gd,n} - x^{max})^+. 
\]

Maximizing the long-term contribution is equivalent to minimizing the long-term cost of meeting demand.

Even with these simplifications, we found that this discretized problem was too large to maintain a covariance matrix directly on the state space. For this reason, we only used the hierarchical VFA of Section 3.2 and compared KG, VPI, and \( \varepsilon \)-greedy. The R-max and \( E^3 \) policies are tied to a discrete value function representation and do not translate easily to the hierarchical model.

**EC.3.4. Nomadic trucker problem**

In the nomadic trucker problem (Powell 2011), a single truck observes demands that arise randomly in different locations (e.g., cities in the US), modeled as elements of a set \( \mathcal{L} \). The trucker travels between these locations to accept those loads that maximize the long-term reward. The state \( S^n = (S_{l,n}, S_{d,n}, S_{k,n}) \) is a vector of attributes representing, respectively, the current location of the trucker, the current day of the week, and the trailer type.

The decision \( x^n \) is modeled as a binary vector with \( x^n_i = 1 \) if we choose the \( i \)th possible decision and \( \sum_i x^n_i = 1 \). 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 to a different location without accepting any load. For each location \( i \in \mathcal{L}, \) the number \( 0 \leq b_i \leq 1 \) represents the probability that a load originating at location \( i \) will appear at a given time step. We let \( p_{ij}^d = p_d b_i (1 - b_j) \) be the probability that, on a given day \( d \) of the week, a load from \( i \) to \( j \) will appear, where \( p_d \) is the probability of loads appearing on day \( d \).
When we select the decision $x_n$, our post-decision state $S_{x,n}$ is determined as if we had already arrived at the destination (that is, we change the location and day-of-week components of the state to the values they will have upon our arrival). The costs $C(S_n,x_n)$ generally depend on the distance $d_{ij}$ that we travel between the current location $i$ and the chosen destination $j$, either with or without a load.

In our implementation, locations lie on a $16 \times 16$ grid placed on a square area of $1000 \times 1000$ miles. Each location is described by coordinates $(x_i,y_i)$, and the origin probabilities are given by

$$b_i = \rho \left( 1 - \frac{h(x_i,y_i) - h_{\text{min}}}{h_{\text{max}} - h_{\text{min}}} \right),$$

where $\rho$ is the arrival intensity of loads and $h$ is the six-hump camelback function

$$h(x,y) = 4x^2 - 2.1x^4 + \frac{1}{3}x^6 + xy - 4y^2 + 4y^4$$

on the domain $[1.5,2] \times [1,1]$, properly scaled to the domain $[0,0] \times [1000,1000]$. The values $h_{\text{min}} = \min_{i \in L} h(x_i,y_i)$ and $h_{\text{max}} = \max_{i \in L} h(x_i,y_i)$ are used to scale $h(x_i,y_i)$ between $[0,1]$. We set $\rho = 1$, which corresponds to an average of approximately 93 outgoing loads from the most popular origin location on the busiest day of the week. We use the load probabilities $p^d = (1,0.8,0.6,0.7,0.9,0.2,0.1)$ for $d$ from Monday to Sunday, representing a situation where loads are more likely to appear at the beginning of the week and toward the end. The cost function is given by

$$C(S,x) = \begin{cases} -d_{ij} & \text{if we choose to move from } i \text{ to } j \text{ without taking a load} \\ r^k d_{ij} b_i & \text{if we move from } i \text{ to } j \text{ with a load of type } k. \end{cases}$$

The trailer type attribute can be either small, medium, or large, and varies in a cyclic fashion, irrespective of the remaining attributes. Larger trailer types result in higher rewards, with $r^1 = 1$, $r^2 = 1.5$, and $r^3 = 2$. Rewards are discounted using $\gamma = 0.95$.

Table EC.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.
### Table EC.1  Aggregation structure for the nomadic trucker problem.

<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 · 3 · 7 = 5376</td>
</tr>
<tr>
<td>1</td>
<td>8 × 8</td>
<td>*</td>
<td>*</td>
<td>64 · 3 · 7 = 1344</td>
</tr>
<tr>
<td>2</td>
<td>4 × 4</td>
<td>*</td>
<td>*</td>
<td>16 · 3 · 7 = 336</td>
</tr>
<tr>
<td>3</td>
<td>4 × 4</td>
<td></td>
<td>*</td>
<td>16 · 7 = 112</td>
</tr>
<tr>
<td>4</td>
<td>2 × 2</td>
<td>-</td>
<td>*</td>
<td>4 · 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>

**EC.3.5. Freight consolidation problem**

In the freight consolidation problem, a decision-maker receives loads to be transported and periodically decides which of these loads should be consolidated in a high-capacity vehicle to be dispatched in the current period. The full mathematical model for this problem is given in Pérez Rivera and Mes (2017) and so we do not repeat it here; for example, the state variable is formally defined in eq. (1) of Pérez Rivera and Mes (2017), the decision is described by eqs. (2a)-(2f), etc. Below, we describe the modifications made to this model in order to adapt it to the setting of our paper.

Our implementation is based on Instance $I^L_6$ from Pérez Rivera and Mes (2017), a delivery-only variant (no pickups). As originally presented, this instance was a finite-horizon minimization problem without exploration; we adapted it to the setting of infinite-horizon, discounted maximization with exploration as follows. First, the discount factor was chosen to be $\gamma = 0.99$. Second, we introduced revenues per container shipped, varying between 325 and 825 depending on the destination. These numbers are chosen in order to be comparable to the costs in Pérez Rivera and Mes (2017); for instance, travel costs are between 250 and 1000 depending on which destinations are visited, while the variable cost per container is between 50 and 100. The probabilities of freights of various types were taken from Pérez Rivera and Mes (2017).

Instance $I^L_6$ has 12 different destinations, 3 possible values for the release day, and 3 possible values for the time window. Because of the large number of dimensions in this problem, we
implemented the method of basis functions as laid out in Section 3.1 and considered two VFAs. The more informative VFA, denoted by VFA2 in Section 5.2, includes a dummy variable for each destination/time-window combination, as well as features counting the numbers of loads headed to each destination. This second set of basis functions distinguishes between (i.e., uses separate features for) urgent loads, less urgent loads, and loads that have not yet been released. This VFA is identical to “VFA3” in Pérez Rivera and Mes (2017). In Section 5.2 we also considered a less informative VFA, denoted by VFA1, which omits the dummy variables for destination/time-window combinations.

The implementation of the KG and VPI policies is new to the present paper, as exploration was not considered in Pérez Rivera and Mes (2017). As mentioned in Section 5.2, although KG with VFA2 became computationally expensive, KG with VFA1 turned out to perform better than the other policies did with the richer VFA2.

References


Powell WB (2011) *Approximate dynamic programming: solving the curses of dimensionality (2nd ed.)* (John Wiley and Sons).