Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.

SIMULATION MODEL CALIBRATION WITH CORRELATED KNOWLEDGE-GRADIENTS

Peter Frazier
Warren B. Powell
Hugo P. Simão

Department of Operations Research & Financial Engineering
Princeton University
Engineering Quadrangle
Olden St. Princeton, N.J. 08544, U.S.A.

ABSTRACT

We address the problem of calibrating an approximate dynamic programming model, where we need to find a vector of parameters to produce the best fit of the model against historical data. The problem requires adaptively choosing the sequence of parameter settings on which to run the model, where each run of the model requires approximately twelve hours of CPU time and produces noisy non-stationary output. We describe an application of the knowledge-gradient algorithm with correlated beliefs to this problem and show that this algorithm finds a good parameter vector out of a population of one thousand with only three runs of the model.

1 INTRODUCTION

CASTLE Laboratory at Princeton University has undertaken a number of projects for which the models developed were required to match the historical performance of a company. These models use approximate dynamic programming (ADP) to optimize an objective function, but the use of optimization is motivated primarily by the desire to mimic tactical human decision-making for the purposes of strategic planning. In particular, the companies that sponsored the development of models would not accept the results until each model matched a series of performance metrics based on history.

Calibrating one of these models requires tuning a vector of parameters that often represent bonuses and penalties used to encourage specific behaviors. These are famously known as "soft costs" (or rewards), and they allow us to combine multiple objectives within a single optimization formulation. Thus, while we may wish to maximize profits (revenues minus costs), in reality we must balance short term profits against customer service, equipment productivity and goals for managing people.

The problem can be posed as an optimization problem. Let ρ be the vector of parameters that we need to tune. Let g_k, $k \in \mathscr{K}$, be target values for a set of metrics (on-time performance, equipment productivity, cost and revenue metrics), and let $G_k(\rho)$ be the average value of metric k produced by the model when run with parameter vector ρ. We assume that the parameter vector ρ resides within a space whose dimensionality is small enough to allow discretization. The values $G_k(\rho)$ are unknown and can only be observed with noise. We wish to find ρ that solves

$$\min_\rho F(\rho), \qquad F(\rho) = \sum_{k \in \mathscr{K}} (G_k(\rho) - g_k)^2. \tag{1}$$

The problem of tuning this vector of parameters introduces a type of simulation-optimization, where we have to find a set of parameters to minimize a weighted metric that evaluates how well a model matches historical performance. In this setting, our model is basically a black box that can be run to evaluate how well a set of parameters matches history. The challenge is that the models are slow. Optimizing the movements of thousands of drivers, locomotives or jets, despite extensive engineering, is not an easy process. These large-scale models can require hours to run in the best case, and may even require several days. In addition, the outcome of a run is noisy.

We use the context of a fleet simulator developed for Schneider National, one of the largest truckload motor carriers in the U.S., as our test environment. The model and its application is described in depth in (Simão et al. 2009). The model is expensive to run and its output is noisy, and so we view the problem as optimizing the inputs to an expensive and noisy black box simulator. Unlike classical simulation models, for which observations arrive from some distribution fixed in time,

ADP requires iterative learning and observations from ADP models undergo a transient initial period while the model learns and the policy stabilizes. Training until the model learns fully is very expensive, and so we would like the ability to perform simulation optimization using only these transient observations. When performing our simulation optimization, we would also like to use our intuition that the ADP model will behave similarly with similar parameter vectors. This paper provides a new method for performing this type of simulation optimization using the concept of the knowledge gradient. The knowledge gradient is a flexible Bayesian-based method that has the ability to learn with transient data, and can exploit correlations in the belief structure around neighboring parameters.

This problem represents an instance of classical simulation-optimization, for which an extensive literature has evolved. We review this literature in section 2. Section 3 provides an overview of the ADP model developed for Schneider, and a method we developed to project the performance of each metric using fewer ADP iterations. Section 4 describes the calibration process, and section 5 summarizes the knowledge-gradient algorithm and its adaptation to the simulation-optimization problem. Section 6 describes experimental results using the method to calibrate the simulation model for Schneider National, showing that we can find a good calibration vector from a population of a thousand with only six iterations. Section 7 concludes the paper.

2 LITERATURE REVIEW

There is an extensive literature that addresses the problem posed by (1). (Spall 2003) provides a thorough review of the stochastic search literature, whose roots can be traced to the seminal papers of (Robbins and Monro 1951) and (Kiefer and Wolfowitz 1952). (Spall 2003) describes a method called the Simultaneous Perturbation Stochastic Approximation method, which uses only two sample realizations to construct a derivative, but the rate of convergence is even slower than when derivatives are available (the value of the technique is a much lower cost per iteration). Response surface methods (Myers and Montgomery 2002) have been widely used in stochastic optimization for problems where derivatives are not available.

The ranking and selection literature (see, e.g., (Bechhofer et al. 1995)) considers a problem similar to (1), in which one must allocate an experimental budget to most accurately and efficiently discover which of several unrelated alternatives is the best. Recognizing that simulations can be expensive and need to be run efficiently, the simulation community has applied ideas from ranking and selection to the problem of simulation optimization (see (Swisher et al. 2003) for a survey). One Bayesian approach is the Optimal Computing Budget Allocation (OCBA) (Chen et al. 1996, Chen et al. 2000). In a separate Bayesian approach, (Gupta and Miescke 1996) proposes the idea of making a choice that maximizes the value of a single measurement. (Chick and Inoue 2001) then proposes a sequential method called $LL(S)$ (sequential selection with linear loss), which allocates a fixed number of replications among each of a set of competing alternatives at each stage until a budget is exhausted. The allocation is based on an approximation of the value of a measurement at each stage. These ranking and selection approaches do not use the structure of the objective function (1) to improve the convergence rate of the search technique.

Bayesian global optimization (Mockus 1989) considers the problem of global optimization when function evaluation is expensive. Within this body of research, (Jones et al. 1998) introduces the expected improvement method, which chooses measurements that maximize the one-step expected improvement in the best observed objective function value. These approaches are able to exploit an objective function's continuity to dramatically improve search efficiency over those ranking and selection methods that assume no relationship between objective function values at different points, but they have not been specialized to consider the special structure (e.g., non-negativity and prevalence of quadratic trends) of calibration objective functions of the form (1).

Knowledge-gradient (KG) policies are a general class of methods for making measurements in learning problems. These methods choose the measurement with the largest single-period value, and understand this value as the derivative of the value of knowledge with respect to that measurement. (Frazier et al. 2008) identified the policy used in (Gupta and Miescke 1996) as a knowledge-gradient policy and established its asymptotic optimality. (Frazier et al. 2009) derived the KG policy for a ranking and selection problem with finitely many alternatives when the beliefs about different alternatives are correlated. This is the KG policy most similar to the KG policy we propose here.

3 THE ADP MODEL

3.1 The Schneider National Business Simulator

The ADP model we consider assigns drivers to loads over time. The reader is referred to (Simão et al. 2009) for a complete description of the model, but we provide a sense of the structure of the system here. The challenge is to manage a fleet of almost 7,000 drivers, each of which is described by a complex vector of attributes that makes it possible to measure statistics such as how often a driver returns home, and the productivity of different driver types such as teams (drivers working in pairs) and solos. The company expected the model to produce performance metrics that closely match historical performance, which was accomplished using a number of tuning parameters in the form of bonuses and penalties to encourage/discourage certain behaviors. The challenge faced in this research is designing a method to tune these parameters.

Let $a \in \mathscr{A}$ be a vector of attributes of a driver (there were 15 dimensions to this vector for this application), and let R_{ta} be the number of drivers with attribute a. Let $d \in \mathscr{D}$ be the decision to assign a driver to a load, move a driver empty to another location, or to hold a driver in his current location. Let x_{tad} be the decision (0 or 1) to act on a driver with attribute a using decision d. If R_t is the resource state vector (the vector with elements R_{ta}) and x_t is the vector of decisions, we let

$$R_{t+1} = R^M(R_t, x_t) + \hat{R}_{t+1}$$

be the vector describing the state of drivers in the next time period given that the current resource state vector is R_t and we act on these resources using the decision vector x_t. $\hat{R}_{t+1}$ captures random (exogenous) changes to the resource state vector, which is used to capture travel delays, equipment failures, and random arrivals to and departures from the system. Let $\hat{D}_t$ be the new demands that were first realized at time t, giving us a state variable of $S_t = (R_t, \hat{D}_t)$. We assign drivers to loads to maximize a contribution function given by

$$C(S_t, x_t|\rho) = \sum_a \sum_d c_{tad}(\rho) x_{tad}.$$

$c_{tad}(\rho)$ captures the revenue from moving a load, the cost of moving a driver empty to a load, and a series of bonuses and penalties to discourage picking up a load early or late, delivering a load early or late, and putting drivers on loads of an appropriate length, and encouraging the model to get drivers home during targeted intervals (often corresponding to weekends roughly every two to three weeks). Schneider's fleet of drivers are organized into different groups with names such as *solo* (company employees who drive alone), *team* (company employees who drive in pairs, allowing one to sleep while the other drives), and *independent contractors* (IC's), who are not employees and as a result own their own tractors, which raises the operating cost since the driver has to use his wages to cover the cost of his equipment. Other fleet types include regional fleets, where drivers would move shorter loads but stay within a particular region of the country.

3.2 Designing an ADP Policy

An optimal policy for assigning drivers to loads can be characterized by

$$V_t(S_t) = \max_{x_t} \left(C(S_t, x_t|\rho) + \mathbb{E}\left[V_{t+1}(S_{t+1})\right] \right)$$

subject to

$$\sum_d x_{tad} = R_{ta}, \quad a \in \mathscr{A},$$
$$x_{tad} \geq 0, \quad a \in \mathscr{A},\ d \in \mathscr{D}.$$

The state S_{t+1} is a function of the starting state S_t, the action x_t and the new demands $\hat{D}_{t+1}$ that are called in before time $t+1$. R_t is a vector, and has a very large number of dimensions because it captures the number of drivers with each attribute

vector a. Good policies can be found using ADP, in which we solve

$$x_t^n = \arg\max_x \left(C(S_t^n, x|\rho) + \sum_{a \in \mathscr{A}} \bar{v}_{ta}^{n-1} R_{ta}^x \right), \tag{2}$$

where R_{ta}^x is an element of $R_t^x = R^M(R_t^n, x)$, which is the status of all the drivers after we make a decision, but before any new information arrives (in the form of $\hat{R}_{t+1}$).

In (2), we are approximating the expectation of the value function using a value function approximation that is linear in the post-decision resource state variable R_t^x. These are estimated using

$$\bar{v}_{t-1,a}^n = (1 - \alpha_{n-1}) \bar{v}_{t-1,a}^{n-1} + \alpha_{n-1} \hat{v}_{ta}^n,$$

where α_{n-1} is a step-size between 0 and 1.

Estimating the slopes $\bar{v}_t^n$ requires stepping forward through time over a horizon (typically a month), where we have to perform roughly 100 iterations to get reasonable estimates for the slopes. At each time period, the optimization problem in equation (2) involves assigning several thousand drivers to several thousand loads. Stepping through a month of activities requires several minutes. Repeating this 100 times requires many hours of CPU time on 3GHz Pentium-class processors.

4 THE CALIBRATION PROBLEM

The success of the model was judged in terms of its ability to match a series of metrics. As described in the introduction, the model produces estimates of how many time-at-home (TAH) events a driver would incur during a month, the average length of haul for each fleet type, and other metrics such as equipment productivity and customer service. Getting the model to calibrate against historical performance requires tuning the calibration vector ρ to strike the right balance. For example, we might outperform history for equipment productivity, but find that we were not getting drivers home. To obtain a realistic model, it is important to avoid underperforming the company in any one dimension.

In our initial model development, the parameter vector ρ was tuned by hand. This was a painstaking process because of the long run times. Furthermore, as the years passed, the data would change as would the performance of the company. As a result, re-tuning the model became an annual exercise that could easily take two weeks of manual experimentation. The goal of this research is to automate the process, while minimizing the time required to find good solutions to the parameter vector.

As a preliminary to discussion the automation of the calibration process, Sections 4.1 and 4.2 discuss estimation of the quality of calibration for a particular value of ρ with a relatively small number of iterations from the ADP model.

4.1 Observing Model Output

In this section we fix ρ and k and discuss how $G_k(\rho)$ may be estimated from the output of the ADP model, where we recall that $G_k(\rho)$ is defined to be the long-run average of output k when we use input parameter vector ρ. This output is non-stationary, and if we examine its value over the iterations we typically see it moving randomly around a mean level that converges exponentially toward $G_k(\rho)$. Figure 1, which shows 200 iterations of the TAH metric for solo company drivers, is typical.

A simple but inefficient way to estimate $G_k(\rho)$ is to average the output over a very large number of iterations. Including early iterations in the average biases the estimate, either below $G_k(\rho)$ as it would for the sample path given in Figure 1, or above in those cases in which the output trends downward. As the number of iterations becomes large, this bias eventually becomes small, but many iterations are required.

A slightly better but still inefficient way to calculate $G_k(\rho)$ would be to run the ADP model until it appears close to convergence, which might be said to occur in Figure 1 near iteration 100, and then average only the output obtained afterward. One problem with this method is that the ADP model may not converge perfectly, even after a large number of iterations, instead growing only infinitessimally closer to convergence. This induces a small bias in the estimate. A larger problem is that this method is very time-consuming, since running the model to apparent convergence generally requires two full days of computer time. Furthermore, it does not offer a quick way of obtaining a rough estimate from just a few iterations. The ability to obtain rough estimates early is particularly important to calibration because it allows us to focus subsequent search effort on those input vectors that are likely to calibrate well.

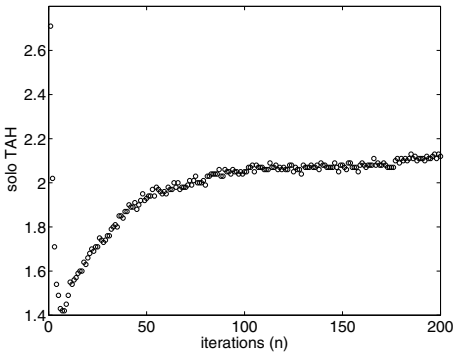


Figure 1: Sample path showing of the time-at-home (TAH) metric for solo company drivers over 200 ADP training iterations

We have designed a method that provides rough estimates early and at the same time provides later estimates that are at least as accurate as post-apparent-convergence averages. It operates by observing the currently available output, which may be from a relatively small number of iterations, estimates how quickly the output is converging to $G_k(\rho)$, and then uses this knowledge of the convergence rate to estimate $G_k(\rho)$.

Our method uses the following statistical model for output k as a function of the iteration n:

$$Y_k^n(\rho) = B_k(\rho) + [G_k(\rho) - B_k(\rho)]\,[1 - \exp(-nR_k(\rho))] + \varepsilon^n, \quad n > n_0. \tag{3}$$

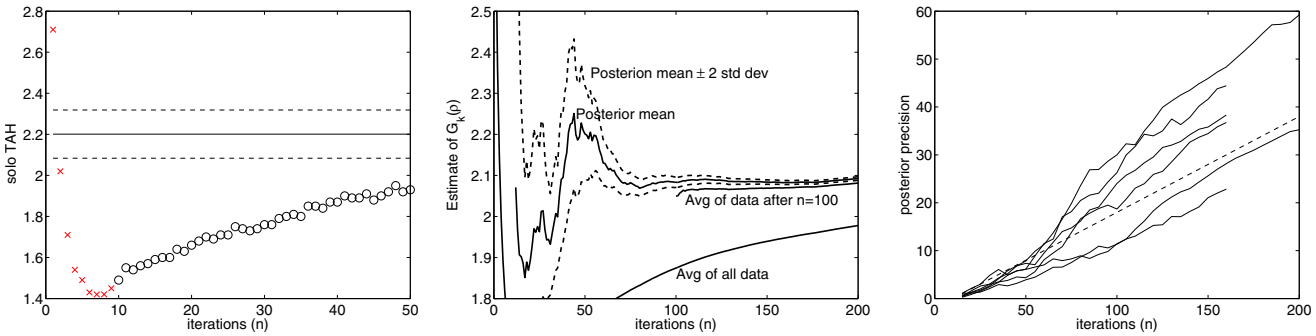
This model contains a number of newly defined quantities: $Y_k^n(\rho)$ is the observation from the ADP model of output type k at iteration n when the parameter vector is ρ; $G_k(\rho)$ was defined earlier and is the limiting value to which this output converges; $R_k(\rho)$ is the rate at which the output converges to its limiting value; n_0 is a threshold usually set to 10 that allows the model to ignore the erratic initial output we see in Figure 1 and in general; $B_k(\rho)$ is a parameter of the model that would be the expected value of the output at the first iteration if not for the erratic initial behavior; and ε^n is an independent unbiased normal random variable with variance σ_ε^2 (which is also new notation) that encapsulates the randomness in the observation process.

This model assumes geometric convergence in n of the mean value of the ADP output. Value iteration is known to converge geometrically (Puterman 1994), and the ADP model from which we are collecting data is using approximate value iteration. Note that the model assumes geometric convergence of the *mean* value of the ADP output, and does not assume geometric convergence of the *estimates* of the value functions maintained within the ADP policy.

Using this statistical model, we employ a Bayesian analysis to estimate $G_k(\rho)$ from the observations $Y_k^1(\rho),\ldots,Y_k^n(\rho)$. We place a prior distribution on the parameters $G_k(\rho)$, $R_k(\rho)$, $B_k(\rho)$, and σ_ε^2, and then obtain a posterior distribution conditioned on the data. Conditioned on $R_k(\rho)$, the prior is the standard noninformative prior for Bayesian linear regression (Gelman et al. 2004), which is uniform on $G_k(\rho), B_k(\rho), \log\sigma_\varepsilon$ over $\mathbb{R}\times\mathbb{R}\times\mathbb{R}_+$. The marginal prior on $R_k(\rho)$ is concentrated on a fixed set of points $\mathscr{R}$.

The marginal prior on $R_k(\rho)$ was obtained from a smoothed version of the empirical distribution of $R_k(\rho)$ calculated using an older dataset. This dataset resulted from Schneider's operations in 2007 and the empirical distribution was obtained by calculating maximum likelihood estimates of $R_k(\rho)$ from data outputted by runs of the ADP model for several different values of k and ρ. 200 iterations were generated by each run of the ADP model used, which is large enough to produce a reasonable expectation of accuracy in the maximum likelihood estimates. We then smoothed this empirical distribution by fitting a normal distribution to it, and chose a finite set of points $\mathscr{R} = \{1/100,\ldots,1/2\}$ that covered the main mass of the fitted distribution. We chose to distribute the inverse of the rates uniformly in $\mathscr{R}$ because the sensitivity of the model to the rate is most pronounced when $R_k(\rho)$ is close to 0, making the difference between rates $1/100$ and $1/99$ approximately as important as the difference between rates of $1/2$ and $1/4$. Although obtaining the empirical distribution required a large amount of computer time to run the ADP model for many iterations with different values of ρ, this one-time investment allows us to calibrate the model under any future dataset, including the one from 2008 calibrated in Section 6.

Given this prior, the mean and variance of the posterior on $G_k(\rho)$ is calculated as follows. For each rate $r \in \mathscr{R}$, we first condition on $R_k(\rho) = r$. Define $\overline{G}_k(\rho,n,r) = \mathbb{E}\left[G_k(\rho) \mid (Y_k^j(\rho))_{j=0}^n, R_k(\rho) = r\right]$ to be the conditional posterior mean of $G_k(\rho)$, $\left(\sigma_k^G(\rho,n,r)\right)^2 = \mathrm{Var}\left[G_k(\rho) \mid (Y_k^j(\rho))_{j=0}^n, R_k(\rho) = r\right]$ to be the conditional posterior variance, and L_r to be the



(a) (b) (c)

Figure 2: Figure 2(a) shows the posterior mean and standard deviation resulting from a Bayesian analysis of the first 50 of the iterations in Figure 1. Figure 2(b) displays estimates from the three different estimation methods discussed in the text as a function of how many iterations the method observes. In Figure 2(c), the solid lines show precision of the posterior on $G_k(\rho)$ as a function of the number of iterations for several different values of k and ρ. The dotted line shows a linear fit of $0.2\,(n-n_0)$, where $n_0 = 10$.

conditional likelihood of $(Y_k^l(\rho))_{j=0}^n$ given $R_k(\rho) = r$. Since the model (3) with $R_k(\rho) = r$ is linear in the parameters, these quantities may all be computed using techniques from Bayesian linear regression (see, e.g., (Gelman et al. 2004)).

Bayes' rule implies that the posterior probability of $R_k(\rho)$ being equal to r is given by $p_r = L_r / \sum_{r' \in \mathscr{R}} L_{r'}$. Using this, we define notation for and compute the conditional mean and variance of $G_k(\rho)$ as

$$\overline{G}_k(\rho,n) = \mathbb{E}\left[G_k(\rho) \mid (Y_k^j(\rho))_{j=0}^n\right] = \sum_{r\in\mathscr{R}} p_r \overline{G}_k(\rho,n,r),$$
$$\left(\sigma_k^G(\rho,n)\right)^2 = \mathrm{Var}\left[G_k(\rho) \mid (Y_k^j(\rho))_{j=0}^n\right] = \sum_{r\in\mathscr{R}} p_r \left(\left(\sigma_k^G(\rho,n,r)\right)^2 + \left(\overline{G}_k(\rho,n,r) - \overline{G}_k(\rho,n)\right)^2\right).$$

The expression for $\overline{G}_k(\rho,n)$ is derived using the tower property, and the expression for $\left(\sigma_k^G(\rho,n)\right)^2$ using the conditional variance formula.

Figure 2(a) shows the result of this Bayesian analysis on one particular sequence of 50 iterations of the TAH solo company driver metric with a fixed value of ρ. These iterations are the first 50 from figure 1. Both the crosses and circles are observations Y_k^n from the model, but the red crosses were discarded because they occurred before the threshold n_0. The solid line is the mean of the posterior, $\overline{G}_k(\rho,50)$, and the distance from the solid line to the dotted lines above and below is twice the standard deviation of the posterior, $2\sigma_k^G(\rho,50)$.

Figure 2(b) shows the three different estimates discussed as a function of how many iterations have been observed. The standard deviation of the posterior is also displayed. We see that the Bayesian method produces rough estimates early whose error is well-quantified by the variance of the posterior, and that even after a large number of iterations its estimate is more accurate than the post-apparent-convergence average. In particular, both averages are biased below $G_k(\rho)$.

Later, we will need to predict the posterior variance of $G_k(\rho)$ that will result from observing more iterations. Equivalently, we may predict the posterior precision, since the precision is the inverse of the variance. In Figure 2(c) we plot the posterior precision of $G_k(\rho)$ as a function of the number of iterations observed. We plot this quantity for several different values of ρ for the TAH metric for both solo company drivers and independent contractors. The relationship between posterior precision and the number of iterations n is roughly linear in n, with the posterior precision approximated by $0.2\,(n-n_0)$. Thus, after observing n iterations, we then approximate the posterior variance that would result from observing ℓ more iterations by

$$\left(\sigma_k^G(\rho,n+\ell)\right)^2 \approx \left(\widehat{\sigma}_k^G(\rho,n,\ell)\right)^2 = \left[\left(\sigma_k^G(\rho,n)\right)^{-2} + 0.2\,\ell\right]^{-1}. \tag{4}$$

4.2 Observing Calibration Quality

In the previous section, we fixed k and ρ and calculated the posterior distribution on $G_k(\rho)$ given observations from the ADP model at parameter vector ρ. We continue with fixed ρ and now consider the posterior distribution on $F(\rho) = \sum_k (G_k(\rho) - g_k)^2$ given this same set of observations. We later combine this posterior distribution with posterior distributions on $F(\rho')$ at values of $\rho' \neq \rho$ to obtain a composite posterior distribution on the overall function F.

Given the independent posterior distributions on $G_k(\rho)$ for $k \in \mathscr{K}$ computed in Section 4.1, and the definition (1) of $F(\rho)$, we compute and define notation for the mean and variance of $F(\rho)$ as

$$\overline{F}(\rho,n) = \mathbb{E}\left[F(\rho) \mid (Y_k^j(\rho))_{j=0}^n, k \in \mathscr{K}\right] = \sum_{k\in\mathscr{K}} \left(\overline{G}_k(\rho,n) - g_k\right)^2 + \left(\sigma_k^G(\rho,n)\right)^2, \tag{5}$$

$$(\sigma(\rho,n))^2 = \mathrm{Var}\left[F(\rho) \mid (Y_k^j(\rho))_{j=0}^n, k \in \mathscr{K}\right] \approx (\widehat{\sigma}(\rho,n))^2 = \sum_{k\in\mathscr{K}} 4\left(\sigma_k^G(\rho,n)\right)^2 \left(\overline{G}_k(\rho,n) - g_k\right)^2 + 2\left(\sigma_k^G(\rho,n)\right)^4. \tag{6}$$

To obtain the approximation $(\widehat{\sigma}(\rho,n))^2$ of the variance, we approximated the iteration-n posterior distribution on $G_k(\rho)$ by a normal distribution with mean $\overline{G}^k(\rho,n)$ and variance $\left(\sigma_k^G(\rho,n)\right)^2$, and then used the definition (1) of $F(\rho)$ together with the moments of the normal distribution.

Later, we will need an estimate of $\sigma(\rho,n+\ell)$ after having observed only n observations, where ℓ is any strictly positive integer. We approximate $(\sigma(\rho,n+\ell))^2$ by

$$(\sigma(\rho,n+\ell))^2 \approx (\widehat{\sigma}(\rho,n,\ell))^2 = \sum_{k\in\mathscr{K}} 4\left(\widehat{\sigma}_k^G(\rho,n,\ell)\right)^2 (\overline{G}_k(\rho,n) - g_k)^2 + 2\left(\widehat{\sigma}_k^G(\rho,n,\ell)\right)^4, \tag{7}$$

where we have used (6) but substituted $\overline{G}_k(\rho,n)$ for $\overline{G}_k(\rho,n+\ell)$ and $\widehat{\sigma}_k^G(\rho,n,\ell)$ for $\sigma_k^G(\rho,n+\ell)$.

5 THE KNOWLEDGE GRADIENT

Our strategy uses the KG algorithm to determine how to sequence the testing of different values of ρ. This KG method is similar to the KG method for correlated beliefs introduced in (Frazier et al. 2009), with the difference residing in the ability of the KG algorithm proposed here to use estimates based on transient behavior.

Note that we work with a discretized parameter space. If the parameter space has relatively few dimensions, say 5 or less, then this discretization presents no difficulty. If, on the other hand, the parameter space has a large number of dimensions, then the number of points into which we must discretize the space grows quite large and we may face computational challenges in computing the KG algorithm. This should be considered before using the KG algorithm in particular calibration problems.

We begin in Section 5.1 by discussing how the posteriors calculated for individual values of ρ in Sections 4.2 may be combined to give a cohesive posterior across all values of ρ. Then, in Section 5, we discuss how the decisions of the KG method are computed.

5.1 The Posterior Distribution on Calibration Policy

In Section 4.2 we saw how a posterior distribution on $F(\rho)$ for a fixed value of ρ, can be obtained from the observations of the model's output at that value of ρ. In this section, we approximate this posterior as normal and treat it as an observation of $F(\rho)$. The value of this so-called observation will be the mean of the posterior $\overline{F}(\rho,n)$ and the sampling variance with which it will have been assumed to have been taken will be the approximate variance of the posterior $(\widehat{\sigma}(\rho,n))^2$.

We begin by letting $\mathscr{I}$ be a set of discrete parameter vectors, which we obtained for this study by discretizing the time-at-home bonuses for solo company drivers and independent contractors. We would like to represent functions with domain $\mathscr{I}$ by vectors in $\mathbb{R}^{|\mathscr{I}|}$. Toward this end, we create a mapping $i : \mathscr{I} \mapsto \{1,\ldots,|\mathscr{I}|\}$ that gives each ρ in $\mathscr{I}$ a unique index. We then define a vector $\vec{F} \in \mathbb{R}^{|\mathscr{I}|}$ by letting $\vec{F}_{i(\rho)} = F(\rho)$ for each $\rho \in \mathscr{I}$. This vector $\vec{F}$ is a discretized version of the function F. We also let $e(\rho)$ be a column vector of 0's with a 1 in component $i(\rho)$.

We then take a multivariate normal prior on $\vec{F}$ with a power exponential covariance matrix (Cressie(1993)). The parameters of the power exponential covariance matrix were tuned to match the number of modes expected in the function F. We define μ^0 to be the mean vector and Σ^0 to be the covariance matrix of this prior. Using this prior, which is a

discrete-domain equivalent of a Gaussian process prior often used within spatial statistics (see, e.g., (Cressie(1993))), we can combine these observations to get a cohesive posterior on $\vec{F}$ across all values $\rho \in \mathscr{I}$.

We index by m the time spent running the ADP model at all values of ρ. Note that this time is different from the time in the stochastic optimization problem being solved, which is indexed by t. For each $\rho \in \mathscr{I}$, let $n(\rho,m)$ be the number of ADP iterations we have observed so far by time m at parameter vector ρ. Using the KG algorithm (described in Section 5.2), we choose from which $\rho \in \mathscr{I}$ to obtain ADP iterations. We give the name ρ^m to the parameter vector chosen. We then obtain ℓ additional iterations from the model at ρ^m, noting that $n(\rho^m,m) = 0$ if we have not yet measured ρ by time m. We have $n(\rho^m,m+1) = n(\rho^m,m)+\ell$, and $n(\rho,m+1) = n(\rho,m)$ for $\rho \neq \rho^m$. In this study, we set ℓ to 25 iterations, which takes about 12 hours of computer time, , which is large enough to amortize the cost of computing the KG decision over a reasonably large number of iterations, but small enough to retain the benefits of the algorithm's sequential nature.

Let $\mathscr{I}^m = \{\rho \in \mathscr{I} : n(\rho,m) > 0\}$ be the set of unique parameter vectors that we have chosen to observe by time m. Then, defining notation $\overline{F}^m(\rho) = \overline{F}(\rho,n(\rho,m))$ and $\widehat{\sigma}^m(\rho) = \widehat{\sigma}(\rho,n(\rho,m))$, we introduce the following approximation. In this approximation, the mean used is the mean of the actual posterior, while the variance used is an estimate.

Approximation 1. *The posterior* $F(\rho) \mid Y_k^j(\rho), j \leq n(\rho,m), k \in \mathscr{K}$ *is normal with mean* $\overline{F}^m(\rho)$ *and variance* $(\widehat{\sigma}^m(\rho))^2$.

Under Approximation 1, from (Gelman et al. 2004), we have that the posterior on $\vec{F}$ given all of the observations so far is multivariate normal with mean vector μ^m and covariance matrix Σ^m, where these can be computed as

$$\mu^m = \Sigma^m \left[(\Sigma^0)^{-1}\mu^0 + \sum_{\rho \in \mathscr{I}^m} (\widehat{\sigma}^m(\rho))^{-2} \overline{F}^m(\rho) e(\rho) \right], \qquad \Sigma^m = \left[(\Sigma^0)^{-1} + \sum_{\rho \in \mathscr{I}^m} (\widehat{\sigma}^m(\rho))^{-2} e(\rho) e(\rho)' \right]^{-1}. \tag{8}$$

5.2 THE KNOWLEDGE GRADIENT POLICY

The KG algorithm is a rule for choosing the next input vector ρ^m with which to run the ADP model given the observations up to the current time m. This rule is

$$\rho^m \in \arg\max_{\rho \in \mathscr{I}} \nu_\rho^{KG}(S^m),$$

where $S^m = (\mu^m, \Sigma^m)$ is called the current state of knowledge and encapsulates all we need to know from the observations so far, and $\nu_\rho^{KG}(S^m)$ is the value of collecting more iterations from the ADP model with input parameter vector ρ given the current state of knowledge. $\nu_\rho^{KG}(S^m)$ is called the *knowledge-gradient (KG) factor* at ρ, and is defined by

$$\nu_\rho^{KG}(S^m) = \mathbb{E}_m \left[\max_j (-\mu_j^{m+1}) | \rho^m = \rho \right] - \max_j (-\mu_j^m). \tag{9}$$

We see from this definition that the KG factor is the expected difference between the value of the best option that we can select given the current state of knowledge, which is $\max_j(-\mu_j^m)$, and the value of the best option that we can select given the new knowledge that would result from observing more iterations at ρ, which is $\max_j(-\mu_j^{m+1})$.

To compute or approximate the KG factor, we must must first determine the time-m conditional distribution of μ^{m+1}. Toward this end, we formalize two approximations and then state a proposition that uses them.

Approximation 2. $\widehat{\sigma}^{m+1}(\rho^m)$ *is equal to its time-m estimate* $\widehat{\sigma}^{m,1}(\rho^m) = \widehat{\sigma}(\rho^m, n(\rho^m,m), \ell)$.

Approximation 3. *The time-m conditional distribution of* $\overline{F}^{m+1}(\rho^m)$ *is normal.*

Proposition 1. *Under Approximations 1 and 2, the conditional variance* $\mathrm{Var}_m\left[\mu^{m+1}\right]$ *is given by* $\widetilde{\sigma}^m(\rho^m)\widetilde{\sigma}^m(\rho^m)'$, *where*

$$\widetilde{\sigma}^m(\rho^m) = \frac{\sqrt{\lambda^m + (\widehat{\sigma}^m(\rho))^2}}{\lambda^m + e(\rho)'\Sigma^m e(\rho)} \left[\Sigma^m e(\rho)\right], \qquad \lambda^m = \left[\left(\widehat{\sigma}^{m,1}(\rho^m)^2\right)^{-1} - \widehat{\sigma}^m(\rho^m)^{-2} \right]^{-1}.$$

If we additionally assume Approximation 3, then the time-m conditional distribution of μ^{m+1} *is multivariate normal with mean vector* μ^m *and covariance matrix* $\widetilde{\sigma}^m(\rho^m)\widetilde{\sigma}^m(\rho^m)$.

Proof. We begin by noting that (8) implies

$$\mu^{m+1} = \Sigma^{m+1}\left[(\Sigma^0)^{-1}\mu^0 + \textstyle\sum_{\rho\in\mathscr{I}^{m+1}}(\widehat{\sigma}^{m+1}(\rho))^{-2}\overline{F}^{m+1}(\rho)e(\rho)\right]. \tag{10}$$

To calculate the conditional variance, we note that $\overline{F}^m(\rho) = \overline{F}^{m+1}(\rho)$ and $\widehat{\sigma}^m(\rho) = \widehat{\sigma}^{m+1}(\rho)$ for all $\rho \neq \rho^m$, and so all terms except the ρ^m term in the sum from (10) do not affect the variance. Thus, $\mathrm{Var}_m\left[\mu^{m+1}\right] = \mathrm{Var}_m\left[\Sigma^{m+1}(\widehat{\sigma}^{m+1}(\rho^m))^{-2}\overline{F}^{m+1}(\rho^m)e(\rho^m)\right]$. Under Approximation 2, $\widehat{\sigma}^{m+1}(\rho^m) = \widehat{\sigma}^{m,1}(\rho^m)$ is known at time m, and thus so is Σ^{m+1}. Thus we can square them and move them outside the variance operator, providing the expression,

$$\mathrm{Var}_m\left[\mu^{m+1}\right] = \left[\Sigma^{m+1}e(\rho^m)\right]\left[\Sigma^{m+1}e(\rho^m)\right]'(\widehat{\sigma}^{m,1}(\rho^m))^{-4}\mathrm{Var}_m\left[\overline{F}^{m+1}(\rho^m)\right]. \tag{11}$$

We first consider the term $(\widehat{\sigma}^{m,1}(\rho^m))^{-4}\mathrm{Var}_m\left[\overline{F}^{m+1}(\rho^m)\right]$. The conditional variance formula and Approximation 2 imply $\mathrm{Var}_m\left[\overline{F}^{m+1}\right] = (\widehat{\sigma}^m(\rho^m))^2 - (\widehat{\sigma}^{m,1}(\rho^m))^2$. Substituting $(\lambda^m)^{-1} + (\widehat{\sigma}^m(\rho^m))^{-2}$ for $(\widehat{\sigma}^{m,1}(\rho^m))^{-2}$ and simplifying implies $(\widehat{\sigma}^{m,1}(\rho^m))^{-4}\mathrm{Var}_m\left[\overline{F}^{m+1}(\rho^m)\right] = (\lambda^m)^{-1}\left[1 + (\lambda^m)^{-1}(\widehat{\sigma}^m(\rho^m))^2\right]$.

We now consider the term $\Sigma^{m+1}e(\rho^m)$ in (11). From (8), Approximation 2, and the Sherman-Morrison-Woodbury matrix identity (Golub and Loan 1996), we have,

$$\begin{aligned}\Sigma^{m+1} &= \left[(\Sigma^m)^{-1} + \left[(\widehat{\sigma}^{m,1}(\rho^m))^{-2} - (\widehat{\sigma}^m(\rho^m))^{-2}\right]e(\rho^m)e(\rho^m)'\right]^{-1} = \left[(\Sigma^m)^{-1} + (\lambda^m)^{-1}e(\rho^m)e(\rho^m)'\right]^{-1} \\ &= \Sigma^m - \left[\lambda^m + e(\rho^m)'\Sigma^m e(\rho^m)\right]^{-1}\left[\Sigma^m e(\rho^m)\right]\left[\Sigma^m e(\rho^m)\right]',\end{aligned}$$

where we understand $(\widehat{\sigma}^m(\rho^m))^{-2}$ to be equal to 0 if we have not measured ρ^m before time m. Multiplying by $e(\rho^m)$,

$$\Sigma^{m+1}e(\rho^m) = \left[\Sigma^m e(\rho^m)\right]\left[1 - \frac{e(\rho^m)'\Sigma^m e(\rho^m)}{\lambda^m + e(\rho^m)'\Sigma^m e(\rho^m)}\right] = \Sigma^m e(\rho^m)\frac{\lambda^m}{\lambda^m + e(\rho^m)'\Sigma^m e(\rho^m)}.$$

Finally, combining these two expressions back into (11), we have

$$\mathrm{Var}_m\left[\mu^{m+1}\right] = \left[\Sigma^m e(\rho^m)\right]\left[\Sigma^m e(\rho^m)\right]'\frac{\lambda^m + (\widehat{\sigma}^m(\rho))^2}{(\lambda^m + e(\rho)'\Sigma^m e(\rho))^2} = \widetilde{\sigma}^m(\rho^m)\widetilde{\sigma}^m(\rho^m)'.$$

This shows the statement about $\mathrm{Var}_m\left[\mu^{m+1}\right]$. To show the statement about the conditional normality of μ^{m+1}, we assume Approximation 3 and note that (10) implies μ^{m+1} is an affine function of a time-m conditionally normal random vector $\overline{F}^{m+1}$, and is thus itself conditionally multivariate normal. □

By Proposition 1, if we let Z^{m+1} be an independent scalar normal random variable, μ^{m+1} is approximately equal in distribution to $\mu^m + \widetilde{\sigma}^m(\rho^m)Z^{m+1}$. Thus the KG factor may be approximated by

$$\nu^{KG}_\rho(S^m) \approx \mathbb{E}_m\left[\max_{\rho\in\mathscr{I}}(-\mu^m_{i(\rho)}) + \widetilde{\sigma}^m(\rho^m)Z^{m+1} \mid \rho^m = \rho\right] = h(-\mu^m, \widetilde{\sigma}^m(\rho)), \tag{12}$$

where the function h is defined by $h(a,b) = \mathbb{E}\max_j a_j + b_j Z^{m+1}$ for generic $|\mathscr{I}|$-dimensional vectors a and b. (Frazier et al. 2009) gives a method for computing $h(a,b)$. We state the method for the special case in which, for each j, there is a strictly positive probability that $\arg\max_j a_j + b_j Z^{m+1}$ is uniquely equal to j. The more general computation is performed by first reducing a and b to smaller vectors with this property, and for further details we refer to (Frazier et al. 2009). To compute $h(a,b)$ in the special case, we first sort the indices of the vectors a and b so that $b_1 < b_2 < \ldots$ and the lines defined by $z \mapsto a_j + b_j z$ are sequenced in order of increasing slope. Then, the two lines $a_j + b_j z$ and $a_{j+1} + b_{j+1}z$ intersect when z is equal to $c_j = \left[a_j - a_{j+1}\right]\left[b_{j+1} - b_j\right]$. We then compute (12) using

$$h(a,b) = \textstyle\sum_{j=1}^{|\mathscr{I}|} a_j\left(\Phi(c_j) - \Phi(c_{j-1})\right) + b_j\left(\varphi(c_{j-1}) - \varphi(c_j)\right).$$

6 EXPERIMENTAL RESULTS

We demonstrate the performance of the KG algorithm by using it to calibrate the ADP model using a dataset recorded from Schneider's operations during 2008. We calibrated two of the more sensitive calibration parameters, which were the bonuses awarded within the model for routing two different types of drivers to their homes. These driver types were solo company drivers and independent contractors.

The progress of the algorithm is shown in Figure 3, with times $m = 3,4,5,6$ in the rows from top to bottom, and the columns representing the mean of our posterior belief on fit deviation, $\mu^m_{i(\rho)}$ (left column), the standard deviation of this belief, $\sqrt{\Sigma_{i(\rho),i(\rho)}}^m$ (center column), and the KG factor ν^m_ρ (right column). Each of the twelve contour plots in this figure put the bonus awarded to solo company drivers along the horizontal axis labeled "SOLO bonus", and the bonus awarded to independent contractors along the vertical axis labeled "IC bonus". Previous measurements are marked with a blue circle, while the next measurement to take, which is also the maximum of the KG factors, is marked with a red circle in the KG factor plots. In the mean plots, the point with the smallest mean is marked with a star. We see in these plots that the KG algorithm strikes a balance between measuring points that have good means, since these are the points about which it is most valuable to have accurate estimates, and measuring points with large standard deviations, since these are the points for which are current estimates are poor.

In Figure 4, which plots $\log_{10} \min_i \mu^m_i$ as a function of m, we see the progress of the algorithm over time. After only three iterations our estimated fit deviation is less than $10^{-1.5}$, which compares well with the calibration quality that Schneider is able to achieve when calibrating by hand, and would be acceptable according to current practice. After six iterations, estimated fit deviation is close to 10^{-2}, and we would likely stop the calibration. These six iterations took 3 days of computer time, compared to the 1 to 2 weeks of computer and employee time that calibration requires when done by hand.

7 CONCLUSION

The major outcome of this work is the automation of a task that previously used ad hoc judgment and significant amounts of time from an analyst. In the tests performed, the KG method calibrated the model in approximately 1.5 days, compared to $7-14$ days when calibrated by hand. This automation decreases cost and improves speed while maintaining quality.

The power of the KG algorithm in this setting arises from the use of the correlation structure in the beliefs about the quality of different calibration vectors. One significant limitation of the procedure is the need to discretize the parameter space. We are currently working on a version of the KG algorithm that can handle continuous parameters without the need to discretize. This approach may alleviate this problem. Despite these limitations, the KG algorithm promises to improve the efficiency of calibration, and make it possible to easily calibrate expensive ADP models.

It remains to be seen how well the KG algorithm performs against other stochastic optimization methods in the calibration of ADP models. Generally speaking, Bayesian global optimization methods incur a greater computational cost per iteration than many other stochastic optimization methods in deciding where to sample next, but require fewer samples to find nearly optimal solutions. This often makes them appropriate when function evaluation is expensive, as it is in the case of ADP model calibration, while methods that do less computation per function evaluation (for example, stochastic approximation methods) are more appropriate when function evaluation is inexpensive. It would be valuable to use further numerical experiments to test whether this general tendency applies to the calibration of ADP models.

One open theoretical question concerns the convergence properties of the KG algorithm. KG algorithms for both of the problems presented in (Frazier et al. 2008) and (Frazier et al. 2009) are known to converge almost surely to global optima, and a wide class of KG algorithms are shown in (Frazier and Powell 2008) to also have this convergence property, and so it may be possible to apply these same proof methods to the KG algorithm presented here.

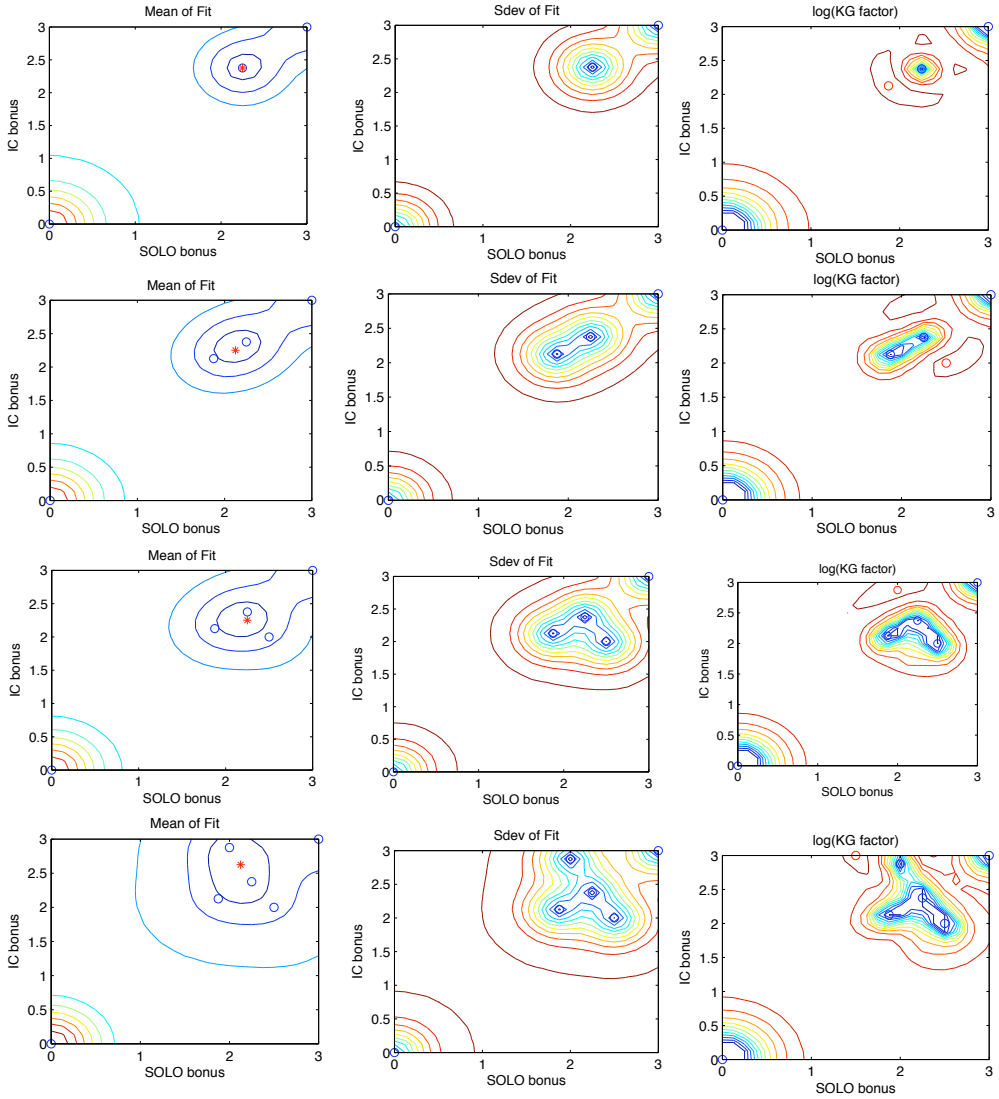


Figure 3: From top to bottom, the rows show times $m = 3,4,5,6$. The left column shows the mean $\mu^m_{i(\rho)}$ of our belief on $F(\rho)$, the central column shows the standard deviation $\sqrt{\Sigma^m_{i(\rho),i(\rho)}}$, and the right column shows the KG factor, $\nu^{KG}_\rho(S^m)$.

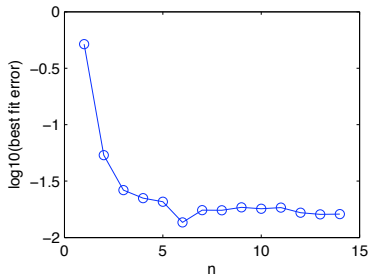


Figure 4: Logarithm of estimated fit deviation at the best-fitting ρ discovered so far, $\min_i \mu^m_i$, as a function of m.

REFERENCES

Bechhofer, R., T. Santner, and D. Goldsman. 1995. *Design and analysis of experiments for statistical selection, screening and multiple comparisons*. New York: J.Wiley & Sons.

Cao, X. R. 2007. *Stochastic learning and optimization*. Springer.

Chen, C., L. Dai, and H. Chen. 1996. A gradient approach for smartly allocating computing budget for discrete event simulation. *In Proceedings of the 1996 Winter Simulation Conference*, eds. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain, 398–405. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Chen, C., J. Lin, E. Yücesan, and S. Chick. 2000. Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization. *Discrete Event Dynamic Systems* 10 (3): 251–270.

Chick, S., J. Branke, and C. Schmidt. 2009. New myopic sequential sampling procedures. Submitted to INFORMS J. on Computing.

Chick, S., and K. Inoue. 2001. New two-stage and sequential procedures for selecting the best simulated system. *Operations Research* 49 (5): 732–743.

Cressie, N.A.C. 1993. *Statistics for Spatial Data, revised edition*, vol. 605. Wiley Interscience.

Frazier, P., and W. B. Powell. 2008. Asymptotic optimality of sequential sampling policies for Bayesian information collection. submitted.

Frazier, P., W. B. Powell, and S. Dayanik. The knowledge-gradient policy for correlated normal rewards. *Informs J. on Computing*.

Frazier, P., W. B. Powell, and S. Dayanik. 2008. A knowledge-gradient policy for sequential information collection. *SIAM J. on Control and Optimization* 47 (5): 2410–2439.

Gelman, A., J. Carlin, H. Stern, and D. Rubin. 2004. *Bayesian data analysis*. second ed. CRC Press.

Golub, G. H., and C. F. V. Loan. 1996. *Matrix computations*. Baltimore, MD: John Hopkins University Press.

Gupta, S., and K. Miescke. 1996. Bayesian look ahead one-stage sampling allocations for selection of the best population. *Journal of statistical planning and inference* 54 (2): 229–244.

He, D., S. Chick, and C. Chen. 2007, SEP. Opportunity cost and OCBA selection procedures in ordinal optimization for a fixed number of alternative systems. *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews* 37 (5): 951–961.

Jones, D., M. Schonlau, and W. Welch. 1998. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization* 13 (4): 455–492.

Kiefer, J., and J. Wolfowitz. 1952. Stochastic estimation of the maximum of a regression function. *Annals Mathematical Statistics* 23:462–466.

Kushner, H., and G. Yin. 2003. *Stochastic approximation and recursive algorithms and applications*. Springer.

Mockus, J. 1989. *Bayesian approach to global optimization: theory and applications*. Kluwer Academic, Dordrecht.

Montgomery, D. 1991. Design and analysis of experiments.

Myers, R., and D. Montgomery. 2002. *Response surface methodology: Process and product optimization using designed experiments*. New York: John Wiley & Sons.

Puterman, M. L. 1994. *Markov decision processes*. New York: John Wiley & Sons.

Robbins, H., and S. Monro. 1951. A stochastic approximation method. *Annals of Math. Stat.* 22:400–407.

Simão, H. P., J. Day, A. P. George, T. Gifford, J. Nienow, and W. B. Powell. 2009. An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science* 43 (1): 1–20.

Spall, J. C. 2003. *Introduction to stochastic search and optimization: Estimation, simulation and control*. Hoboken, NJ: John Wiley & Sons.

Swisher, J., S. Jacobson, and E. Yücesan. 2003. Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: A survey. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 13 (2): 134–154.

AUTHOR BIOGRAPHIES

PETER FRAZIER will receive a Ph.D. in Operations Research & Financial Engineering from Princeton University in 2009. He then begins an appointment as Assistant Professor in the department of Operations Research and Information Engineering at Cornell University. His research interest is in the optimal acquisition of information, with applications in simulation, medicine and operations management. His web address is <http://people.orie.cornell.edu/pfrazier> and his email address is <pfrazier@princeton.edu>.

WARREN B. POWELL is a professor in the department of Operations Research and Financial Engineering at Princeton University, and director of CASTLE Laboratory (<www.castlelab.princeton.edu>). An Informs Fellow, he has coauthored over 100 refereed publications in stochastic optimization, stochastic resource allocation and related applications. He is the author of *Approximate Dynamic Programming*, and is currently involved in applications in energy, transportation, finance and homeland security. His email address is <powell@princeton.edu>.

HUGO P. SIMÃO is a senior staff member at CASTLE Laboratory, in Princeton University. He holds a Ph.D. in Civil Engineering and Operations Research from Princeton University. He has worked on planning systems in the areas of transportation, distribution, logistics, and supply-chain. He has also taught courses on computer programming, basic probability and statistics, and resource and information management at the School of Engineering in Princeton. His e-mail address is <hpsimao@princeton.edu>.