OPERATIONS RESEARCH
Vol. 61, No. 3, May–June 2013, pp. 644–665
ISSN 0030-364X (print) | ISSN 1526-5463 (online)

http://dx.doi.org/10.1287/opre.2013.1164

METHODS

Optimal Sequential Exploration: Bandits, Clairvoyants, and Wildcats

David B. Brown, James E. Smith
Fuqua School of Business, Duke University, Durham, North Carolina 27708 {dbbrown@duke.edu, jes9@duke.edu}

This paper was motivated by the problem of developing an optimal policy for exploring an oil and gas field in the North Sea. Where should we drill first? Where do we drill next? In this and many other problems, we face a trade-off between earning (e.g., drilling immediately at the sites with maximal expected values) and learning (e.g., drilling at sites that provide valuable information) that may lead to greater earnings in the future. These "sequential exploration problems" resemble a multiarmed bandit problem, but probabilistic dependence plays a key role: outcomes at drilled sites reveal information about neighboring targets. Good exploration policies will take advantage of this information as it is revealed. We develop heuristic policies for sequential exploration problems and complement these heuristics with upper bounds on the performance of an optimal policy. We begin by grouping the targets into clusters of manageable size. The heuristics are derived from a model that treats these clusters as independent. The upper bounds are given by assuming each cluster has perfect information about the results from all other clusters. The analysis relies heavily on results for bandit superprocesses, a generalization of the multiarmed bandit problem. We evaluate the heuristics and bounds using Monte Carlo simulation and, in the North Sea example, we find that the heuristic policies are nearly optimal.

Subject classifications: dynamic programming; multiarmed bandits; bandit superprocesses; information relaxations.
Area of review: Decision Analysis.
History: Received March 2012; revisions received August 2012, October 2012; accepted February 2013. Published online in *Articles in Advance* May 24, 2013.

1. Introduction

This paper was motivated by the problem of developing an optimal policy for exploring an oil and gas field in the North Sea, off the coast of Norway: Where should we drill first? Where do we drill next? Clearly the choices for later targets may depend on what we observe at earlier wells—particularly when drilling "wildcat" wells in regions that are not well understood. For example, positive results in one region of the field may lead us to explore other targets that are nearby or share geologic features. Conversely, negative results may lead us to explore other regions or quit altogether.

Researchers considering R&D projects face similar problems: with dependent projects, which projects should they pursue first? Similarly, in Web-based advertising, one must decide which advertisement to show to a user visiting a Web page. Here too the performances of similar advertisements (e.g., for different shoes) may be correlated and the sequence of ads chosen may reflect the responses to earlier ads. In these and many other problems, we face a trade-off between earning (e.g., drilling immediately at targets with large expected values) and learning (e.g., drilling at targets that provide valuable information) that may lead to greater earnings in the future.

These "sequential exploration problems" can be viewed as variations on the classic multiarmed bandit problem. A multiarmed bandit consists of a set of arms, each corresponding to a Markov reward process. In each period, the decision maker (DM) selects an arm to play and receives a random reward; the state of the selected arm then randomly changes and the process begins again with the DM choosing which arm, if any, to play next. If the state changes for the arms are independent, the multiarmed bandit problem has a very elegant solution: Gittins and Jones (1974) showed that it is optimal to pull the arm with the largest "Gittins index," where these Gittins indices can be calculated by considering each arm in isolation. Thus, with independent arms, the multiarmed bandit problem can be solved by decomposing the problem into a series of computationally manageable subproblems, one for each arm.

The sequential exploration problems that we consider are like multiarmed bandits, but with dependent arms. Although the problems can be formulated as stochastic dynamic programs, without independence, the models suffer the "curse of dimensionality" and may be difficult to solve to optimality. For instance, in the North Sea example, we have 25 possible targets and the state space for this dynamic program has approximately 10^{15} elements; this problem is much too large to solve exactly.

Our goal in this paper is to develop tractable heuristic policies for sequential exploration problems and complement these heuristics with upper bounds on the performance of an optimal policy. Our analysis relies heavily on results for "bandit superprocesses," a generalization of the multiarmed bandit problem where the arms are

independent but correspond to Markov *decision* processes, rather than Markov *reward* processes: i.e., in addition to choosing which arm to play, the DM has a choice of how to play the arm. Bandit superprocesses, unlike multiarmed bandits, generally cannot be solved using decomposition and the optimal policies need not have an index structure.

Our model of a sequential exploration problem begins with a set of "targets" that represent Markov decision processes; these are like the arms in a bandit superprocess, but may be dependent. We then partition the set of targets into "clusters," which can be viewed as larger, more complicated arms. Ideally, we want to choose the clusters in a way that captures the most important dependencies within the chosen clusters, while maintaining tractability. Given a set of clusters, we do the following:

- We generate heuristics and lower bounds on the optimal value by assuming the clusters evolve independently, thereby approximating the model with a bandit superprocess. We then further approximate the bandit superprocess with a classical multiarmed bandit by assuming a fixed policy for choosing actions within clusters. The optimal policy for the approximating multiarmed bandit has an index policy that prescribes a feasible heuristic that can be evaluated using Monte Carlo simulation.
- We generate upper bounds by assuming each cluster has perfect information (clairvoyance) about the results for all other clusters. This leads to a series of bandit superprocesses for different possible scenarios. Although we cannot solve these bandit superprocesses exactly, we develop some new results and computational methods for bandit superprocesses that we use to generate an upper bound on the optimal value for each of these scenario-specific bandit superprocesses. (We also consider bounds based on Lagrangian relaxations.) Averaging these bounds across scenarios, we arrive at an upper bound on the performance of an optimal policy for the sequential exploration problem.

Intuitively, we properly capture the learning opportunities within each cluster, but approximate learning across clusters. The heuristic-based lower bounds underestimate values because they do not fully capture cross-cluster learning. In contrast, the clairvoyant upper bounds overestimate values because they include more cross-cluster learning than is actually possible. In the North Sea example, we find that, with a suitable choice of clusters, the difference between the lower and upper bounds is small; this implies that the heuristic policies achieving the lower bound are nearly optimal.

1.1. Literature Review

This paper builds on and contributes to three strands of literature. First, we build on the existing literature on sequential exploration problems. Bickel and Smith (2006) studied sequential exploration problems using dynamic programming methods. Although Bickel and Smith discuss other applications (e.g., to R&D), they focus on an oil exploration example with six targets that are either wet or dry; this leads to a dynamic program with $3^6 = 729$ states. Bickel et al. (2008) considered an oil exploration example with five targets with more complex geologic uncertainties that leads to a dynamic program with approximately 59,000 states. Although these examples could be solved exactly, the North Sea example we consider, with 10^{15} states, is much too large to solve exactly. The North Sea example is adapted from Martinelli et al. (2011) who develop the Bayesian network and discuss probabilistic inference in this example. Martinelli et al. (2012) consider "naive," myopic, and limited-look-ahead heuristic drilling policies in this example; we will compare our heuristics with theirs in §6.5.

Second, we build on the literature on the multiarmed bandit problem. The large literature on this topic is reviewed in the recent book by Gittins et al. (2011). We rely on and extend Whittle's (1980) results for bandit superprocesses in our clairvoyant bounds. Specifically, we show that what we call the "Whittle integral" provides an upper bound on the value of a bandit superprocess. We also describe how we can calculate these Whittle integrals efficiently. Moreover, we compare these bandit superprocess bounds to an alternative approach based on a Lagrangian relaxation of a weakly coupled dynamic program (see, e.g., Whittle 1988, Hawkins 2003, Adelman and Mersereau 2008): we prove that the Whittle integral bounds are tighter than the Lagrangian relaxation bounds in this setting.

The problem of bandits with "correlated arms" is not as well understood as the standard multiarmed bandit problem, though there has been recent work considering particular forms of probabilistic dependence. For example, Wang et al. (2005) explored the use of "side observations" for bandit problems with two arms; their focus is on asymptotic results. Mersereau et al. (2009) derived asymptotic results for a multiarmed bandit model where the mean reward for each arm is a linear function of an unknown scalar; Rusmevichientong and Tsitsiklis (2010) extended these results to the case where the rewards depend linearly on a multivariate random vector. Ryzhov et al. (2012) studied a one-period look-ahead heuristic for multiarmed bandit problems where rewards are correlated with a multivariate normal distribution. Pandey et al. (2007) studied multiarmed bandit models with Bernoulli rewards where arms in the same "cluster" may be correlated; they claim that index policies are optimal in their setting. Our approach also involves grouping arms into clusters; however, these clusters are generally not independent, and, even when they are, index policies may not be optimal.

Finally, we build on the recent literature considering the use of information relaxations to provide performance bounds in dynamic programs. Brown et al. (2010) developed theory and methodology for general dynamic programming models, building on related work by Haugh and Kogan (2004), Andersen and Broadie (2004), Rogers (2002), and others for providing bounds in option pricing problems; Rogers (2007) also developed methods for information-relaxation-based bounds for Markov decision problems.

Brown et al. (2010) considered applications in inventory management and option pricing with stochastic volatility and interest rates. Lai et al. (2010) applied these methods when studying natural gas storage problems. Brown and Smith (2011) considered applications in portfolio management with transaction costs. The clairvoyant bounds in this paper are novel in that we use a new form of partial information relaxation and the solution of the resulting subproblems uses new techniques for bandit superprocesses.

1.2. Outline

We begin by describing the North Sea example in §2; we will use this example to illustrate concepts throughout the paper. We describe our formal model of the sequential exploration problem in §3. We discuss bandits and bandit superprocesses in §4. In §5, we use bandits and bandit superprocesses to generate heuristics and bounds for sequential exploration problems. We present numerical results for the North Sea example in §6. We describe computational methods in Appendix A. Some proofs and detailed discussions are relegated to the online appendix (available as supplemental material at http://dx.doi.org/10.1287/opre.2013.1164).

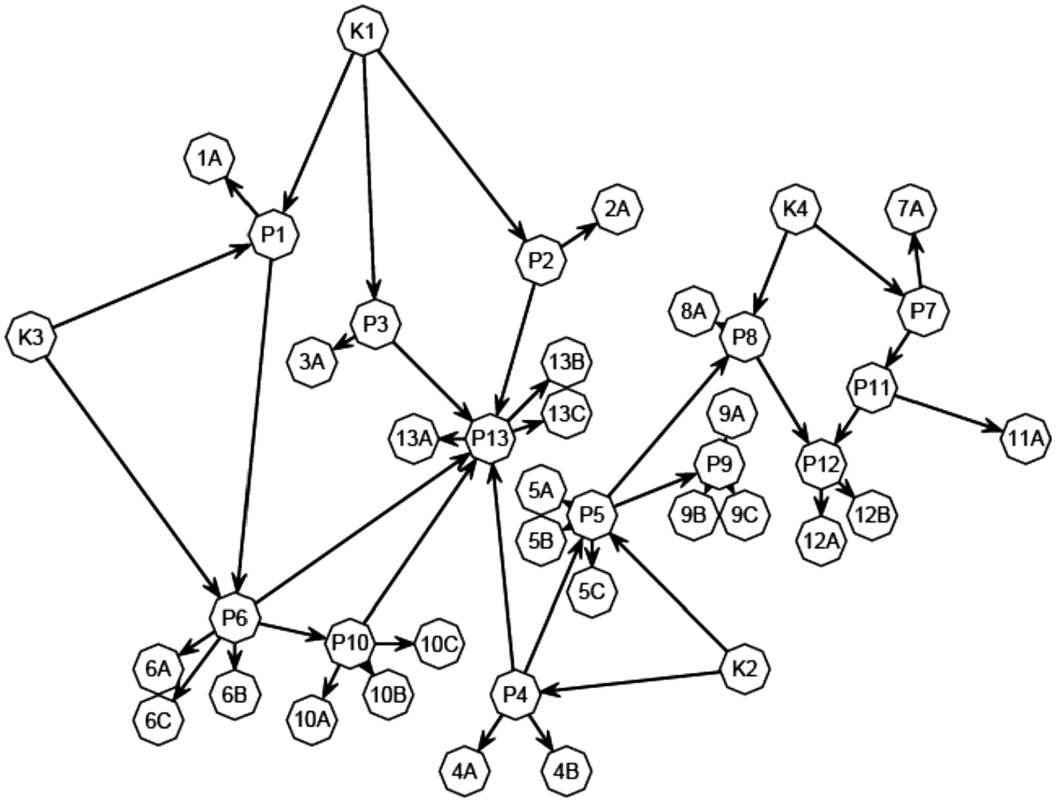
2. The Motivating Example: Oil and Gas Exploration in the North Sea

As mentioned in the introduction, our work on this problem was motivated by the problem of developing an optimal policy for exploring an oil and gas field in the North Sea, off the coast of Norway, in an area referred to as the Tampen region. Martinelli et al. (2011), working with experts at the Norwegian oil company Statoil, developed a Bayesian network model that describes the uncertainty about the generation and migration of hydrocarbons over the geologic history of the field. Figure 1, from Martinelli et al. (2011), provides a high level overview of the model. Nodes in the network represent locations or geological structures in the field and arrows represent possible migration paths between these locations or geological structures.

At the highest level of the network, the nodes with labels beginning with "*K*" represent *kitchens*, which are locations where conditions may have been appropriate to "bake" organic materials into oil or gas. There are three possible states for each kitchen: the kitchens may have produced oil, produced gas, or be dry.

At the next level, the nodes beginning with "*P*" represent *prospects*, which are geologic structures where hydrocarbons may have collected. These prospects may contain oil or gas or be dry. For a prospect to contain oil or gas, at least one of its parents (either a kitchen or another prospect) must contain oil or gas. However, there is uncertainty about whether the oil or gas migrated from the parent to the prospect or was captured at the prospect. Thus, even if the parent structure contains oil or gas, there is some chance that the prospect will be dry. The probabilities for the prospects' states are specified as

Figure 1. Bayesian network model for the example.



conditional probabilities that depend on the state of the parent kitchen(s) or prospect(s).

Finally, at the lowest level of the network, each prospect has associated *targets*, which represent potential drilling sites; kitchens and prospects cannot be drilled. For example, the nodes labeled 6A, 6B, and 6C represent three different targets associated with prospect 6. These targets may also contain oil or gas or be dry. For a target to contain oil or gas, its parent (a prospect) must also contain oil or gas. However, there is also a chance of a "local failure" where a target will not contain oil or gas, even though its parent prospect does.

In total, the model includes four kitchens, 13 prospects, and 25 targets. The probabilities for the model are described in detail in Martinelli et al. (2011). We will follow their assumptions exactly, except we will at times introduce uncertainty about the state of the kitchens. Martinelli et al. assume that the kitchens certainly produced gas, based on observations of results for other nearby fields. Introducing kitchen uncertainty makes the problem more of a "wildcat" play and increases the possibilities for learning about one target from the results at other targets. In the cases where we introduce uncertainty about the state of the kitchens, we assume that at each kitchen there is a 40% chance that the kitchen produced gas, a 40% chance that it produced oil, and a 20% chance that it is dry.

The DM has access to a single drilling rig and in each period must decide which target, if any, to drill. Drilling takes a few months and, after drilling is completed, the target's physical state (oil, gas, or dry) is revealed. The DM then decides whether to continue drilling and which target to drill. This process continues until all of the targets have been drilled or the DM chooses to quit. Drilling costs vary by target and the values vary by target and depend on whether the target contains oil, gas, or is dry. We will use the values and costs as given in Martinelli et al. (2012). We assume a single-period discount factor of 0.98, where the period corresponds to the amount of time it takes to drill a well; this corresponds an annual discount rate of about 10%.

The probabilistic model is implemented using the Bayes Net Toolbox for MATLAB (Murphy 2001). This is an open-source, general-purpose software package that performs probabilistic inference using Bayesian network techniques. We use this software to calculate, for example, the probability of finding oil or gas at one target (or a set of targets) after observing results for other targets.

3. The Sequential Exploration Problem

In this section, we formally describe the sequential exploration problem and then illustrate it with the North Sea example.

3.1. The Basic Model

The model consists of a set of targets organized into N clusters. The uncertainty in the model is described by a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ where the outcomes $\boldsymbol{\omega}$ in Ω are decomposed into a vector of cluster-specific outcomes $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_N)$ and ω_i is drawn from Ω_i. The σ-algebra of events $\mathcal{F}$ for Ω is the product algebra $\mathcal{F}_1 \otimes \cdots \otimes \mathcal{F}_N$ of σ-algebras $\mathcal{F}_i$ of events for Ω_i. The probability distribution (or measure) $\mathbb{P}$ need not decompose, however, as $\mathcal{F}_1, \ldots, \mathcal{F}_N$ need not be independent.

The cluster outcomes ω_i are not directly observed and the cluster state x_i summarizes what the DM knows about cluster i at any given time. These cluster states x_i range over a finite set X_i and the system state $\mathbf{x} = (x_1, \ldots, x_N)$ ranges over the product of the cluster state spaces, $\prod_{i=1}^{N} X_i$. The system begins in an initial state $\mathbf{x}^\circ = (x_1^\circ, \ldots, x_N^\circ)$ with a joint probability distribution $P(\boldsymbol{\omega} \mid \mathbf{x}^\circ)$ on Ω corresponding to $\mathbb{P}$. We write the corresponding marginal distribution for cluster i outcomes Ω_i as $P(\omega_i \mid \mathbf{x}^\circ)$.

In each period, the DM can quit or select a cluster to work on and an action for that cluster. The possible actions a_i for cluster i are selected from a finite set of possible actions $A_i(x_i)$ that depends on the current state for the cluster. If the DM works on cluster i, the state of that cluster transitions to a next-period state $\tilde{x}_i(x_i, a_i, \omega_i)$ that is a function of the cluster's current state x_i, the action selected a_i, and the cluster outcome ω_i; the uncertainty in the state transition is due to uncertainty about the cluster outcome ω_i. The states for the inactive clusters do not change and thus the next-period system state is $\tilde{\mathbf{x}}(\mathbf{x}, a_i, \boldsymbol{\omega}) = (x_1, \ldots, x_{i-1}, \tilde{x}_i(x_i, a_i, \omega_i), x_{i+1}, \ldots, x_N)$. As actions are taken, the system state $\mathbf{x}$ evolves and the joint distribution over outcomes, $P(\boldsymbol{\omega} \mid \mathbf{x})$, is updated using Bayes' rule to reflect the information observed up to that time. We assume that the system state $\mathbf{x}$ is a sufficient statistic in that the conditional probability distribution on Ω given the current $\mathbf{x}$ does not depend on earlier system states.

If the DM works on cluster i, the DM receives a reward $\tilde{r}_i(x_i, a_i, \omega_i)$ that depends on the current state x_i for that cluster, the action selected a_i, and the cluster i outcome ω_i. The goal is to sequentially choose clusters to work on and associated actions to maximize the expected discounted reward. We will assume that the discount factor δ satisfies $0 \leqslant \delta < 1$. Formulating this as a stochastic dynamic program, we can write the value function $V(\mathbf{x})$ for the sequential exploration problem recursively as

$$V(\mathbf{x}) \equiv \max\Big\{0, \max_i \max_{a_i \in A_i(x_i)} \mathbb{E}[\tilde{r}_i(x_i, a_i) + \delta V(\tilde{\mathbf{x}}(\mathbf{x}, a_i)) \mid \mathbf{x}]\Big\}, \qquad (1)$$

where, to streamline our notation, we suppress $\boldsymbol{\omega}$ and let $\tilde{r}_i(x_i, a_i)$ and $\tilde{\mathbf{x}}(\mathbf{x}, a_i)$ denote the random reward and next-period state. The expectation $\mathbb{E}[\tilde{r}_i(x_i, a_i) + \delta V(\tilde{\mathbf{x}}(\mathbf{x}, a_i)) \mid \mathbf{x}]$ in (1) may be written explicitly as

$$\int_{\omega_i \in \Omega_i} (\tilde{r}_i(x_i, a_i, \omega_i) + \delta V((x_1, \ldots, x_{i-1}, \tilde{x}_i(x_i, a_i, \omega_i), x_{i+1}, \ldots, x_N))) \, \mathrm{d}P(\omega_i \mid \mathbf{x}).$$

To ensure that these expectations are well defined, we assume the rewards $\tilde{r}_i(x_i, a_i, \omega_i)$ and state transitions

$\tilde{x}_i(x_i, a_i, \omega_i)$ are $\mathcal{F}_i$-measurable functions of ω_i for each cluster state x_i and action a_i.

In applications, there is significant flexibility in the definition of clusters, as one can always combine any two clusters (or targets) into a more complicated cluster by capturing the choice between the two clusters (the maximization over i in (1)) within the choice among actions (the maximization over a_i in (1)) for the combined cluster. As long as we define outcomes, states, rewards, actions sets, and transition probabilities appropriately, combining clusters in this way has no effect on the values and optimal policies for the full model. However, the cluster definitions will play a key role in the approximations of the model that we will use to generate heuristics and bounds. We will discuss the choice of clusters in more detail in §5.3.

3.2. Modeling the North Sea Example

We consider four different definitions of clusters in the North Sea example. In the first case, we consider 25 clusters, each consisting of a single target. There are three possible outcomes ω_i for each target: the target could contain oil, contain gas, or be dry. There are four possible states x_i for each target: the target could be (i) drilled and known to contain oil, (ii) drilled and known to contain gas, (iii) drilled and known to be dry, or (iv) undrilled. Initially, all 25 targets are undrilled. If target i is undrilled, the action set $A_i(x_i)$ contains a single action, representing the possibility of drilling; if target i has already been drilled, the action set $A_i(x_i)$ is empty. Drilling a target leads the state of the target to transition from undrilled to one of the three drilled states and pays a reward, with the next state and reward depending on the outcome ω_i, i.e., whether the target contains oil, contains gas, or is dry.

For the second case of clusters, we consider 13 clusters where each cluster contains all of the targets associated with a given prospect. For example, one cluster consists of the three targets associated with prospect 13; the outcome ω_i ranges over the 3^3 possible outcomes for the three targets in the cluster and the cluster state x_i ranges over the 4^3 different possible states for the targets. The actions associated with these clusters represent drilling at one of the targets contained in the cluster.

We also consider larger clusters with targets grouped as shown in Figures 2 and 3. Clusters 3 and 5 in Figure 2 both have six targets. They therefore have $3^6 = 729$ outcomes and $4^6 = 4{,}096$ states; the actions correspond to drilling one of the undrilled targets in the cluster. The largest cluster we will consider is cluster 3 in Figure 3; this cluster has nine targets, $3^9 = 19{,}683$ outcomes, and $4^9 = 262{,}144$ states.[1]

In the North Sea example, the transition probabilities for the next-period state $\tilde{\mathbf{x}}(\mathbf{x}, a_i)$ can be calculated from the Bayesian network for any state $\mathbf{x}$ and action a_i. For example, if we start in the state with all targets undrilled and then choose an action a_i that corresponds to drilling at, say, target 9A, the probabilities of transitioning to the states

Figure 2. Example with medium clusters.

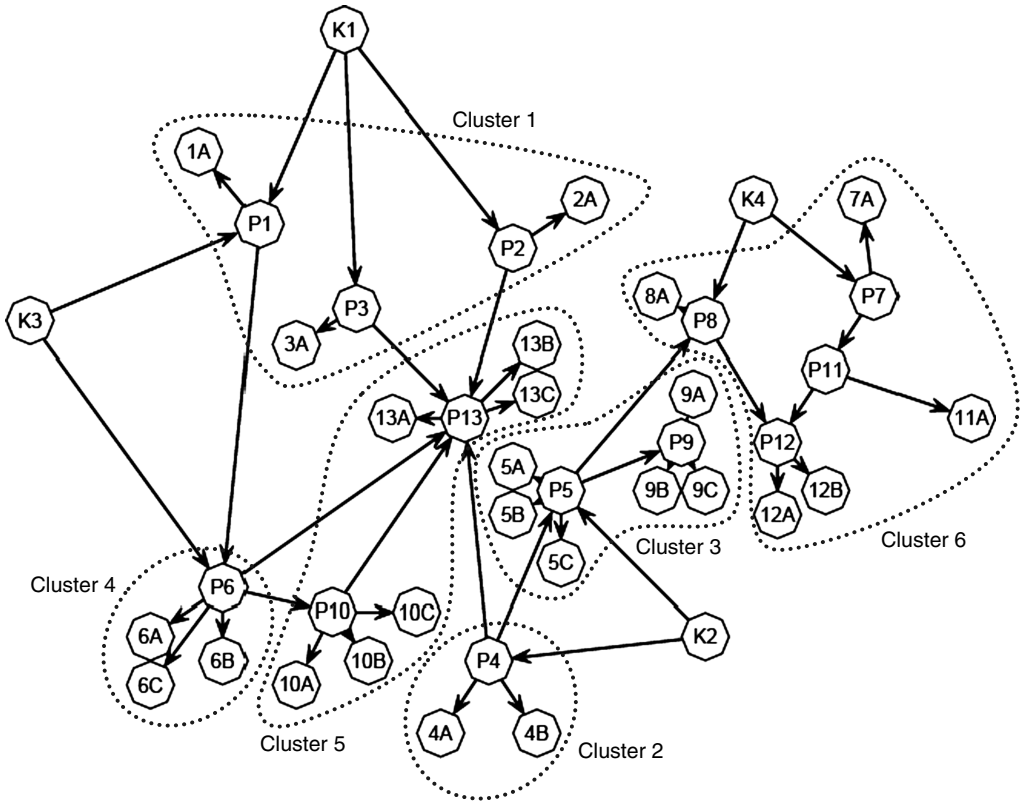
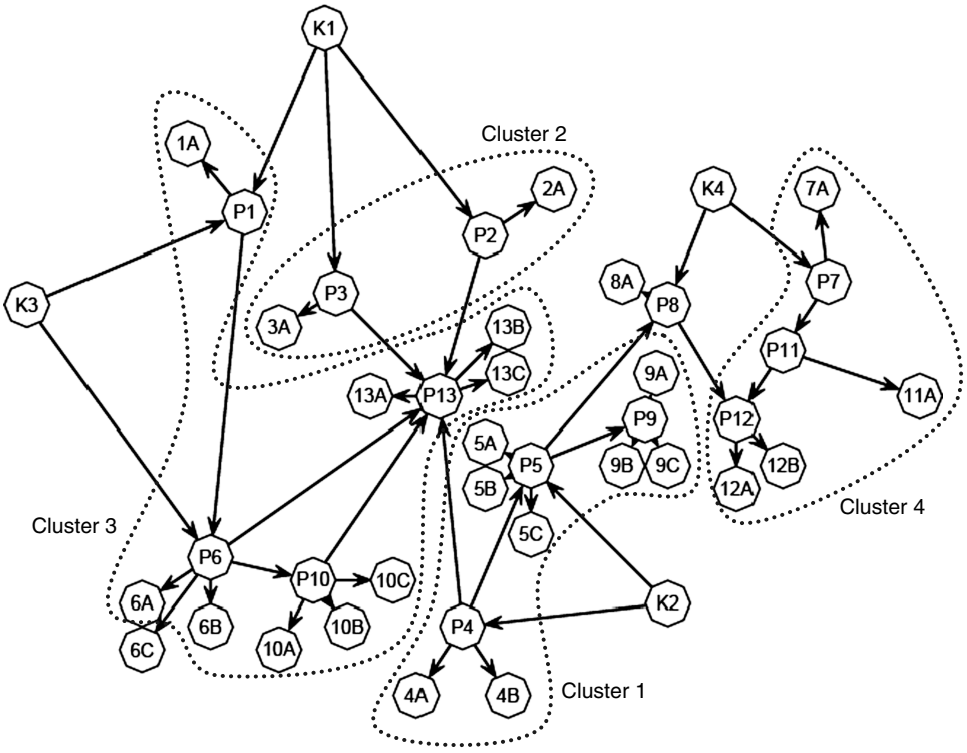


Figure 3. Example with large clusters.



where the target is known to contain oil or gas or to be dry are given by the marginal probabilities associated with that node in the Bayesian network. If we then start in a state where target 9A is known to be dry and all other targets are undrilled, we can calculate transition probabilities associated with drilling other targets by updating the Bayesian network to reflect the fact that target 9A is known to be dry and then calculating new marginal probabilities for other targets. These calculations can be rather involved, but the algorithms used in the Bayes Net Toolbox (Murphy 2001) take advantage of the network structure to perform these calculations reasonably efficiently.

As discussed in the introduction, these sequential exploration problems suffer the curse of dimensionality and scale poorly. In this example, regardless of how the targets are grouped into clusters, there are approximately 10^{15} different states to consider, with up to 25 different actions to evaluate in each state and each state-action combination may have a different set of transition probabilities. This model is much too large to solve exactly, so we will instead consider heuristic policies and performance bounds.

4. Multiarmed Bandits and Bandit Superprocesses

In the special case where the clusters are probabilistically independent, the sequential exploration problem (1) is known as a bandit superprocess. In this section, we will study properties of bandit superprocesses. Although these results do not apply directly with dependent clusters, these results provide key tools for our study of sequential exploration problems. We return to the full problem in §5 and describe how we will use these tools to generate heuristics and bounds in a setting with dependence among clusters. In this section, we first discuss some basic properties of bandit superprocesses, then discuss the Whittle integral for generating bounds on value functions for bandit superprocesses and consider some examples. We then compare the Whittle integral bounds to bounds provided by a Lagrangian relaxation.

4.1. Definitions and Basic Results

Throughout our discussion of bandit superprocesses, we will assume that the clusters are independent in that the initial underlying probability distribution factors as $P(\boldsymbol{\omega} \mid \mathbf{x}^\circ) = \prod_{i=1}^{N} P(\omega_i \mid \mathbf{x}^\circ)$. This independence is preserved as the system state evolves and the probabilities for one cluster will always be independent of the state of other clusters (i.e., $P(\omega_i \mid \mathbf{x}) = P(\omega_i \mid x_i)$). Thus, we can consider transition probabilities and conditional expectations conditioned on the active state x_i rather than the full system state $\mathbf{x}$.

When studying bandit superprocesses, it is useful to follow Whittle (1980) and consider a variation on the problem where quitting results in a payment of M, rather than zero as assumed in (1). Given a retirement value of M, we can define the system-wide value function Φ as

$$\Phi(\mathbf{x}, M) \equiv \max\Big\{M, \max_i \max_{a_i \in A_i(x_i)} \mathbb{E}[\tilde{r}_i(x_i, a_i) + \delta\Phi(\tilde{\mathbf{x}}(\mathbf{x}, a_i), M) \mid x_i]\Big\}. \quad (2)$$

The ability to choose an action a_i for a cluster distinguishes bandit superprocesses from the multiarmed bandit problem: in the multiarmed bandit problem, $A_i(x_i)$ contains at most one element and the only question is which cluster to work on (or arm to play).

We can define a cluster-specific value function that considers rewards and actions for cluster i only as follows:

$$\phi_i(x_i, M) \equiv \max\Big\{M, \max_{a_i \in A_i(x_i)} \mathbb{E}[\tilde{r}_i(x_i, a_i) + \delta\phi_i(\tilde{x}_i(x_i, a_i), M) \mid x_i]\Big\}. \quad (3)$$

The *Gittins index* $M_i^*(x_i)$ for cluster i in state x_i is the smallest M such that $\phi_i(x_i, M) = M$, i.e., the retirement value that makes the DM indifferent between retiring and continuing from this state. With discounting ($\delta < 1$) and bounded rewards, we know that the Gittins index $M_i^*(x_i)$ will exist and be less than $r_i^*/(1-\delta)$, where r_i^* is the maximum possible reward for cluster i. We let $M^*(\mathbf{x})$ denote the maximum Gittins index for a given system state, i.e., $M^*(\mathbf{x}) = \max_i M_i^*(x_i)$.

We note the following basic properties of these value functions and their derivatives, $\Phi'(\mathbf{x}, M) \equiv \partial\Phi(\mathbf{x}, M)/\partial M$ and $\phi_i'(\mathbf{x}, M) \equiv \partial\phi_i(\mathbf{x}, M)/\partial M$, that will be used in our analysis.

Lemma 4.1 (Properties of Value Functions). *For any* $\mathbf{x}$,

(i) $\Phi(\mathbf{x}, M)$ *is piecewise linear, nondecreasing, and convex in* M;

(ii) $\Phi'(\mathbf{x}, M)$ *exists for almost all* M, *and where it exists,*

(a) $\Phi'(\mathbf{x}, M)$ *is piecewise constant, nonnegative, and nondecreasing in* M;

(b) $\Phi'(\mathbf{x}, M) < 1$ *for* $M < M^*(\mathbf{x})$; *and*

(c) $\Phi'(\mathbf{x}, M) = \mathbb{E}[\delta^{\tilde{\tau}(\mathbf{x}, M)} \mid \mathbf{x}]$, *where* $\tilde{\tau}(\mathbf{x}, M)$ *is the* (*random*) *time of retirement when following a policy that is optimal with retirement value* M *and starting in state* $\mathbf{x}$;

(iii) *If it is optimal to retire in state* $\mathbf{x}$ *with retirement value* M, *it is also optimal to retire in state* $\mathbf{x}$ *with any retirement value* $M' \geq M$.

These results also hold for all cluster-specific value functions $\phi_i(x_i, M)$ *and their derivatives* $\phi_i'(x_i, M)$ *with* $M_i^*(x_i)$ *in place of* $M^*(\mathbf{x})$ *in* (ii)(b). *In addition, we have* $\phi_i'(x_i, M) = 1$ *for* $M > M_i^*(x_i)$.

These results follow from standard arguments for multiarmed bandits (see, e.g., Whittle 1980 or Bertsekas 1995) that also apply with bandit superprocesses. For completeness, we provide a proof in the online appendix.

4.2. The Whittle Integral and Bound

We now show how we can provide an upper bound on the system-wide value function $\Phi(\mathbf{x}, M)$ in terms of the cluster-specific value functions $\phi_i(x_i, M)$. From the fundamental theorem of calculus, for any B, we can write

$$\Phi(\mathbf{x}, M) = \Phi(\mathbf{x}, B) - \int_{m=M}^{B} \Phi'(\mathbf{x}, m)\,\mathrm{d}m. \quad (4)$$

If we choose B to be large enough so that, if the DM were offered retirement value B, it would be optimal to retire immediately, we would then have $\Phi(\mathbf{x}, B) = B$ and can simplify (4) accordingly. We then approximate the derivative $\Phi'(\mathbf{x}, m)$ with the product of cluster-specific value function derivatives $\prod_{i=1}^N \phi_i'(x_i, m)$; this leads to an upper bound on the system-wide value function.

Proposition 4.2 (The Whittle Integral and Bound). *For any* $\mathbf{x}$ *and any* B *greater than or equal to the maximum Gittins index* $M^*(\mathbf{x})$, *the Whittle integral,*

$$\hat{\Phi}(\mathbf{x}, M) \equiv B - \int_{m=M}^{B} \prod_{i=1}^{N} \phi_i'(x_i, m)\,\mathrm{d}m, \quad (5)$$

provides an upper bound on the system-wide value function: $\hat{\Phi}(\mathbf{x}, M) \geq \Phi(\mathbf{x}, M)$.

This proposition is essentially an unnoticed or unappreciated intermediate result in Whittle's (1980) proof of the optimality of the Gittins index-based policies for a particular form of bandit superprocesses. We will discuss Whittle's result immediately after the proof and then provide some intuition for this approximation of the derivative $\Phi'(\mathbf{x}, m)$. A few remarks on the result before we provide the proof:

(i) As long as B is greater than or equal to the maximum Gittins index $M^*(\mathbf{x})$, the value of B does not affect the value of $\hat{\Phi}(\mathbf{x}, M)$ because, as noted in Lemma 4.1, $\phi_i'(x_i, M) = 1$ for $M > M_i^*(x_i)$.

(ii) The Whittle integral $\hat{\Phi}(\mathbf{x}, M)$, like $\Phi(\mathbf{x}, M)$, is piecewise linear, nondecreasing, and convex in M. This follows from the fact that the $\phi_i'(x_i, m)$ are piecewise constant, positive, and nondecreasing in m.

(iii) Proposition 4.2 implies that it is optimal to retire for any M such that $M \geq M^*(\mathbf{x})$: for these values of M, $\hat{\Phi}(\mathbf{x}, M) = M$. Since $\hat{\Phi}(\mathbf{x}, M)$ is an upper bound on the true value function $\Phi(\mathbf{x}, M)$, it follows that immediate retirement must be optimal. This, in turn, implies that $\Phi'(\mathbf{x}, M) = 1$ for $M > M^*(\mathbf{x})$.

Proof. Let $Q_i(\mathbf{x}, M) \equiv \prod_{j \neq i} \phi_j'(x_j, M)$. Note that, because the clusters are independent, $Q_i(\mathbf{x}, M)$ does not depend on x_i. Also note that, from Lemma 4.1, $Q_i(\mathbf{x}, M)$ is nonnegative and nondecreasing in M and equal to one when $M > M_{(i)}^*$, where $M_{(i)}^* \equiv \max_{j \neq i} M_j^*(x_j)$. Thus, for fixed $\mathbf{x}$, $Q_i(\mathbf{x}, M)$ can be viewed as a cumulative probability function in M with the probability measure's mass concentrated below $M_{(i)}^*$, which is less than or equal to $M^*(\mathbf{x})$. Using this and integrating by parts, for any cluster i, we can represent $\hat{\Phi}(\mathbf{x}, M)$ as

$$\hat{\Phi}(\mathbf{x}, M) = \phi_i(x_i, M) Q_i(\mathbf{x}, M) + \int_{m=M}^{M^*(\mathbf{x})} \phi_i(x_i, m)\,\mathrm{d}_m Q_i(\mathbf{x}, m), \quad (6)$$

where $M^*(\mathbf{x}) \equiv \max_i M_i^*(\mathbf{x})$. (We also use the fact that $\phi_i(x_i, B) = B$ for $B \geq M^*$ here.) Thus $\hat{\Phi}(\mathbf{x}, M)$ can be interpreted as the expected value of $\phi_i(x_i, m)$ with random m

having mass $Q_i(\mathbf{x}, M)$ at M and the rest of the mass distributed between M and $M^*(\mathbf{x})$, as specified by the probability distribution $Q_i(\mathbf{x}, m)$.

Since $\phi_i(x_i, M) \geqslant M$ for all M and any cluster i, interpreting (6) as an expectation, we see

$$\hat{\Phi}(\mathbf{x}, M) \geqslant M. \tag{7}$$

Similarly, since $\phi_i(x_i, m)$ is greater than or equal to the value associated with choosing action a_i (i.e., $\phi_i(x_i, m) \geqslant \mathbb{E}[\tilde{r}_i(x_i, a_i) + \delta\phi_i(\tilde{x}_i(x_i, a_i), M) \mid x_i]$, from Equation (3)), Equation (6) also implies

$$\begin{aligned}
&\hat{\Phi}(\mathbf{x}, M)\\
&= \phi_i(x_i, M)Q_i(\mathbf{x}, M) + \int_{m=M}^{M^*(\mathbf{x})} \phi_i(x_i, m)\, \mathrm{d}_m Q_i(\mathbf{x}, m)\\
&\geqslant \mathbb{E}[\tilde{r}_i(x_i, a_i) + \delta\phi_i(\tilde{x}_i(x_i, a_i), M) \mid x_i] Q_i(\mathbf{x}, M)\\
&\quad + \int_{m=M}^{M^*(\mathbf{x})} \mathbb{E}[\tilde{r}_i(x_i, a_i) + \delta\phi_i(\tilde{x}_i(x_i, a_i), m) \mid x_i]\, \mathrm{d}_m Q_i(\mathbf{x}, m)\\
&= \mathbb{E}[\tilde{r}_i(x_i, a_i) \mid x_i] + \mathbb{E}\Bigg[\delta(\phi_i(\tilde{x}_i(x_i, a_i), M)Q_i(\mathbf{x}, M)\\
&\qquad\qquad + \int_{m=M}^{M^*(\mathbf{x})} \phi_i(\tilde{x}_i(x_i, a_i), m)\, \mathrm{d}_m Q_i(\mathbf{x}, m)) \,\Big|\, x_i\Bigg]\\
&= \mathbb{E}[\tilde{r}_i(x_i, a_i) + \delta\hat{\Phi}(\tilde{\mathbf{x}}(\mathbf{x}, a_i), M) \mid x_i].
\end{aligned} \tag{8}$$

To see this last equality, recall that $Q_i(\mathbf{x}, m)$ does not depend on x_i (because the clusters are independent) and that only x_i will change when cluster i is active with action a_i selected; thus $Q_i(\tilde{\mathbf{x}}(\mathbf{x}, a_i), m) = Q_i(\mathbf{x}, m)$. Then, from (6), we have

$$\begin{aligned}
&\hat{\Phi}(\tilde{\mathbf{x}}(\mathbf{x}, a_i), M)\\
&\quad = \phi_i(\tilde{x}_i(x_i, a_i), M)Q_i(\tilde{\mathbf{x}}(\mathbf{x}, a_i), M)\\
&\qquad + \int_{m=M}^{M^*(\mathbf{x})} \phi_i(\tilde{x}_i(x_i, a_i), m)\, \mathrm{d}_m Q_i(\tilde{\mathbf{x}}(\mathbf{x}, a_i), m)\\
&\quad = \phi_i(\tilde{x}_i(x_i, a_i), M)Q_i(\mathbf{x}, M)\\
&\qquad + \int_{m=M}^{M^*(\mathbf{x})} \phi_i(\tilde{x}_i(x_i, a_i), m)\, \mathrm{d}_m Q_i(\mathbf{x}, m),
\end{aligned}$$

which establishes the last equality above.

Combining (7) and (8), we have

$$\begin{aligned}
\hat{\Phi}(\mathbf{x}, M) \geqslant \max\Big\{ M, \max_i \max_{a_i \in A_i(x_i)} &\mathbb{E}[\tilde{r}_i(x_i, a_i)\\
&+ \delta\hat{\Phi}(\tilde{\mathbf{x}}(\mathbf{x}, a_i), M) \mid x_i]\Big\}.
\end{aligned} \tag{9}$$

Thus $\hat{\Phi}(\mathbf{x}, M)$ satisfies a recursion analogous to Equation (2) defining $\Phi(\mathbf{x}, M)$, but with an inequality replacing the equality. This is sufficient to ensure that $\hat{\Phi}(\mathbf{x}, M) \geqslant \Phi(\mathbf{x}, M)$; see, e.g., Puterman (1994), Theorem 6.2.2. □

As mentioned earlier, the proof above is essentially due to Whittle (1980), though he did not consider the use of the Whittle integrals to provide bounds on the value function for bandit superprocesses. His main goal in that paper was to provide a simple dynamic programming-based proof of the optimality of Gittins-index-based policies for the multiarmed bandit problem. However, he also observed that the index policy result holds for bandit superprocesses satisfying a particular property. We say cluster i satisfies the *Whittle condition* in state x_i if there is a *dominant action* a_i in $A_i(x_i)$ that achieves the maximum in (3) for all values of M such that $0 \leqslant M \leqslant M_i^*(x_i)$ or if quitting is optimal for all $M \geqslant 0$. Whittle's (1980) result for bandit superprocesses can then be stated as follows.

Proposition 4.3 (Whittle 1980). *If the Whittle condition holds for all clusters, in all states, then, for any retirement value $M \geqslant 0$,*

(i) *the Whittle integral is exactly equal to the value function, i.e., $\hat{\Phi}(\mathbf{x}, M) = \Phi(\mathbf{x}, M)$;*

(ii) *for any state $\mathbf{x}$, if the retirement value M exceeds the maximum Gittins index $M^*(\mathbf{x})$, it is optimal to retire. Otherwise, it is optimal to work on the cluster that achieves the maximum Gittins index in state $\mathbf{x}$ and to choose the dominant action for that cluster in the current state.*

Proof. Given a system state $\mathbf{x}$, if the Whittle condition is satisfied by the cluster with the maximum Gittins index and its dominant action is a_i, we then have $\phi_i(x_i, m) = \mathbb{E}[\tilde{r}_i(x_i, a_i) + \delta\phi_i(\tilde{x}_i(x_i, a_i), m) \mid x_i]$ for all m such that $0 \leqslant m \leqslant M^*(\mathbf{x})$. This implies that the inequality (8) holds with equality. If the dominant action in state $\mathbf{x}$ is to quit, then (7) holds with equality. If the Whittle condition holds for all states and clusters, (9) holds with equality in all states and, consequently, $\hat{\Phi}(\mathbf{x}, M) = \Phi(\mathbf{x}, M)$ (see, e.g., Puterman 1994, Theorem 6.2.2.). In this case, the optimal policy is to work on the cluster with the largest Gittins index and to choose the dominant action for that cluster and state, as these are the actions that achieve equality in (8). □

A classic multiarmed bandit has a single choice of action in each state and thus automatically satisfies the Whittle condition. More generally, for a bandit superprocess, if the Whittle condition holds in all states for all clusters, the optimal choice of action for a cluster is unambiguous and we can reduce the bandit superprocess to a multiarmed bandit problem by assuming the choice of the dominant action in each state: that is, without loss of optimality, we can replace $A_i(x_i)$ with a singleton containing just the dominant action for state x_i. Glazebrook (1982) showed that this Whittle condition is necessary (in a sense that he makes precise) for bandit superprocesses to have optimal policies with an index structure. As we will discuss in the example in §4.3, if the Whittle condition is not satisfied, the choice of action a_i for a cluster may depend on the state of the other clusters.

This intuition helps explain why we take the derivative $\Phi'(\mathbf{x}, m)$ of the system-wide value function to be the product of derivatives for the cluster-specific value functions

$\prod_{i=1}^{N} \phi_i'(x_i, m)$ in the Whittle integral (5). If the Whittle condition is satisfied and the optimal actions do not depend on the other clusters, one might conjecture that, for a given retirement value M and initial state $\mathbf{x}$, the time until retirement for the full system $\tilde{\tau}(\mathbf{x}, M)$ would be equal to the sum of the times until retirement $\tilde{\tau}_i(x_i, M)$ for the individual clusters when considered in isolation. Recalling from Lemma 4.1 that the derivatives of the value functions $\Phi'(\mathbf{x}, M)$ and $\phi_i'(x_i, M)$ are equal to the expected discount factor at retirement, $\mathbb{E}[\delta^{\tilde{\tau}(\mathbf{x},M)} \mid \mathbf{x}]$ and $\mathbb{E}[\delta^{\tilde{\tau}_i(x_i,M)} \mid x_i]$ (respectively), if this conjecture is true, we have

$$\begin{aligned}\Phi'(\mathbf{x}, M) &= \mathbb{E}[\delta^{\tilde{\tau}(\mathbf{x},M)} \mid \mathbf{x}] = \mathbb{E}[\delta^{\sum_{i=1}^{N} \tilde{\tau}_i(x_i,M)} \mid \mathbf{x}] \\ &= \mathbb{E}\left[\prod_{i=1}^{N} \delta^{\tilde{\tau}_i(x_i,M)} \,\middle|\, \mathbf{x}\right] = \prod_{i=1}^{N} \mathbb{E}[\delta^{\tilde{\tau}_i(x_i,M)} \mid x_i] \qquad (10) \\ &= \prod_{i=1}^{N} \phi_i'(x_i, M) = \hat{\Phi}'(\mathbf{x}, M).\end{aligned}$$

Whittle's result (Proposition 4.3) proves that this conjecture is true. (The fourth equality above follows from the fact that the clusters are independent.)

However, if there is no dominant action for some cluster in some states, the state of other clusters may affect the choice of actions at this cluster and the retirement time for the full system $\tilde{\tau}$ may no longer be equal to the sum of the retirement times $\tilde{\tau}_i$ for the clusters viewed in isolation; $\tilde{\tau}$ may be more or less than the sum of the cluster-specific retirement times $\tilde{\tau}_i$. Thus the second equality in (10) need not hold. Therefore the Whittle integral need not provide an exact value function for a bandit superprocess if the Whittle condition is not satisfied in all states for all clusters, but, as shown in Proposition 4.2, it does provide an upper bound.

4.3. Examples

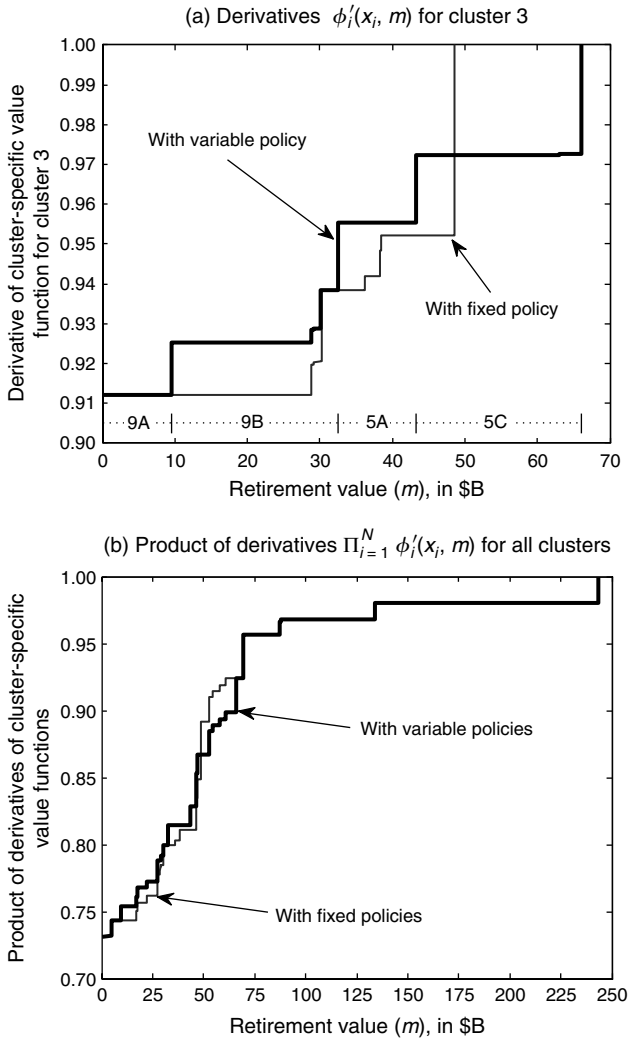
To illustrate these ideas, let us consider Whittle integrals for the North Sea example with clusters as defined in Figure 2, but with the clusters assumed to be independent so the model is a bandit superprocess. Specifically, we will set the probabilities for each cluster to be the marginal distribution for the cluster, i.e., the initial joint distribution $\mathrm{P}(\boldsymbol{\omega} \mid \mathbf{x}^\circ)$ on outcomes is replaced by $\prod_{i=1}^{N} P(\omega_i \mid \mathbf{x}^\circ)$; we focus on the case where there is uncertainty about the state of the kitchens.

We will consider two different sets of possible actions $A_i(x_i)$ for the cluster-specific subproblems (3) underlying the Whittle integral. First, we consider the case where the DM has full flexibility in choosing actions in each state, for all values of m. Second, to illustrate the role of the Whittle condition and the potential slack in using the Whittle integral as a bound, we consider the case where the actions $A_i(x_i)$ for each cluster are restricted to a singleton with action a_i selected by a fixed policy; here we take the fixed policies to be the policies that solve the cluster-specific dynamic program (3) with $M = 0$. Note that the retirement decisions are not fixed by this policy; only the actions if the DM continues are fixed. Fixing the policies in this way reduces the bandit superprocess to a multiarmed bandit problem. (We will use such bandits with fixed policies when considering heuristics for the sequential exploration problem in §5.1.)

Figure 4(a) shows the derivatives $\phi_i'(x_i, m)$ as a function of the retirement value m retirement value m for cluster 3 in Figure 2 with x_i corresponding with all targets undrilled. The thick black line shows $\phi_i'(x_i, m)$ with full flexibility in the choice of actions and the thin blue line shows the derivatives in the case with the fixed policy for choosing actions. These derivatives were calculated using the frontier algorithm described in Appendix A and are piecewise constant and increasing, as noted in Lemma 4.1.

If cluster 3 satisfied the Whittle condition in all states, the derivatives with fixed and variable policies would be identical, as the actions selected given retirement value $M = 0$ would be optimal for larger m. Here, the optimal choice of actions for cluster 3 varies with m. For instance,

Figure 4. Example results for clusters in Figure 2.



for values of m less than \$9.5B, the optimal policy calls for drilling at target 9A first. Target 9A is not very attractive in itself (in fact, it has a negative expected value if taken in isolation), but it is relatively inexpensive to drill and provides a great deal of information about other targets in the cluster. For values of m greater than \$66B, it is optimal to retire and not work on this cluster at all. For values of m between \$43B and \$66B, it is optimal to drill at target 5C first; 5C has the largest unconditional expected value, but provides less information about other targets. With a large value of m, the additional information provided by drilling at 9A is not as valuable as the DM would rather retire than drill at many of the prospects in the cluster. Interestingly, the optimal initial target changes several times for different values of m, switching from 9A to 9B to 5A to 5C as we increase m; the switch points are noted at the bottom of Figure 4(a).

Because the choice of first action varies in the cluster-specific problem with the retirement value m, the best choice for this cluster in the full system problem may depend on the states of other clusters. When solving the full system problem, the DM does not actually "retire" from a cluster and may return to work on the cluster after working elsewhere. The optimal choice of first action for the cluster in the full system problem may depend on how long the DM expects it to take before returning to work on the cluster. If the DM expects to return quickly, it may be optimal to invest by drilling at sites that provide a great deal of information, as suggested by the cluster policy with $m=0$ or other small values of m. On the other hand, if the DM does not expect to return for a long time (or never), it may be optimal to not make such an investment, as suggested by the policies given by larger values of m.

Figure 4(b) shows the product of derivatives, $\prod_{i=1}^{N}\phi_i'(x_i,m)$, for all clusters in Figure 2. The Whittle integral given zero retirement value, $\hat{\Phi}(x_i,0)$, is equal to the area above $\prod_{i=1}^{N}\phi_i'(x_i,m)$ and inside the bounding box. In the case where the DM has full flexibility in the choice of policies for all clusters (the thick black line), the Whittle integral value is \$17,294M; this provides an upper bound on the value of the bandit superprocess. In the case with fixed policies for the clusters, the Whittle integral is \$17,261M; this provides an exact value for the bandit problem with these fixed policies. As the solution for the bandit problem with this fixed policy is feasible for the bandit superprocess with variable policies, this fixed-policy value provides a lower bound on the value with an optimal policy for the bandit superprocess. The difference between the upper and lower bound, \$17,294M − \$17,261M = \$33M, is an upper bound on the slack in the Whittle integral: we know that the value associated with the optimal policy is within \$33M or 0.19% of the upper bound given by the Whittle integral with variable policies.

The slack in the Whittle integral bounds will vary with the clusters and distributions assumed. In the case where clusters correspond to individual targets, there is at most a single action to choose from and the Whittle condition is satisfied; thus, the Whittle integral is equal to the value function in this case. In the other cases, the Whittle condition is not satisfied. Performing a similar analysis as in the example above for the other clusters with kitchen uncertainty, we find upper bounds on slack of \$8M, \$33M (as above), \$31M (0.05%, 0.19%, 0.17%) for the cases with clusters corresponding to prospects and as in Figures 2 and 3 (respectively). Without kitchen uncertainty, these bounds on the slack are \$6M, \$68M, and \$68M (0.03%, 0.29%, 0.29%). Thus, in these examples, the Whittle integral provides a fairly tight bound on the optimal value, even though the Whittle condition is not satisfied.

4.4. Bounds Based on Lagrangian Relaxations

Rather than using the Whittle integral to bound the value function for the bandit superprocess, another approach that is commonly used in similar settings is a Lagrangian relaxation bound. This approach was used in Whittle (1988) in a study of "restless bandits" and is developed more fully in Hawkins (2003) and Adelman and Mersereau (2008), which both consider applications in bandit or bandit-like problems. Farias and Madan (2011) used this approach in a study of "irrevocable bandit problems." It is natural to consider using a Lagrangian relaxation with bandit superprocesses.

In this approach, we introduce Lagrange multipliers for the constraint that we can work on at most one cluster at a time or, equivalently, the constraint that we must rest at least $N-1$ clusters in each period. If we assume the Lagrange multipliers are constant across states, we can interpret the Lagrange multiplier as a retirement value M and the Lagrangian decomposes into the sum of cluster-specific value functions:

$$\hat{L}(\mathbf{x},M)=\sum_{i=1}^{N}\phi_i(x_i,M)-M(N-1). \tag{11}$$

We provide the details of this derivation in the online appendix. This Lagrangian $\hat{L}(\mathbf{x},M)$ provides an upper bound on the system-wide value function with zero retirement value, $\Phi(\mathbf{x},0)$, for any $M\geqslant 0$; we can vary M to find the smallest Lagrangian bound. In assuming a constant Lagrange multiplier across states, this Lagrangian relaxation can be interpreted as requiring the resting constraint to hold "on average"—more precisely, a form of discounted expectation (see, e.g., Adelman and Mersereau 2008)—rather than requiring the constraint to hold in each state.

This Lagrangian relaxation is like the Whittle integral in that it provides an upper bound on the system-wide value function in terms of the cluster-specific value functions $\phi_i(x_i,M)$. We can, however, prove that the Lagrangian relaxation bound is weaker than the bound provided by the Whittle integral.

Proposition 4.4 (Comparing the Whittle Integral and Lagrangian Relaxation Bounds). *Let* $\Phi(\mathbf{x},M)$ *denote the value function for the bandit superprocess as*

defined in Equation (2) *and let* $\hat{\Phi}(\mathbf{x}, M)$ *be the Whittle integral defined by Equation* (5)*. For any* $\mathbf{x}$ *and any* $M \geqslant 0$*, we have*

$$\Phi(\mathbf{x}, M) \leqslant \hat{\Phi}(\mathbf{x}, M) \leqslant \hat{L}(\mathbf{x}, M). \tag{12}$$

Moreover, $\Phi(\mathbf{x}, 0) \leqslant \hat{\Phi}(\mathbf{x}, 0) \leqslant \inf_{\{M \geqslant 0\}} \hat{L}(\mathbf{x}, M)$.

Proof. We established $\Phi(\mathbf{x}, M) \leqslant \hat{\Phi}(\mathbf{x}, M)$ in Proposition 4.2, so we need to prove $\hat{\Phi}(\mathbf{x}, M) \leqslant \hat{L}(\mathbf{x}, M)$. The proof relies on the Weierstrass product inequality: given $\alpha_1, \ldots, \alpha_N$ such that $0 \leqslant \alpha_i \leqslant 1$ for all i, the Weierstrass product inequality says that $\prod_{i=1}^{N}(1-\alpha_i) + \sum_{i=1}^{N} \alpha_i \geqslant 1$. Rearranging into the form in which we will use it, this is equivalent to $-\prod_{i=1}^{N}(1-\alpha_i) \leqslant -1 + \sum_{i=1}^{N} \alpha_i$. Now, starting with the definition of the Whittle integral, for any $B \geqslant M^*(\mathbf{x})$ we have

$$\begin{aligned}
\hat{\Phi}(\mathbf{x}, M) &\equiv B - \int_{m=M}^{B} \prod_{i=1}^{N} \phi_i'(x_i, m)\, dm \\
&\leqslant B + \int_{m=M}^{B} \left(-1 + \sum_{i=1}^{N} (1 - \phi_i'(x_i, m)) \right) dm \\
&= \sum_{i=1}^{N} \phi_i(x_i, M) - (N-1)M = \hat{L}(\mathbf{x}, M).
\end{aligned}$$

The inequality follows from the Weierstrass product inequality taking $\alpha_i = (1 - \phi_i'(x_i, m))$. The next equality follows from basic calculus and algebra, noting that $\phi_i(x_i, B) = B$ for $B \geqslant M^*(\mathbf{x})$.

The final part of the proposition, $\Phi(\mathbf{x}, 0) \leqslant \hat{\Phi}(\mathbf{x}, 0) \leqslant \inf_{\{M \geqslant 0\}} \hat{L}(\mathbf{x}, M)$, follows from noting that $\Phi(\mathbf{x}, M)$ and $\hat{\Phi}(\mathbf{x}, M)$ are both nondecreasing in M, though $\hat{L}(\mathbf{x}, M)$ need not be nondecreasing in M. □

These Lagrangian bounds are particularly weak in the North Sea example. In this example, we can show that the best Lagrangian bound is given by taking $M = 0$, for all choices of clusters; see the online appendix. In this case, the Langrangian $\hat{L}(\mathbf{x}, M)$ reduces to the value given by allowing all of the clusters to be pursued immediately—the sequential aspect of the problem is entirely disregarded. The problem is that, in this example, with a limited number of targets to drill, we can satisfy the constraint of resting $N-1$ clusters "on average" by working on all clusters immediately and resting after the clusters have drilled all of their prospects or quit.

More generally, however, the best Lagrangian bound may be achieved by a value of $M > 0$. Given that $\hat{L}(\mathbf{x}, M)$ is piecewise linear and convex in M (this follows from the fact that the $\phi_i(x_i, M)$ are piecewise linear and convex), we can identify the best Lagrangian bound by a simple line search, e.g., using bisection focusing on the breakpoints for the piecewise linear function.

4.5. Computational Methods

In Appendix A, we describe the "frontier algorithm" that we use to evaluate multiarmed bandits and bandit superprocesses. These calculations are at the heart of our methods for studying sequential exploration problems and it is important to perform them efficiently. In the frontier algorithm, we formulate the cluster-specific dynamic programs (3) as linear programs and then use techniques from parametric linear programming to examine the impact of varying the retirement value. Sequential explorations problems in general, and the North Sea example in particular, have a lot of special structure that can be exploited in this algorithm. For a given cluster, one pass of this algorithm calculates $\phi_i(x_i, m)$ for all states x_i and retirement values m and Gittins indices for all states. The run times for this frontier algorithm varies strongly with cluster size. For instance, with a cluster containing six targets (such as cluster 3 in Figure 2), the algorithm runs in 0.08 seconds. For a cluster with nine targets (such as cluster three in Figure 3), the algorithm takes about four minutes.[2]

5. Heuristics and Bounds for Sequential Exploration Problems

Having studied bandits and bandit superprocesses, which assume independence among the clusters, we now return to the full sequential exploration problem with dependence among clusters. In this section, we consider heuristics and bounds for the sequential exploration problem that are based on the results and methods for bandits and bandit superprocesses.

5.1. Bandit-Based Approximations and Heuristics

To find heuristics, we simplify and approximate the sequential exploration problem (1) in two ways:

- We approximate the initial (true) joint probability distribution $P(\boldsymbol{\omega} \mid \mathbf{x}^\circ)$ by a distribution that assumes the clusters evolve independently according to their marginal distributions under the true distribution. That is, we replace $P(\boldsymbol{\omega} \mid \mathbf{x}^\circ)$ with $P_\alpha(\boldsymbol{\omega} \mid \mathbf{x}^\circ) \equiv \prod_{i=1}^{N} P(\omega_i \mid \mathbf{x}^\circ)$ and thereby approximate the sequential exploration problem with a bandit superprocess.
- We approximate this bandit superprocess with a multiarmed bandit by assuming fixed policies $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N)$ for choosing actions within each cluster. In this case, rather than the DM selecting actions from a full set of actions $A(x_i)$ for each cluster as in (2), the set of actions consists of a single action $\pi_i(x_i)$ from $A_i(x_i)$ selected by policy π_i or the empty set if π_i calls for retiring in state x_i.

We can then write the approximating multiarmed bandit problem with fixed policies $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N)$ as

$$\begin{aligned}
\Phi_\alpha^\pi(\mathbf{x}) \equiv \max \Big\{ 0, \max_{\{i:\, \pi_i(x_i) \neq \varnothing\}} \mathbb{E}_\alpha [\tilde{r}_i(x_i, \pi_i(x_i)) \\
+ \delta \Phi_\alpha^\pi(\tilde{\mathbf{x}}(\mathbf{x}, \pi_i(x_i))) \mid x_i] \Big\},
\end{aligned} \tag{13}$$

where $\mathbb{E}_\alpha$ denotes expectations taken using the approximate distribution P_α. The optimal solution for this approximate

model is to work on the cluster with the maximum Gittins index for the given state or to quit if the maximum Gittins index is less than or equal to zero. These Gittins indices can be calculated using the frontier algorithm of Appendix A.

There is considerable flexibility in the choice of fixed policies in the approximate bandit model (13). In our numerical experiments with the North Sea example, we will use fixed policies π_i that are optimal for cluster i with retirement value $M=0$ (i.e., corresponding to $\phi_i(x_i,0)$), given the independent marginal distribution for the cluster, $P(\omega_i \mid \mathbf{x}^\circ)$. These policies can also be calculated using the frontier algorithm of Appendix A.

There are many other possible choices for fixed policies. For example, we could use a policy corresponding to $\phi_i(x_i,M)$ for some value of M other than zero. In a different setting, Pandey et al. (2007) suggest choosing the action in state x_i that maximizes $\phi_i(x_i,M_i(x_i))$, where $M_i(x_i)$ is the Gittins index for that state. We experimented with such a policy in the North Sea example and it did not perform as well as choosing actions that maximize $\phi_i(x_i,0)$. Intuitively, as discussed in §4.3, the actions selected with large retirement values assume that the DM will retire before drilling at the less attractive targets in a cluster. However, in reality, rather than "retiring," the DM may return to work on the cluster after working elsewhere. With a large value of M, the optimization problem may not appreciate actions that provide information about or otherwise benefit these less attractive targets. However, these additional benefits could be valuable if the DM returns to work on the cluster. In contrast, the optimization problem with $M=0$ assumes that the DM will work on cluster i without interruption and may overvalue actions that provide information about the less attractive targets. Thus, the performance of different fixed policies for a cluster may depend on the options available at other clusters as this determines when/if the DM will return to a cluster after initially "retiring" and when/if the benefits of learning will be realized.

The optimal policy for this approximate multiarmed bandit (13) can be used to define a heuristic policy for the original sequential exploration problem. We can evaluate the performance of these heuristics using Monte Carlo simulation, by drawing samples from the true joint distribution for the full system $P(\boldsymbol{\omega} \mid \mathbf{x}^\circ)$ (taking into account all dependence in the model) and calculating the rewards generated using this heuristic in each simulated scenario.[3] We can use these bandit approximations in either a static or sequential approach:

- In the static approach, we approximate the full model once before running the simulation. For each cluster, this requires performing Bayesian inference to calculate the cluster-specific distributions $P(\omega_i \mid \mathbf{x}^\circ)$ given the initial system state $\mathbf{x}^\circ$ and determining the corresponding fixed policies π_i and Gittins indices for all states. In the simulation, in each period and in any state encountered, we work on the cluster with the maximum Gittins index for that state and choose actions according to the assumed fixed policy for that cluster. This process continues until we encounter a system state in which no clusters have positive Gittins indices.
- In the sequential approach, we update the distributions and policies used in the approximate bandit in each period, taking into account all observed results up to that time. That is, in each period, we update $P_\alpha(\boldsymbol{\omega} \mid \mathbf{x})$ to be $\prod_{i=1}^N P(\omega_i \mid \mathbf{x})$, where $\mathbf{x}$ is the current system state and then resolve the approximate model (13). In each period, we use Bayesian inference to calculate the cluster-specific distributions $P(\omega_i \mid \mathbf{x})$, for each cluster, given the current state; we then determine the corresponding fixed policies π_i and Gittins indices for each cluster.[4] In the simulation, we then work on the cluster with the maximum Gittins index in the current version of the approximate model, choosing actions according to the current fixed policy for the chosen cluster. This process continues until we encounter a system state in which, after updating, no clusters have positive Gittins indices.

Intuitively, with the static approach, we capture the learning within each cluster (e.g., how results for one target affect decisions about other targets in the same cluster) but entirely ignore the value of learning across clusters. The sequential approach requires more work than the static approach but captures cross-cluster learning in an approximate manner by updating the bandit approximation as results are revealed. Both start with the same approximate model and thus select the same first action; the recommendations may differ in later periods as the approximations are updated in the sequential approach.

5.2. Upper Bounds on the Value of an Optimal Policy

As these bandit-based heuristics are feasible for the sequential exploration problem, the expected discounted reward generated by a heuristic provides a lower bound on the value associated with an optimal policy. We complement these lower bounds on the optimal value with upper bounds given by considering an information relaxation (Brown et al. 2010) in which the DM makes optimal decisions using more information than would be available in the real problem.

We will consider an information relaxation where we evaluate each cluster in isolation, assuming that it has perfect information about the results for all other clusters. We study this "clairvoyant relaxation" using simulation. In each trial of the simulation, we generate a random outcome $\hat{\boldsymbol{\omega}}=(\hat{\omega}_1,\ldots,\hat{\omega}_N)$ of the underlying uncertainty for the full model, with $\hat{\omega}_i$ representing the outcome for cluster i. The samples are drawn according to the full joint probability distribution $P(\boldsymbol{\omega} \mid \mathbf{x}^\circ)$, taking into account all dependence in the model. In each simulated scenario, we do the following:

- We first calculate marginal distributions on outcomes for each cluster, conditioned on observing the outcomes for all other clusters; that is, we calculate $P(\omega_i \mid \hat{\boldsymbol{\omega}}_{\bar{i}}, \mathbf{x}^\circ)$, where

$\hat{\boldsymbol{\omega}}_{\bar{i}}$ denotes the sample outcome for all clusters other than cluster i.

- We then approximate the sequential exploration problem with a bandit superprocess (2) by replacing the initial (true) joint probability distribution $P(\boldsymbol{\omega}\,|\,\mathbf{x}^\circ)$ with $P_{\hat{\boldsymbol{\omega}}}(\boldsymbol{\omega}\,|\,\mathbf{x}^\circ) \equiv \prod_{i=1}^{N} P(\omega_i\,|\,\hat{\boldsymbol{\omega}}_{\bar{i}}, \mathbf{x}^\circ)$, so each cluster evolves independently according to the clairvoyant marginal distributions.

If we let $\mathbb{E}_{\hat{\boldsymbol{\omega}}}$ denote expectations taken with respect to this clairvoyant distribution $P_{\hat{\boldsymbol{\omega}}}$, we can write the approximating bandit superprocess for the sample scenario $\hat{\boldsymbol{\omega}}$ as

$$\Phi_c(\mathbf{x};\hat{\boldsymbol{\omega}}) \equiv \max\Big\{0, \max_i \max_{a_i \in A_i(x_i)} \mathbb{E}_{\hat{\boldsymbol{\omega}}}[\tilde{r}_i(x_i,a_i) + \delta\Phi_c(\tilde{\mathbf{x}}(\mathbf{x},a_i);\hat{\boldsymbol{\omega}})\,|\,x_i]\Big\}. \tag{14}$$

Averaging the values found in each inner problem (14) provides an estimate of an upper bound on the value of the original problem (1).

Proposition 5.1 (Clairvoyant Bound). *Let $\tilde{\boldsymbol{\omega}}$ denote the random selection of outcomes drawn according to the full joint distribution with initial state $\mathbf{x}^\circ$. Then $V(\mathbf{x}^\circ) \leqslant \mathbb{E}[\Phi_c(\mathbf{x}^\circ;\tilde{\boldsymbol{\omega}})]$.*

Proof. See the online appendix. □

The information relaxation used here is different from those considered in Brown et al. (2010) or elsewhere in that different parts of the model are given different sets of information. We liken this information relaxation to the state of information in the card game "blind man's bluff," where each player knows the cards of all other players, but not their own. Intuitively, with this information relaxation, we properly capture the learning within each cluster (e.g., how results for one target affect decisions about other targets in the same cluster) but we overestimate the value of learning across clusters.

Although we may not be able to solve the scenario-specific bandit superprocesses (14) exactly, we can calculate upper bounds on $\Phi_c(\mathbf{x};\hat{\boldsymbol{\omega}})$ using the Whittle integral or Lagrangian relaxation, as discussed in §4. To formalize this, we let $\hat{\Phi}_c(\mathbf{x};\hat{\boldsymbol{\omega}})$ and $\hat{L}_c(\mathbf{x};\hat{\boldsymbol{\omega}})$ denote the Whittle integral (5) and Lagrangian relaxation (11) defined for the bandit superprocess (14) with outcome $\hat{\boldsymbol{\omega}}$, taking the retirement value M that yields the minimum value. (For the Whittle integral, the minimizing M is always zero. In the North Sea example, the minimizing value for the Lagrangian bound is always $M=0$, but, more generally, the minimizing value of M may vary with $\hat{\boldsymbol{\omega}}$.) Then, combining Proposition 5.1 with results in §4, we have the following bounds.

Corollary 5.2 (Clairvoyant Whittle and Lagrangian Bounds). *Let $\tilde{\boldsymbol{\omega}}$ denote the random selection of outcomes drawn according to the full joint distribution with initial state $\mathbf{x}^\circ$, as in Proposition 5.1. Then*

$$V(\mathbf{x}^\circ) \leqslant \mathbb{E}[\Phi_c(\mathbf{x}^\circ;\tilde{\boldsymbol{\omega}})] \leqslant \mathbb{E}[\hat{\Phi}_c(\mathbf{x}^\circ;\tilde{\boldsymbol{\omega}})] \leqslant \mathbb{E}[\hat{L}_c(\mathbf{x}^\circ;\tilde{\boldsymbol{\omega}})]. \tag{15}$$

We will refer to $\mathbb{E}[\hat{\Phi}_c(\mathbf{x}^\circ;\tilde{\boldsymbol{\omega}})]$ and $\mathbb{E}[\hat{L}_c(\mathbf{x}^\circ;\tilde{\boldsymbol{\omega}})]$ as the *clairvoyant Whittle bound* and the *clairvoyant Lagrangian bound*, respectively. We present results for these two bounds in the North Sea example in §6.

5.3. On the Choice of Clusters

Combining clusters into larger clusters will lead to tighter clairvoyant bounds as there is less information gained in the information relaxation. For instance, in the North Sea example, the clairvoyant bound given by a model that treats each target as its own cluster will be weaker than the bound given by a model that groups together all of the targets associated with a given prospect. Similarly, a bound given by grouping prospects together (such as that of Figure 2) will be better than the bound given by treating prospects (or targets) as separate clusters. Larger clusters will also typically lead to better lower bounds, as the heuristics will better capture the dependence in the problem. The trade-off is that it typically requires more computational effort to evaluate larger clusters.

How should we choose clusters? A natural approach is to start with the smallest clusters possible, then compute upper and lower bounds with these simple clusters; if the gap between bounds is small, we are done. Otherwise, we can group together some of the clusters in order to reduce the gap between lower and upper bounds.

One way to identify clusters to be combined is to see which clusters are benefitting most from the additional information in the clairvoyant relaxation. To examine this, we can calculate the cluster-specific value function $\phi_{ci}(x_i^\circ;\hat{\boldsymbol{\omega}}_{\bar{i}})$ given knowledge of all other cluster results in each trial of the simulation (assuming zero retirement value); this requires little additional work as this value would be determined as part of the frontier algorithm used to calculate Whittle integrals or Lagrangian bounds. The average of these cluster-specific values, $\mathbb{E}[\phi_{ci}(x_i^\circ;\tilde{\boldsymbol{\omega}}_{\bar{i}})]$, provides an upper bound on the maximum expected reward that can be obtained from cluster i under any policy. We can contrast this upper bound with the cluster-specific value $\phi_i(x_i^\circ)$ computed using the unconditional marginal distribution of cluster i. A large difference between these values suggests that cluster i is benefitting a great deal from the information provided by other clusters.

We would consider combining those clusters that are benefitting significantly from the additional information with other clusters that are providing this valuable information. In the North Sea example, the clusters that are most strongly related to a given cluster are its upstream or downstream neighbors in the Bayesian network. In other applications, we might identify strongly related clusters by looking at a correlation matrix or a multiple regression model that relates rewards or outcomes for the different clusters. We could also perhaps use entropy (or mutual information) methods to suggest strongly related clusters. In practice, we would expect application-specific features to

play a key role in guiding the choice of clusters. Moreover, we would expect there to be some degree of experimentation in this process.

6. Numerical Results for the North Sea Example

In this section, we discuss the use of Monte Carlo simulation to evaluate the performance of our heuristics and bounds in the North Sea example. We first describe the computations and then discuss the results.

6.1. Computations and Run Times

In our simulations, we consider four different sets of clusters: one cluster for each target, one cluster for each prospect, the "medium cluster" case shown in Figure 2, and the "large cluster" case of Figure 3. To illustrate the effects of different degrees of dependence among targets, we consider the model with and without uncertainty about the state of the kitchens, as discussed in §2. We ran 400 trials for each simulation, using one set of samples for all cases without kitchen uncertainty and another set of samples for the cases with kitchen uncertainty. In each trial of the simulation, we generate outcomes for all 25 targets according to the full joint probability distribution, taking into account all dependence in the problem.

In these simulations, we will consider static and sequential bandit-based heuristics (as discussed in §5.1) and the clairvoyant Whittle and Lagrangian bounds (as discussed in §5.2). We use the Bayes Net Toolbox for MATLAB (Murphy 2001) to calculate and update the probability distributions and our own MATLAB code to implement the frontier algorithm of Appendix A. Because both clairvoyant bounds (and the first-action-fixed clairvoyant bounds discussed in §6.3) require essentially the same information, we calculated these bounds simultaneously and report a single run time for all clairvoyant bounds. The run times and results are summarized in Table 1((a) and (b)).

As is evident in Table 1, the run times vary by cluster size. With smaller clusters, the run times are dominated by the time it takes to update the probability distributions for each cluster; the frontier algorithm is very fast. However, with larger clusters, the run times are dominated by the time it takes to run the frontier algorithm. This is evident in the run times for the clairvoyant bounds in the cases with the large clusters: each trial takes approximately five minutes and 400 trials takes more than 30 hours. Note, however, that the mean standard errors in this case are quite small and we could have achieved good accuracy with many fewer trials. Because of run time concerns, we did not attempt to evaluate the sequential heuristic in the cases with large clusters.

The run times also differ across the cases with and without kitchen uncertainty because of the differences in the probability distributions involved. On one hand, the heuristic policies are easier to evaluate with kitchen uncertainty because the policies tend to quit sooner; the possibility that

Table 1(a). Results without kitchen uncertainty (400 samples).

	Clusters = one target	Clusters = one prospect	Medium clusters (Figure 2)	Large clusters (Figure 3)
Static bandit heuristic				
Mean ($M)	20,217	22,451	23,019	23,150
Mean standard error ($M)	104	41	33	5
Run time (seconds)	280	281	290	422
Sequential bandit heuristic				
Mean ($M)	22,714	22,991	23,148	—
Mean standard error ($M)	65	42	34	—
Run time (seconds)	697	3,892	2,735	—
Clairvoyant Whittle bound				
Mean ($M)	25,244	24,077	23,416	23,248
Mean standard error ($M)	19	13	8	2
Duality gap ($M)	2,530	1,086	268	98
Gap as a fraction of bound (%)	10.0	4.5	1.1	0.4
Clairvoyant Lagrangian bound				
Mean ($M)	27,435	26,138	25,141	24,479
Mean standard error ($M)	33	20	8	1
Duality gap ($M)	4,721	3,147	1,993	1,329
Gap as a fraction of bound (%)	17.2	12.0	7.9	5.4
Clairvoyant Whittle bound with fixed first action				
Mean ($M)	25,079	23,960	23,304	23,248
Mean standard error ($M)	19	11	5	2
Target with max	1A	13B	13B	10B
Duality gap ($M)	2,365	969	156	98
Gap as a fraction of bound (%)	9.4	4.0	0.7	0.4
Run time for all clairvoyant bounds (seconds)	417	227	332	111,374

Table 1(b). Results with kitchen uncertainty (400 samples).

	Clusters = one target	Clusters = one prospect	Medium clusters (Figure 2)	Large clusters (Figure 3)
Static bandit heuristic				
Mean ($M)	12,272	16,328	17,058	17,717
Mean standard error ($M)	146	84	63	19
Run time (seconds)	215	243	278	420
Sequential bandit heuristic				
Mean ($M)	16,915	17,409	17,688	—
Mean standard error ($M)	100	71	51	—
Run time (seconds)	595	3,163	2,660	—
Clairvoyant Whittle bound				
Mean ($M)	20,875	19,639	18,464	17,894
Mean standard error ($M)	34	30	15	6
Duality gap ($M)	3,960	2,230	776	177
Gap as a fraction of bound (%)	19.0	11.4	4.2	1.0
Clairvoyant Lagrangian bound				
Mean ($M)	22,437	21,101	19,594	18,698
Mean standard error ($M)	75	46	22	6
Duality gap ($M)	5,522	3,692	1,906	981
Gap as a fraction of bound (%)	24.6	17.5	9.7	5.2
Clairvoyant Whittle bound with fixed first action				
Mean ($M)	20,515	19,404	18,239	17,894
Mean standard error ($M)	30	26	15	6
Target with max	10B	13B	12A	10B
Duality gap ($M)	3,600	1,995	551	177
Gap as a fraction of bound (%)	17.5	10.3	3.0	1.0
Run time for all clairvoyant bounds (seconds)	419	230	354	126,585

the kitchens may be dry makes all targets less attractive. On the other hand, kitchen uncertainty increases the possibilities for cross-prospect learning; this tends to lead to more complicated policies and more iterations in the frontier algorithm, particularly with clusters that include more than one prospect.

The means and mean standard errors in Table 1 have been adjusted using control variates. We describe these control variates in detail in the online appendix and provide a brief overview here:

• For the clairvoyant bounds, the control variates are based on the expected values for the individual clusters with a fixed policy. These cluster-specific expected values can be computed exactly and are also estimated in the simulations; the differences between the simulated values and exact means can then be used as control variates in a standard multiple-regression-based approach (see, e.g., Glasserman 2004). In our experiments, these control variates reduce the mean standard errors by a factor of 10–20.

• For the heuristic policies, we use the "approximating martingale-process method" of Henderson and Glynn (2002). The basic idea of this approach is to use samples and expectations for an approximate value function (here the value function for the approximating multiarmed bandit (13)) to construct a control variate for the more complex system. The effectiveness of these control variates will depend on how well the approximate value function replicates values for the more complex system. In our experiments, these control variates reduce the mean standard errors by a factor of 5–100.

In general, these control variate calculations require very little additional work. The improvements in accuracy provided by these control variate corrections are very important: without them, it would be difficult to make meaningful comparisons of the heuristics and bounds without running many more trials.

6.2. Results

In Table 1, we see that the sequential heuristic outperforms the static heuristic in every case. This is not surprising: As discussed in §5.1, the static heuristic captures learning within clusters, but does not capture learning across clusters. The sequential heuristic incorporates cross-cluster learning by updating this approximate model in each period. With larger clusters, both heuristics do better and the differences between the two heuristics decrease as more learning is captured within the clusters. We also note that the differences between the sequential and static heuristics are more pronounced when there is uncertainty about the state of the kitchens; cross-cluster learning plays a greater role in this case.

Examining the clairvoyant bounds, we see that the Whittle and Lagrangian bounds both improve with larger clusters, as there is less information gained in the information relaxation with larger clusters. Also as expected, the Lagrangian bounds are substantially worse than the Whittle bounds: the differences range from approximately $800M to $2,200M. The duality gap—the difference between the upper bound and the best lower bound (the sequential heuristic, when computed)—is smallest for the Whittle

bounds with the large clusters: the static heuristic is within 0.4% of the optimal value without kitchen uncertainty and within 1.0% with kitchen uncertainty. Comparing the magnitudes of these duality gaps to the estimates of the slack in the Whittle integrals presented at the end of §4.3, it appears that with the small and medium clusters most of the duality gap is due to the information relaxation (i.e., the first inequality in (15)) as opposed to using the Whittle integral as a bound (the second inequality in (15)). However, in the case with the large clusters, there is much less information lost in the relaxations and the slack in the Whittle integral, though small in an absolute sense, may represent a larger fraction of the relatively small duality gap.

The mean standard errors associated with each estimate also decrease for larger clusters. For the heuristics, this improved accuracy is a consequence of the fact that the control variates improve as the underlying approximate value functions improve with larger clusters. For the clairvoyant bounds, the improved accuracy reflects the fact that there is less information gained with larger clusters; this implies that the clairvoyant bounds are less variable across the different conditioning scenarios.

6.3. Refinements

Although we have presented the results for all cases simultaneously, our choices of clusters proceeded iteratively with the choices at each stage guided by the detailed results for earlier, smaller clusters. As discussed in §5.3, after each simulation, we looked at how much each cluster, when considered in isolation, benefitted from the additional information provided in the information relaxation. When moving from smaller to larger clusters, we combined those clusters that benefited most with other clusters that were closely related in the Bayesian network of Figure 1. For example, after examining the results for the case with clusters corresponding to prospects, these considerations led us to combine prospects 7 and 11 and their neighbors into one cluster and prospects 5 and 9 into another cluster. (We provide detailed cluster-specific results in the online appendix.)

Similarly, we looked at the details of the optimal policies for the cluster-specific problems to see how likely we were to drill various targets. As discussed in §A.1, the decision variables $\boldsymbol{\theta}$ in the linear programming representation (18) of the cluster-specific dynamic program represent the discounted expected time spent in each state-action pair. If a target is never drilled, the entries in $\boldsymbol{\theta}$ corresponding to drilling that target will always be zero. In early results with smaller clusters, we noticed that targets 6C and 8A were never drilled in any scenario encountered (with or without kitchen uncertainty). Considering these targets more carefully, we were able to prove that these two targets should never be drilled (see the online appendix). Given this, we dropped these targets in the case with large clusters, meaning we assumed that the targets would never be drilled and that their results would never be observed. Dropping these targets has no effect on the values for the heuristic policies, as the targets are never drilled with these heuristic policies. However, dropping the targets may improve (and cannot harm) the clairvoyant bounds as it reduces the information provided to other clusters.

Finally, we looked at the possibility of constraining the actions selected in the clairvoyant bounds. Although any nonclairvoyant DM must start by drilling a particular target and cannot change this choice in response to unseen future events, the clairvoyant DM has the flexibility to change this choice based on outcomes for other clusters. We can refine the clairvoyant bounds by calculating a set of bounds—one bound for each possible first-period action—where the first-period action is constrained to match the given action. Because any feasible exploration policy has to drill at some target first, the maximum of these first-action-fixed bounds provides an upper bound for the optimal sequential exploration problem; moreover, this bound cannot be any worse than the unconstrained clairvoyant bound. Because the frontier algorithm provides cluster-specific value and derivative information for all states simultaneously, these fixed-first-action bounds require little additional work beyond that required for computing the unconstrained Whittle or Lagrangian bounds.

Table 1 reports the best first-action-fixed Whittle bounds, as well as the target that attains the maximum in each case. Here we see that imposing this constraint on the first action improves the bounds, particularly in the cases with smaller clusters. There is no improvement in the cases with large clusters, as in these cases, the clairvoyant DM is learning relatively little and the first action does not change across scenarios. We discuss the calculation of these first-action-fixed bounds in detail in the online appendix and provide full results for the case with large clusters.

6.4. Recommendations

So, which target should we drill first? And then what? In both cases with and without kitchen uncertainty, for all cluster sizes, the bandit-based heuristics recommend drilling at target 10B. Although we cannot prove that is optimal to start with 10B, we can rule out starting at any target other than 10B or 13B in the case without kitchen uncertainty (10B, 6B, or 13B in the model with kitchen uncertainty): considering the clairvoyant bounds with fixed actions (see Table 2 in the online appendix), the upper bounds on performance for policies that start with the ruled-out targets are worse than the expected value following the static heuristic. It is also telling that with the large clusters, the unconstrained clairvoyant bounds match those bounds that are restricted to start with 10B. Although this does not prove that starting with 10B is optimal, we can be sure that if we start with 10B and follow the static heuristic with large clusters, we will be within 0.4% of the expected value with an optimal policy (or 1.0% for the case with kitchen uncertainty).

Assuming we drill 10B first, the choice for the next well to drill depends on the outcome for 10B. If we find oil or

gas at 10B, in all cases, the heuristics recommend drilling at target 10C. We learn little new by drilling at 10C next, because finding oil or gas at 10B has already revealed that prospect 10 contains oil or gas. Instead, we reap some of the benefits of what we learned from 10B: success at 10B makes 10C very attractive and we drill there right away.

If we find 10B to be dry, the recommendations vary by heuristic and by cluster size. For example, without kitchen uncertainty and with medium clusters, the static heuristic recommends drilling at 6A next, whereas the sequential heuristic with medium clusters and static heuristic with large clusters recommend drilling 13B next. In the medium cluster case, prospect 6 is in a different cluster than 10; the static heuristic therefore does not take into account the negative implications that failure at 10B has for target 6A. With large clusters, prospects 6, 10, and 13 are all in the same cluster and these negative implications are captured by the static heuristic. The sequential heuristic with medium clusters also takes these negative implications into consideration when it updates the approximation after seeing 10B is dry. We would recommend following the more sophisticated heuristics and drilling at 13B in those cases where 10B is found to be dry.

6.5. Comparison with Other Heuristics

It is instructive to compare our bandit-based heuristics with the *naive*, *myopic*, and *limited-look-ahead* heuristics that Martinelli et al. (2012) considered for the North Sea example. In the naive heuristic, the DM neglects all learning and ranks all targets according to the (initial) unconditional expected value associated with drilling at that target; the DM then drills those that have positive expected values in order. This naive heuristic is exactly equivalent to the static bandit heuristic with clusters corresponding a single target: it is not hard to see that in this case the Gittins indices would rank targets in the same way as the unconditional expected values. The myopic heuristic also ranks targets by their unconditional expected values, but updates these expected values at each stage taking into account any observed results; this is exactly equivalent to sequential bandit heuristic with clusters corresponding to individual targets. In Table 1 we see that the myopic heuristic outperforms the naive heuristic, but does not perform as well as the heuristics based on larger clusters.

A limited-look-ahead heuristic (see, e.g., Bertsekas 1995) truncates the time horizon of the dynamic programming model and at each stage uses a decision based on looking ahead a small number of stages. For instance, in this example, a one-step look-ahead heuristic chooses a cluster and action (i and a_i) (i.e., a target to drill) in a given period to maximize $\mathbb{E}[\tilde{r}_i(x_i, a_i) + \delta \bar{V}(\tilde{\mathbf{x}}(\mathbf{x}, a_i)) \,|\, \mathbf{x}]$, where $\mathbf{x}$ represents the current state of the full system and $\bar{V}$ is some easy-to-compute approximation of the true continuation value function. The heuristic calls for quitting if the values are all negative. Martinelli et al. (2012) propose taking $\bar{V}$ to be the value under the naive heuristic.

In a setting with many possible actions, the look-ahead heuristics can be time consuming to evaluate. For instance in the North Sea example, for each available action in the first stage (there are 25 choices), we need to calculate marginal probabilities for each possible outcome (three possibilities) conditional on the current state of information. To evaluate the approximate continuation value, we need conditional probabilities for the possible outcomes for the next period for each possible (24) action in the next period in every first period scenario: in all, we need to solve $25 \times 3 \times 24 = 1{,}800$ different Bayesian inference problems just to calculate these approximate continuation values for the first-stage decision. A two-step look-ahead heuristic would require solving $1{,}800 \times 3 \times 23 = 124{,}200$ different Bayesian inference problems to calculate the approximate continuation values for the first decision. Thus, looking ahead in depth is quite computationally expensive.

We evaluated these one-step look-ahead heuristics using the same set of 400 scenarios used to generate the results in Table 1. In the case with no kitchen uncertainty, the simulation took approximately 16,400 seconds (4.5 hours) to run and, using control variates, the mean was \$22,239M with a mean standard error of \$70M. Strikingly, this one-step look-ahead heuristic performs worse than the simpler myopic heuristic; it also performs worse than the easier-to-evaluate heuristics based on larger clusters. In the case with kitchen uncertainty, the simulation took 16,492 seconds and the mean was \$16,891M, with a mean standard error of \$70M. This is comparable to the myopic heuristic but worse than the heuristics based on larger clusters.

To understand the performance of the one-step look-ahead heuristics, it helps to consider the actions selected by the heuristics; we will focus on the case with no kitchen uncertainty. As discussed above, the myopic heuristic and cluster-based heuristics start by drilling at target 10B. The one-step look-ahead heuristic instead starts at target 9B, which is one of the targets that was ruled out as optimal in §6.4. Target 9B provides a lot of information about other targets, but is not very attractive in itself. The one-step look-ahead heuristic chooses to drill target 9B first because in the one-step look-ahead model the DM only has one period to "learn" and places a great deal of value of information provided by this first well. In the next period, the heuristic again assumes the DM has only one period to learn and chooses accordingly, thereby deferring the earning associated with drilling at attractive targets. In contrast, the bandit heuristics based on larger clusters recognize that the DM can choose the most attractive target initially and then learn later.

As discussed in the introduction, the key trade-off in these sequential exploration problems is between earning immediately and learning to improve future earnings. Although it can be expensive to "look ahead" in models with many actions, some degree of looking ahead is necessary to properly value opportunities for learning. We can view the bandit-based heuristics as being a form of limited-look-ahead heuristic that manages computational

complexity by looking ahead only within the chosen clusters, where learning is likely to play the biggest role. We then make the trade-off between earning and learning by working on the cluster with the largest Gittins index, with the index reflecting both learning and earning potential for a given cluster.

7. Conclusions

Although we cannot claim to have fully solved the North Sea example that motivated this work, by iteratively refining our approximations and heuristics, we have come very close to an optimal policy. Moreover, through the clairvoyant bounds, we *know* that we have come very close to optimality. These clairvoyant bounds are very useful for modelers: it is always tempting to tinker with heuristics to see if we can do better and, in complex models, evaluating these variations can be time consuming. Without an upper bound on the performance, we would never know when we have reached the point of diminishing returns and may tinker endlessly. With good bounds, we know when we cannot do much better.

What is required for this approach for studying sequential exploration problems to work well? The main requirement appears to be the ability to capture "most" of the dependence in the model within clusters that are of a manageable size, i.e., of a size such that we can find the Gittins indices required for the bandit-based heuristics and repeatedly evaluate the Whittle integrals in a reasonable amount of time. Typically, the state spaces for these cluster-specific problems grows exponentially as we combine simple clusters (or targets) into larger clusters. Thus it helps to have small, discrete state-action spaces for the targets and a correlation structure among targets that is, intuitively, "approximately block diagonal" with blocks (corresponding to clusters) that are not too large. We also need to be able to do probabilistic inference (i.e., Bayesian updating) in the full model reasonably efficiently in order to calculate the cluster-specific marginal distributions used in the heuristics and bounds.

Several interesting questions and challenges remain:

- We chose clusters by studying the network and detailed simulation results, proceeding in a rather ad hoc manner. Is there some way to systematically choose and refine the definition of clusters?
- In other applications of information relaxations to provide bounds for stochastic dynamic programs, it is helpful to incorporate penalties that charge the clairvoyant DM for using additional information (see Brown et al. 2010). Although the bounds in our example are "good enough" without such penalties, would it be possible to obtain good bounds with less effort (e.g., with smaller clusters) using some kind of penalty?
- Our analysis has assumed that the DM must proceed sequentially, considering one cluster or, in the example, one target at a time. What if there was a possibility of using more than one drilling rig so the DM can drill two or more targets simultaneously? What kinds of heuristics and bounds would be helpful in these settings? Does the Whittle integral generalize?
- Finally, we might consider the case of "restless bandits with correlated arms" where cluster states may change when not active or, more generally, "weakly coupled dynamic programs" with dependent subproblems. In this setting, we could proceed as in §5 and generate heuristics and upper and lower bounds on values by considering independent and clairvoyant approximations of the full problem with dependence. Although we could no longer use bandit and bandit superprocess methods to study these approximate models, we could perhaps use Lagrangian and/or linear programming relaxations instead (as in, e.g., Adelman and Mersereau 2008).

Although we have focused on a specific example of oil and gas exploration in this paper, we believe that there are many other problems that have a similar structure. As discussed in the introduction, many classic applications of the bandit problem (e.g., job scheduling, targeted advertising, clinical trials) may have dependent arms. In these settings, we may be able to use heuristics and bounds that are analogous to those used here. Moreover, the results and algorithms for bandit superprocesses that we have developed may be useful in these and other settings.

Supplemental Material

Supplemental material to this paper is available at http://dx.doi.org/10.1287/opre.2013.1164.

Appendix A. Computational Methods

To evaluate our heuristics and bounds for sequential exploration problems, we need to calculate cluster-specific value functions $\phi_i(x_i, M)$ and their derivatives for a variety of retirement values and calculate Gittins indices for a variety of states. We will formulate the cluster-specific dynamic programs as linear programs and solve them using techniques that combine ideas from parametric linear programming and policy iteration methods for solving dynamic programs.

Chen and Katehakis (1986) and Kallenberg (1986) discussed the use of linear programming and parametric linear programming (respectively) for calculating Gittins indices for standard multi-armed bandit problems. We build on these ideas and describe how these methods can be used with bandit superprocesses to compute the cluster-specific value functions and derivatives required to calculate Whittle integrals and Lagrangian bounds, as well as Gittins indices. Although the Whittle integral is frequently used as a theoretical construct to prove the optimality of index-based policies for multiarmed bandit problems (as in Whittle 1980), we are not aware of any prior work that considers actually computing these integrals for multiarmed bandits or bandit superprocesses.

A.1. Formulating the Dynamic Programs as Linear Programs

We first describe the linear programming formulation of the problem of finding the value function $\phi_i(x_i, M)$ for the cluster-specific

dynamic program given by (3) with retirement value M. Linear programming formulations of dynamic programs are standard (see, e.g., Puterman 1994 or Bertsekas 1995).

Let $\boldsymbol{\phi}$ denote the vector of values representing the cluster-specific value function $\phi_i(x_i, M)$; the length of the vector is equal to the number of possible states, $|X_i|$. We let $\boldsymbol{\phi}(x_i)$ denote the entry of $\boldsymbol{\phi}$ corresponding to state x_i. We can then write the linear program as

$$\begin{aligned} \underset{\boldsymbol{\phi}}{\text{minimize}} \quad & \sum_{x_i} \boldsymbol{\phi}(x_i) \\ \text{subject to} \quad & \boldsymbol{\phi}(x_i) - \delta \mathbb{E}[\boldsymbol{\phi}(\tilde{x}_i(x_i, a_i)) \,|\, x_i] \\ & \geqslant \mathbb{E}[\tilde{r}_i(x_i, a_i) \,|\, x_i] \\ & \quad \text{for all } x_i \in X_i, a_i \in A(x_i) \\ & \boldsymbol{\phi}(x_i) \geqslant M \quad \text{for all } x_i \in X_i. \end{aligned} \tag{16}$$

Intuitively, the problem is to find a minimal vector $\boldsymbol{\phi}$ that satisfies the Bellman inequalities, represented as constraints. The first set of constraints captures the maximum over actions a_i in the recursive Equation (3). The second set of constraints captures the possibility of retiring and receiving retirement value M. These inequality constraints will be binding if the action (action a_i or retirement) is optimal for state x_i. Basic feasible solutions for (16) represent different possible policies for the dynamic program and at least one constraint will be binding for each state.

To streamline our discussion, it is convenient to rewrite (16) using matrix and vector notation as

$$\begin{aligned} \underset{\boldsymbol{\phi}}{\text{minimize}} \quad & \mathbf{e}^\top \boldsymbol{\phi} \\ \text{subject to} \quad & \mathbf{A}^\top \boldsymbol{\phi} \geqslant \mathbf{c} + M\mathbf{d}. \end{aligned} \tag{17}$$

Here $\mathbf{A}^\top$ is a $m \times n$ matrix where the m rows represent possible state-action pairs and the n columns represent states; $\mathbf{c}$ is an m-vector representing the expected rewards for $\mathbb{E}[\tilde{r}_i(x_i, a_i) \,|\, x_i]$ for each state-action pair, with zeroes for the retirement actions; $\mathbf{d}$ is an m-vector with ones for state-action pairs corresponding to retirement (earning reward M) and zeroes otherwise; $\mathbf{e}$ is an n-vector of ones.

Note that the basis matrices for this linear program will be of the form $\mathbf{B}^\top = (\mathbf{I} - \delta\mathbf{P})$, where $\mathbf{P}$ is a substochastic matrix representing the probability transition matrix with the policy represented by the basis.[5] Since we have assumed $\delta < 1$, we know $\mathbf{B}^{-1} = (\mathbf{I} - \delta\mathbf{P}^\top)^{-1} = (\mathbf{I} + \delta\mathbf{P} + \delta^2\mathbf{P}^2 + \cdots)^\top$.

The dual of (17) is a linear program in standard form:

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\text{maximize}} \quad & (\mathbf{c} + M\mathbf{d})^\top \boldsymbol{\theta} \\ \text{subject to} \quad & \mathbf{A}\boldsymbol{\theta} = \mathbf{e} \\ & \boldsymbol{\theta} \geqslant 0. \end{aligned} \tag{18}$$

Here the decision variable $\boldsymbol{\theta}$ is an m-vector, with one entry for each state-action pair. In any basic feasible solution, the entries in $\boldsymbol{\theta}$ corresponding to the action selected in a given state (the basic variables $\boldsymbol{\theta}_\mathbf{B}$) will be positive—in fact, greater than one—and the remainder will be zero; thus all basic feasible solutions will be nondegenerate. The $\boldsymbol{\theta}$ variables represent the discounted total expected time spent in each state-action pair with the given policy, summing over the times for all initial states. To see that the basic variables $\boldsymbol{\theta}_\mathbf{B}$ will be greater than one, let $\mathbf{P}$ be the substochastic matrix representing the transitions for the policy represented by basis $\mathbf{B}$ and note that $\boldsymbol{\theta}_\mathbf{B} = \mathbf{B}^{-1}\mathbf{e} = (\mathbf{I} - \delta\mathbf{P}^\top)^{-1}\mathbf{e} = (\mathbf{I} + \delta\mathbf{P} + \delta^2\mathbf{P}^2 + \cdots)^\top\mathbf{e} \geqslant \mathbf{e}$.

A.2. The Frontier Algorithm

To calculate $\phi_i(x_i, M)$ for varying retirement values M, we need to repeatedly solve the linear program (18). Because $\phi_i(x_i, M)$ is piecewise linear, increasing, and convex in M (see Lemma 4.1), this amounts to finding the slopes of the pieces and the breakpoints between pieces. We do this using an algorithm that we will call the "frontier algorithm." The frontier algorithm is a variation of the Gass-Saaty algorithm for parametric linear programming (Gass and Saaty 1955), which can be seen as a variation of the simplex method for solving a linear program.

Algorithm 1 (Frontier algorithm: For all x_i, find $\phi_i(x_i, M)$, $\phi_i'(x_i, M)$ for all $M \geqslant 0$ and $M_i^*(x_i)$)

let $\mathbf{B}_0 = \mathbf{I}$ be an initial basis matrix that is optimal for all $M \geqslant M_0$;
let $\mathbf{g} = 0$ (initialize vector of Gittins indices);
let $j = 0$;
loop
  let $\boldsymbol{\lambda}_{cj}$ and $\boldsymbol{\lambda}_{dj}$ be the solutions to $\mathbf{B}_j^\top \boldsymbol{\lambda}_{cj} = \mathbf{c}_\mathbf{B}$ and $\mathbf{B}_j^\top \boldsymbol{\lambda}_{dj} = \mathbf{d}_\mathbf{B}$ (calculate dual variables);
  let $\bar{\mathbf{c}} = \mathbf{c} - \mathbf{A}^\top \boldsymbol{\lambda}_{cj}$ and $\bar{\mathbf{d}} = \mathbf{d} - \mathbf{A}^\top \boldsymbol{\lambda}_{dj}$ (calculate reduced costs);
  if $\bar{\mathbf{d}} \geqslant 0$ **then**
    let $M_j = 0$ and Stop. (The next M_j is $-\infty$; we are done.);
  end if
  let M_j and $\mathcal{J}$ be the max and argmax (resp.) of $\left\{-\frac{\bar{c}_k}{\bar{d}_k} \colon \bar{d}_k < 0\right\}$;
  if $M_j \leqslant 0$ **then**
    let $M_j = 0$ and Stop. (We are done.);
  else
    for all states x_i included in $\mathcal{J}$, let $\mathbf{g}(x_i) = \max\{\mathbf{g}(x_i), M_j\}$ (update Gittins indices)
    let $\mathbf{B}_{j+1}$ be a new basis matrix with the rows of $\mathbf{B}_j$ representing the states in $\mathcal{J}$ replaced by the corresponding rows of $\mathbf{A}$, ensuring that the basis matrix $\mathbf{B}_j$ consists of rows of $\mathbf{A}$ representing one action for each state (ties among actions for the same state may be broken arbitrarily);
    let $j = j + 1$;
  end if
end loop

The frontier algorithm begins with a large retirement value M_0 and a basis matrix $\mathbf{B}_0$ that is optimal for all $M \geqslant M_0$. Since the discount factor δ is assumed to be strictly less than one, we can take $M_0 = c^*/(1-\delta)$ where the c^* is the maximum element in $\mathbf{c}$. In all states, it is optimal to retire immediately for all $M \geqslant M_0$ and we can take the initial basis matrix $\mathbf{B}_0$ to be an identity matrix. Thus we have $\phi_i(x_i, M) = M$ and $\phi_i'(x_i, M) = 1$ for $M \geqslant M_0$. We then calculate the remaining values, slopes, and breakpoints by iteratively updating the basis while reducing the retirement value M.

Suppose we are given a basis matrix $\mathbf{B}_j$ that is optimal for M_j. This basis will remain optimal for smaller values of M as long as the reduced costs associated with the basis are nonpositive. We can decompose the dual variables and reduced costs associated with this solution into components associated with rewards and retirement. The dual variables $\boldsymbol{\lambda}_{cj}$ and $\boldsymbol{\lambda}_{dj}$ associated with

rewards and retirement are given as the solutions to $\mathbf{B}_j^\top \boldsymbol{\lambda}_{cj} = \mathbf{c}_\mathbf{B}$ and $\mathbf{B}_j^\top \boldsymbol{\lambda}_{dj} = \mathbf{d}_\mathbf{B}$, where $\mathbf{c}_\mathbf{B}$ and $\mathbf{d}_\mathbf{B}$ are the elements of $\mathbf{c}$ and $\mathbf{d}$ associated with the current basis. The reduced cost components are then given as $\bar{\mathbf{c}} = \mathbf{c} - \mathbf{A}^\top \boldsymbol{\lambda}_{cj}$ and $\bar{\mathbf{d}} = \mathbf{d} - \mathbf{A}^\top \boldsymbol{\lambda}_{dj}$. The combined reduced costs $\bar{\mathbf{c}} + M\bar{\mathbf{d}}$ will be nonpositive as long as $M \geqslant M_{j+1} = \max\{-\bar{c}_k/\bar{d}_k : \bar{d}_k < 0\}$. Thus $\mathbf{B}_j$ will remain optimal for $M \geqslant M_{j+1}$.

The algorithm terminates if $M_{j+1} \leqslant 0$ or if $\bar{\mathbf{d}} \geqslant 0$ (which would imply $M_{j+1} = -\infty$). Otherwise, we update the basis matrix $\mathbf{B}_j$. The set of indices $\mathcal{I}$ achieving the maximum ratio $M_{j+1} = \max\{-\bar{c}_k/\bar{d}_k : \bar{d}_k < 0\}$ represents the state-action pairs entering the basis in this iteration. We replace the rows in $\mathbf{B}_j$ corresponding to the set of states in $\mathcal{I}$ with the rows from $\mathbf{A}$ representing the state-action pairs in $\mathcal{I}$. To ensure the validity of the new basis, we need to be sure that the basis includes one action for each state. If the index set $\mathcal{I}$ includes two or more state-action pairs for a single state, we can break the tie among actions arbitrarily and choose any one of these actions. This leads to a new basis that is optimal over the next range of retirement values. (We prove this as part of the proposition below.)

The value function and its derivatives are given by the dual variables constructed in the algorithm and the Gittins indices are determined by noting the value of M_{j+1} that leads to the first change action for a given state; these are stored in $\mathbf{g}$ in the algorithm. We summarize these results as follows.

Propositon A.1 (The Frontier Algorithm). *Let M_0 satisfy $M_0 \geqslant c^*/(1-\delta)$, where the c^* is the maximum element in $\mathbf{c}$ and let $\mathbf{B}_0$ be an identity matrix.*

(i) *For all $j \geqslant 0$, the basis matrices $\mathbf{B}_j$ generated by the frontier algorithm are optimal for M in $[M_{j+1}, M_j]$. In this range, $\phi_i(x_i, M) = \boldsymbol{\lambda}_{cj}(x_i) + M\boldsymbol{\lambda}_{dj}(x_i)$ and $\phi_i'(x_i, M) = \boldsymbol{\lambda}_{dj}(x_i)$.*

(ii) *For any state x_i with a positive Gittins index, the Gittins index $M_i^*(x_i)$ is equal to $\mathbf{g}(x_i)$.*

(iii) *The algorithm will terminate after a finite number of steps.*

Proof. See the online appendix. □

Thus, with one pass of the frontier algorithm, we can calculate the cluster-specific value function and its derivatives for all states and retirement values $M > 0$ and Gittins indices for all states with nonnegative indices. Given the slopes and breakpoints for all clusters in a given state $\mathbf{x}$, we can easily (and quickly) calculate the Whittle integral $\hat{\Phi}(\mathbf{x}, 0)$; we provide details of these calculations in the online appendix.

A.3. Discussion of the Frontier Algorithm

This frontier algorithm differs from the Gass-Saaty algorithm for parametric linear programming in several ways. First, we are able to use properties of the bandit problem to identify an initial basis that is optimal for large values of M, rather than starting the procedure by solving a linear program for some value of M.

Second, we do "multiple pivots" in the event of a tie in the ratio test for choosing which variables to add to the basis: we swap in all elements of $\mathcal{I}$ achieving the maximum ratio, taking care to ensure that the basis consists of one action per state. Because the linear program represents a stochastic dynamic program, we can be sure that such a multiple pivot will result in a valid basis (i.e., a basis matrix with linearly independent columns). In our examples, ties in the ratio test are common and the sets $\mathcal{I}$ may be large (e.g., containing hundreds or thousands of elements); efficiency is greatly enhanced by doing multiple pivots simultaneously.

The final point of difference between the frontier algorithm and the Gass-Saaty algorithm is also a consequence of the fact that (18) represents a stochastic dynamic program. Because of this, we know that the basic feasible solutions will be nondegenerate and no cycling will occur. The Gass-Saaty algorithm requires additional assumptions to ensure that cycling will not occur (see Klee and Kleinschmidt 1990). Here we cannot be sure that M_j will decrease in each iteration, but we can be sure that the objective function without retirement rewards ($\mathbf{e}^\top \boldsymbol{\lambda}_{cj}$) will increase in each iteration.

In light of this last observation, we can view this frontier algorithm as a form of the policy iteration method (see, e.g., Puterman 1994) for solving the stochastic dynamic program (16) with zero retirement value. In each iteration, we modify the policy, ensuring that the objective function without retirement value improves. A full policy iteration method would proceed in the same way as our frontier algorithm, but would take the changing index set $\mathcal{I}$ to include all state-action pairs with positive reduced costs $\bar{\mathbf{c}}_k$: these basis changes represent the possible improvements in the current policy given the value function $\boldsymbol{\lambda}_{cj}$ associated with the current policy. (If there is more than one action with positive reduced cost for a given state, the convention is to select the action with the largest reduced cost for that state.) Policy iteration—in the full form or in the form of the frontier algorithm—continues until no reduced costs $\bar{\mathbf{c}}_k$ are positive, meaning the policy cannot be further improved.

What can we say about the number of iterations involved in the frontier algorithm? If a cluster satisfies the Whittle condition in a given state, the actions for that state can only change from retirement to the dominant action as we decrease the retirement value M; by Lemma 4.1(iii), the decisions will not switch back to retirement. Thus, if a cluster satisfies the Whittle condition in all states, there cannot be more iterations than there are states.

If the cluster does not satisfy the Whittle condition in some state, we would still never switch back to retirement as we decrease the retirement value, but the choice of action may change as we change the retirement value. We know the frontier algorithm will stop after a finite number of steps, but we cannot say much about how many steps will be required. Other researchers (see, e.g., Melekopoglou and Condon 1994) have shown that some simple policy iteration schemes have a worst-case number of iterations that is exponential in the number of state-action pairs. However, Ye (2011) has a recently shown that another form of simple policy iteration has a worst-case number of iterations that is strongly polynomial in the number of states and actions. Although we have not fully investigated the worst-case performance of the frontier algorithm, in our examples, the number of iterations required are substantially fewer than the number of states, even when the Whittle condition is not satisfied.

A.4. Application in the North Sea Example

The North Sea example has some nice features that can be exploited in the frontier algorithm; these features may be present in many other applications as well. First, we note that the constraint matrices for the linear programs (16) are quite sparse. When the action is to drill, there are just three possible next-period states and the corresponding row in the constraint matrix ($\mathbf{A}^\top$) has just four nonzero elements. If the action is to retire, the corresponding row will have a single nonzero element, a one on the diagonal. More generally, whenever we consider clusters that

are a combination of simpler "subclusters" (as discussed in §3.1 and §5.3), the constraint matrices will be sparse because only one subcluster will change state in any period.

A second key feature of the example is that we can order the states to ensure that the state index will be strictly increasing with each transition. States where a target is undrilled are given a low index and states where the target contains oil or gas or is dry are assigned higher indices: drilling then leads to a state with a higher index. If we order the states this way and ensure that the rows in the basis matrix correspond to this order of states, the basis matrices $\mathbf{B}_j$ will be lower triangular with ones along the diagonal. This simplifies the calculations significantly in that we can solve $\mathbf{B}_j^{\mathsf{T}}\boldsymbol{\lambda}_{cj} = \mathbf{c}_{\mathbf{B}}$ and $\mathbf{B}_j^{\mathsf{T}}\boldsymbol{\lambda}_{cj} = \mathbf{c}_{\mathbf{B}}$ for the dual variables $\boldsymbol{\lambda}_{cj}$ and $\boldsymbol{\lambda}_{dj}$ using forward substitution. For problems lacking this structure, we may want to maintain and update a LU decomposition of the basis matrix as we proceed through the algorithm, as is usually done in professional software for solving linear programs using the simplex method (see, e.g., Gill et al. 1987).

To get a sense of size of these problems, consider cluster 3 in Figure 2, which consists of six targets. The cluster-specific linear program (17) for this cluster has $4^6 = 4{,}096$ decision variables (one for each state) and the constraint matrix has 10,240 rows (one for each state-action pair) and 4,096 columns, but only 28,672 nonzero elements. The number of steps involved in the frontier algorithm will depend on the specific probabilities for the cluster. In the case where the probabilities are set to the marginal distribution for the cluster (with kitchen uncertainty), the frontier algorithm requires 174 iterations, performing 3,175 basis swaps in the process. The cluster-specific value function for the initial state has 16 pieces, as shown in Figure 4(a). The algorithm runs in approximately 0.08 seconds, in MATLAB on a personal computer.

The fact that the number of basis swaps in the frontier algorithm greatly exceeds the number of iterations indicates that ties in the ratio rule are common. In this example, the maximum number of swaps in one iteration was 781 (i.e., the largest set $\mathcal{J}$ contained 781 elements). These ties reflect the structure of the underlying probabilistic model. For example, once gas is discovered at a target, the DM knows that the associated prospect contains gas. Once a prospect is "proven" to contain gas, the results for other targets no longer affect the probability of finding oil or gas at any target associated with the proven prospect. Thus, when changing the retirement value M leads to a change in decision in a state where a prospect has been proven to contain gas, the same change may occur simultaneously in many other states.

The largest cluster that we consider is Cluster 3 in Figure 3, which consists of nine targets. Here, the cluster-specific linear program (17) has a constraint matrix with 851,968 rows (state-action pairs), 262,144 columns (state variables), and 2,621,440 nonzero elements. In the case with probabilities set to the marginal distribution for the cluster (with no kitchen uncertainty), the frontier algorithm requires 4,935 iterations and performs 229,301 basis swaps; the maximum number of swaps in one iteration involves 28,672 elements. In this case, the frontier algorithm takes about four minutes to run in MATLAB and produces a cluster-specific value function for the initial state that has 228 pieces.

Endnotes

1. The clusters in Figure 3 do not include targets 6C or 8A; in §6.3 we will argue that these targets should never be drilled and hence can be dropped from the model.

2. All run times are for a Dell desktop computer with a 3.07 GHz Intel Xeon CPU and 12.0 GB of RAM, running MATLAB 7.12.0 (2011a; 64-bit) in Windows 7 on a single processor.

3. In the North Sea example, the simulation horizon is finite as the DM will quit in finite time. We can simulate infinite horizon problems using the techniques described in Fox and Glynn (1989) that provide unbiased estimators in finite time by randomizing the stopping time.

4. The careful reader may note that we need not update the active cluster in this sequential process. The conditional probability distribution for the active cluster includes the relevant probabilities for all possible next-period states and the frontier algorithm produces recommended actions and Gittins indices for all states—including the next-period state—for a given cluster. However, we do need to update all inactive clusters as the probability distributions for the clusters may change based on the results observed for the active cluster; if these distributions change, the policies and Gittins indices for the inactive clusters may also change.

5. The matrix $\mathbf{P}$ is substochastic rather than stochastic because the retirement decisions are modeled as exiting the system (i.e., there is no "retired" state); the transition probabilities in $\mathbf{P}$ for rows corresponding to retirement decisions will all be zero.

Acknowledgments

The authors are grateful to Jo Eidsvik and Gabriele Martinelli for sparking interest in this problem, for sharing their model for the North Sea example, and for many helpful conversations. The authors are also very grateful for the helpful feedback provided by three anonymous referees and an associate editor.

References

Adelman D, Mersereau AJ (2008) Relaxations of weakly coupled stochastic dynamic programs. *Oper. Res.* 56(3):712–727.

Andersen L, Broadie M (2004) Primal–dual simulation algorithm for pricing multidimensional American options. *Management Sci.* 50(9):1222–1234.

Bertsekas D (1995) *Dynamic Programming and Optimal Control*, 2nd ed. (Athena Scientific, Belmont, MA).

Bickel JE, Smith JE (2006) Optimal sequential exploration: A binary learning model. *Decision Anal.* 3(1):16–32.

Bickel JE, Smith JE, Meyer JL (2008) Modeling dependence among geologic risks in sequential exploration decisions. *SPE Reservoir Evaluation & Engrg.* 3(4):233–251.

Brown DB, Smith JE (2011) Dynamic portfolio optimization with transaction costs: Heuristics and dual bounds. *Management Sci.* 57(10):1752–1770.

Brown DB, Smith JE, Sun P (2010) Information relaxations and duality in stochastic dynamic programs. *Oper. Res.* 58(4):785–801.

Chen YR, Katehakis MN (1986) Linear programming for finite state multi-armed bandit problems. *Math. Oper. Res.* 11(1):180–183.

Farias VF, Madan R (2011) The irrevocable multiarmed bandit problem. *Oper. Res.* 59(2):383–389.

Fox BL, Glynn PW (1989) Simulating discounted costs. *Management Sci.* 35(11):1297–1315.

Gass S, Saaty T (1955) The computational algorithm for the parametric objective function. *Naval Res. Logist. Quart.* 2(1–2):39–45.

Gill PE, Murray W, Saunders MA, Wright MH (1987) Maintaining LU factors of a general sparse matrix. *Linear Algebra and Its Appl.* 88/89:239–270.

Gittins J, Jones D (1974) A dynamic allocation index for the sequential allocation of experiments. Gani J, ed. *Progress in Statistics* (North Holland, Amsterdam), 241–266.

Gittins J, Glazebrook KD, Weber RR (2011) *Multi-Armed Bandit Allocation Indices*, 2nd ed. (John Wiley & Sons, Chichester, UK).

Glasserman P (2004) *Monte Carlo Methods in Financial Engineering* (Springer, New York).

Glazebrook KD (1982) On a sufficient condition for superprocesses due to Whittle. *J. Appl. Probab.* 19(1):99–110.

Haugh MB, Kogan L (2004) Pricing American options: A duality approach. *Oper. Res.* 52(2):258–270.

Hawkins J (2003) A Lagrangian decomposition approach to weakly coupled dynamic optimization problems and its applications. Ph.D. thesis, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA.

Henderson SG, Glynn PW (2002) Approximating martingales for variance reduction in Markov process simulation. *Math. Oper. Res.* 27(2):253–271.

Kallenberg LCM (1986) A note on M. N. Katehakis' and Y.-R. Chen's computation of the Gittins index. *Math. Oper. Res.* 11(1):184–186.

Klee V, Kleinschmidt P (1990) Geometry of the Gass-Saaty parametric cost LP algorithm. *Discrete Comput. Geometry* 5(1):13–26.

Lai G, Margot F, Secomandi N (2010) An approximate dynamic programming approach to benchmark practice-based heuristics for natural gas storage valuation. *Oper. Res.* 58(3):564–582.

Martinelli G, Eidsvik J, Hauge R (2012) Dynamic decision making for graphical models applied to oil exploration. Working paper, Norwegian University of Science and Technology, Trondheim, Norway.

Martinelli G, Eidsvik J, Hauge R, Førland MD (2011) Bayesian networks for prospect analysis in the North Sea. *AAPG Bull.* 95(8): 1423–1442.

Melekopoglou M, Condon A (1994) On the complexity of the policy improvement algorithm for Markov decision processes. *INFORMS J. Comput.* 6(2):188–192.

Mersereau A, Rusmevichientong P, Tsitsiklis J (2009) A structured multiarmed bandit problem and the greedy policy. *IEEE Trans. Automatic Control* 54(12):2787–2802.

Murphy K (2001) The Bayes net toolbox for MATLAB. *Comput. Sci. Statist.* 33:331–350.

Pandey S, Chakrabarti D, Agarwal D (2007) Multi-armed bandit problems with dependent arms. *Proc. 24th Internat. Conf. Machine Learn.*, ACM International Proceedings Series (ACM, New York), 721–728.

Puterman ML (1994) *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (John Wiley & Sons, New York).

Rogers LCG (2002) Monte Carlo valuation of American options. *Math. Finance* 12(3):271–286.

Rogers LCG (2007) Pathwise stochastic optimal control. *SIAM J. Control Optim.* 46(3):1116–1132.

Rusmevichientong P, Tsitsiklis JN (2010) Linearly parameterized bandits. *Math. Oper. Res.* 35(2):395–411.

Ryzhov IO, Powell WB, Frazier PI (2012) The knowledge gradient algorithm for a general class of online learning problems. *Oper. Res.* 60(1):180–195.

Wang C-C, Kulkarni SR, Poor HV (2005) Bandit problems with side observations. *IEEE Trans. Automatic Control* 50(3):338–355.

Whittle P (1980) Multi-armed bandits and the Gittins index. *J. Royal Statist. Soc. Ser. B* (*Methodological*) 42(2):143–149.

Whittle P (1988) Restless bandits: Activity allocation in a changing world. *J. Appl. Probab.* 25A:287–298.

Ye Y (2011) The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate. *Math. Oper. Res.* 36(4):593–603.

David B. Brown is an associate professor of decision sciences at the Fuqua School of Business, Duke University. His research focuses on developing methods for solving optimization problems involving uncertainty.

James E. Smith is J. B. Fuqua Professor of Business Administration at the Fuqua School of Business, Duke University. His research interests are primarily in decision analysis and focus on developing methods for formulating and solving dynamic decision problems. He is an INFORMS Fellow.