No. 2008–95

CONSTRAINED OPTIMIZATION IN SIMULATION: A NOVEL APPROACH

By Jack P.C. Kleijnen, Wim van Beers, Inneke Van Nieuwenhuyse

November 2008

ISSN 0924-7815

Constrained Optimization in Simulation: a Novel Approach

November 6, 2008

Jack P.C. Kleijnen, Wim van Beers

Department of Information Management
Tilburg University
Postbox 90153, 5000 LE Tilburg, Netherlands

Inneke Van Nieuwenhuyse

Department of Decision Sciences and Information Management
K.U.Leuven
Leuven, Belgium

Abstract This paper presents a novel heuristic for constrained optimization of random computer simulation models, in which one of the simulation outputs is selected as the objective to be minimized while the other outputs need to satisfy prespecified target values. Besides the simulation outputs, the simulation inputs must meet prespecified constraints including the constraint that the inputs be integer. The proposed heuristic combines (i) experimental design to specify the simulation input combinations, (ii) Kriging (also called spatial correlation modeling) to analyze the global simulation input/output data that result from this experimental design, and (iii) integer nonlinear programming to estimate the optimal solution from the Kriging metamodels. The heuristic is applied to an (s, S) inventory system and a realistic call-center simulation model, and compared with the popular commercial heuristic OptQuest embedded in the ARENA versions 11 and 12. These two applications show that the novel heuristic outperforms OptQuest in terms of search speed (it moves faster towards high-quality solutions) and consistency of the solution quality.

Keywords Simulation; Design of experiments; Statistical analysis

JEL C0,C1,C9

1 Introduction

In this paper, we present a novel heuristic for constrained optimization of random computer simulation models. The importance of optimization, and the crucial role of computer simulation are widely recognized—especially in engineering (see Oden 2006). Though many other fields (e.g., logistics, supply chain management) have since long embraced the possibilities of simulation, its use in these areas has focused on system evaluation and what-if analysis. The interest in simulation for optimization purposes is still quite new, and has undoubtedly been stimulated by the availability of optimization tools in commercial software packages, such as OptQuest in the Arena simulation package; see Kelton et al. (2007).

In general, simulation optimization is recognized as a hard problem. Simulation outputs are implicit functions of the inputs, specified by the simulation code. Moreover, in random simulation, these outputs exhibit noise. The number of inputs may be very large, and running the simulation model may be computationally expensive. Simulation models for logistics systems present some additional problems. Typically, these are discrete-event models, where the input factors of interest are discrete (often integer) variables. Historically, research efforts in simulation optimization have focused on the continuous variable case (Fu 2002). Moreover, models of logistics systems produce multiple outputs. The optimization problem needs to reflect the trade-offs between these outputs. This problem can be approached in various ways (see the surveys by Beyer and Sendhoff 2007, Greenberg and Morisson 2008, and Rosen et al. 2007). A classic approach selects one output as 'the' objective, while requiring the other outputs to satisfy given threshold values; this is referred to as the *mathematical programming approach.* However, good algorithms for handling random constraints are scarce (Fu 2002).

In this paper, we present a novel heuristic for constrained nonlinear optimization of random simulation models that allows for integer inputs. It combines methodologies from three research fields: Design of Experiments (DOE), Kriging metamodeling, and Mathematical Programming (MP). We apply our heuristic to a realistic call-center simulation model, and compare its performance with the most popular commercial heuristic in simulation optimization, namely OptQuest. The empirical results show that our heuristic outperforms OptQuest in terms of search speed (it moves faster towards high-quality solutions) and consistency of the solution quality. This is undoubtedly desirable in case of computationally expensive experiments, but also in general because obtaining a high quality solution in the fewest number of evaluations is a core problem (Kelly 2002).

The remainder of this paper is organized as follows. Section 2 presents an Integer Non-

Linear Programming (INLP) formalization of the optimization problem. Next, Section 3 briefly presents the basics of Kriging metamodeling. Section 4 details our heuristic. Section 5 compares its computational results with results obtained by OptQuest for two test problems, namely an (s, S) inventory optimization model (Section 5.1), and a call-center staffing problem (Section 5.2). Section 6 presents our conclusions and topics for future research.

2 Mathematical Programming formulation

Formally, we try to solve the following *constrained nonlinear stochastic optimization problem*:

$$\min_{\mathbf{d}} E(w_0(\mathbf{d})) \tag{1}$$

where w_0 denotes the output that is selected as goal or objective variable, and $\mathbf{d}$ denotes the k-dimensional input (or decision) vector. The other (say) $(r-1)$ outputs $w_{h'}$ must satisfy the prespecified constraints

$$E(w_{h'}(\mathbf{d})) \leq a_{h'} (h' = 1, \ldots, r-1) \tag{2}$$

where the right-hand coefficients $a_{h'}$ are given. The objective function and the constraint functions are estimated through simulation; the simulation estimates are denoted by $\overline{w_h(\mathbf{d})}$ $(h = 0, \ldots, r-1)$. The k (deterministic) inputs d_j $(j = 1, \ldots, k)$ are required to be non-negative integers:

$$d_j \in \mathbb{N} \tag{3}$$

where $\mathbb{N}$ denotes the set of natural numbers including zero. Moreover, the inputs must satisfy given constraints:

$$f_{h''}(\mathbf{d}) \leq c_{h''} \text{ with } h'' = 1, \ldots, v. \tag{4}$$

These constraints may include box constraints of the form $l_j \leq d_j \leq h_j$ where l_j and h_j refer to the lower and upper bounds of input factor j $(j = 1, \ldots, k)$. Other linear or nonlinear constraints are also possible, as we shall show in our two applications.

A well-known academic example of this type of problems is the (s, S) inventory control system. In this system, the re-order level s and the order-up-to level S need to be determined in order to minimize expected total cost, which typically consists of the sum of expected inventory carrying costs and expected ordering costs. In practice, disservice costs are hard to quantify, so (dis)service is commonly treated as a constraint; e.g., the expected disservice percentage (complement of the fill rate) should not exceed 5% so $a_1 = 0.05$ in (2). The search area for s and S is typically limited by predetermined upper and lower bounds, and $s < S$.

The research literature on stochastic simulation optimization is dominated by continuous-variable problems. An example is the Sequential Kriging Optimization method (Huang et al. 2006), which is based on the Efficient Global Optimization (EGO) method (Jones et al. 1998), and which focuses on continuous and unconstrained optimization. Bashyam and Fu (1998) propose a feasible directions procedure that is generally applicable for continuous optimization problems with a single noisy constraint. For unconstrained discrete-valued problems, random search methods (e.g., simulated annealing) are mostly used (Fu 2002, Fu et al. 2005, Andradóttir 1999, 2002). Research on integer constrained optimization of stochastic simulations seems to be scarce. The procedure by Abspoel et al. (2001) uses local linear approximations, and hence may encounter problems when either the objective or the constraints are highly nonlinear. Moreover, as demonstrated by Driessen et al. (2006a), the use of local approximations for solving integer optimization problems can easily result in inferior solutions. Recently, Atlason et al. (2008) published an algorithm for constrained integer simulation optimization based on the cutting-plane method; this method, however, requires that the constraint functions are pseudoconcave. The algorithm developed by Cezik and L' Ecuyer (2008) runs into similar problems when exploring regions of the search area where the constraint functions are nonconcave. Recently, Davis and Ierapetritou (2007) combined the use of Kriging metamodels (for global exploration) and Response Surface Modelling (for local exploration) for optimizing process synthesis problems. In their setting, however, the uncertainty stems solely from the noise on the input variables; both objective and constraints are explicit functions of the input variables. As described above, the issues in discrete-event simulation tend to be different.

3 Ordinary Kriging

Kriging is a type of metamodel that enables adequate approximations of the input/output (I/O) function implied by the underlying simulation model when the simulation experiment covers a "big" area; i.e., the experiment is global, not local (see Kleijnen 2008a). Kriging models are commonly used for prediction. A general review of Kriging is given in Kleijnen (2009). Ordinary Kriging—from now on, briefly called Kriging—is univariate in nature, and makes the following two assumptions:

(i) the *model assumption* is that the simulation output $\overline{w(\mathbf{d})}$ at input combination $\mathbf{d}$ consists of a constant μ and an error term $\delta(\mathbf{d})$:

$$\overline{w(\mathbf{d})} = \mu + \delta(\mathbf{d}); \tag{5}$$

(ii) the *predictor assumption* is that the predictor $y(\mathbf{d}_c)$ at an arbitrary input combination $\mathbf{d}_c$ is a weighted linear combination of all the observed output data $\overline{w(\mathbf{d}_i)}$ at n already simulated input combinations $\mathbf{d}_i$ $(i = 1, ..., n)$:

$$y(\mathbf{d}_c) = \sum_{i=1}^{n} \lambda_i \overline{w(\mathbf{d}_i)}. \tag{6}$$

To select the *optimal* weights in (6), Kriging uses the *Best Linear Unbiased Predictor (BLUP)* criterion, which minimizes the Mean Squared Error (MSE) of the predictor y. This "Unbiased" condition means

$$E[y(\mathbf{d}_c)] = E[w(\mathbf{d}_c)]. \tag{7}$$

It can be proven that the solution of the constrained minimization problem implied by the BLUP criterion implies

$$\sum_{i=1}^{n} \lambda_i = 1. \tag{8}$$

Furthermore, it can be proven that the *optimal* weights are given by the vector

$$\lambda = \mathbf{\Gamma}^{-1}(\gamma + \mathbf{1}\frac{1 - \mathbf{1}'\mathbf{\Gamma}^{-1}\gamma}{\mathbf{1}'\mathbf{\Gamma}^{-1}\mathbf{1}}) \tag{9}$$

where

$\mathbf{\Gamma}$ $=cov(\overline{w(\mathbf{d}_i)}, \overline{w(\mathbf{d}_{i'})})$ with $i, i' = 1, \ldots, n$ is the $n \times n$ (symmetric and positive semi-definite) matrix with the covariances between the n available outputs;

γ $=cov(\overline{w(\mathbf{d}_i)}, \overline{w(\mathbf{d}_c)})$ is the n-dimensional vector with the covariances between the n available outputs and the output to be predicted.

Consequently, the optimal weights (9) depend on the covariances in $\mathbf{\Gamma}$ and γ. Kriging assumes that these covariances are determined by the distances between the input combinations. Moreover, Kriging usually assumes that the covariances in the k-dimensional space are the product of the k individual covariances. A popular correlation function is

$$\rho = \exp(-\sum_{j=1}^{k} \theta_j |d_{j;i} - d_{j;c}|^{p_j}) = \prod\nolimits_{j=1}^{k} \exp(-\theta_j |d_{j;i} - d_{j;c}|^{p_j}) \tag{10}$$

where

$|d_{j;i} - d_{j;c}|$ denotes the distance between the j-th input factor of combination i and combination c;

θ_j denotes the importance of input factor j; i.e., the higher θ_j is, the faster the correlation decreases with the distance;

p_j denotes the smoothness of the correlation function; so-called "exponential" and "Gaussian" correlation functions have $p = 1$ and $p = 2$ respectively.

Other types of correlation functions (e.g., spherical, cubic, linear) can be found in Lophaven et al. (2002). All these types imply correlation functions that are decreasing in the distances between the inputs. This implies that the optimal weight λ_i associated with output $\overline{w(\mathbf{d}_i)}$ in (6) will be relatively high for $\mathbf{d}_i$ close to $\mathbf{d}_c$.

The true correlation function is unknown, so both the type and the parameter values θ_j must be estimated—for each simulation output separately—from the available data (i.e., the n available input combinations $\mathbf{d}_i$ with the associated simulation output $\overline{w_h(\mathbf{d}_i)}$, $h = 0, .., r-1$). To estimate these parameters, Kriging usually applies Maximum Likelihood Estimators (MLEs). The MLEs of the correlation parameters θ_j require constrained maximization. This optimization is a hard problem, because matrix inversion is necessary, multiple local maxima may exist, etc.; see Martin and Simpson (2004, 2005). To estimate the correlation functions and the corresponding optimal weights, we use the Matlab Kriging toolbox called *DACE*, which is well documented and free of charge; see Lophaven et al. (2002).

We point out that some of the weights may be negative. Furthermore, for an "old" input combination $\mathbf{d}_i$ the predictor is an *exact interpolator*; i.e., it equals the observed average simulation output:

$$y(\mathbf{d}_i) = \overline{w(\mathbf{d}_i)}, \tag{11}$$

which implies that all weights are zero except the weight of the observed output, which has the value one. Recently, Ankenman et al. (2008) investigated Kriging (for random simulation) that is not an exact interpolator; however, no software is available yet. As Kriging (and, hence, DACE) treats the simulation outputs as univariate, it does not incorporate cross-correlations between the r random simulation outputs. This could be accomplished through so-called multivariate Kriging (discussed in Chapter 5 of Chilès and Delfiner 1999). Our heuristic, however, uses univariate Kriging per output type.

4 Novel simulation-optimization heuristic

Our approach is iterative; for ease of reference, the different steps are summarized in Figure 1. Each step will be discussed in detail in the following subsections; here, we present an overview.

We start by selecting an initial space-filling design (step 1), and simulate all input combinations of this design (step 2). This yields the multiple simulation outputs $\overline{w_h(\mathbf{d}_i)}$ ($h = 0, .., r-1$) for each input combination $\mathbf{d}_i$. Next, we fit r univariate Kriging metamodels to these simulation I/O data (step 3) and validate these models (step 4). As long as one or more metamodels is judged to be invalid, the design is updated by simulating an additional input combination (step

5) from the global search area, in order to further fine-tune the metamodels. The heuristic then loops back to step 3, to refit the Kriging models using the newly available I/O information. When all r Kriging metamodels are judged to be valid, an INLP solver is used to estimate the location of the optimum based on the Kriging models (step 6). We check whether the indicated optimum has already been simulated (step 7). If this is the case indeed, then this implies that the I/O data of this point are already available in the design, and we rerun the INLP code to determine the "next best" point (step 8), i.e., that input combination that is optimal when all already simulated points are excluded from the optimization. The newly found point (from step 6 or step 8) is then simulated, and its I/O data are added to the design (step 9). In step 10, we compare the simulation output data of the newly simulated point with the output data of the best point found so far. If the last a INLP runs were unable to point out an input combination yielding a statistically significant improvement in the objective, we stop the procedure and accept the point that currently has the best value for the objective as the optimum. The parameter a can be chosen by the user; we set it equal to 30 in our experiments. As long as the stopping condition is not met, the Kriging metamodels are updated with the newly available I/O information, and the heuristic continues its search.

Note that the initial design is sequentially updated: additional points are added in steps 5 and 9 respectively (a similar sequential approach is proposed in Bates et al. 2006 and Kenett and Steinberg 2006). Points in step 5 are selected to improve the metamodel, which is called "geometry improving" in Driessen et al. (2006a); points in step 9 are selected to find the optimum. These two reasons are also related to "exploration" and "exploitation" respectively, in classic simulation optimization; see e.g. Fu (2007).

4.1 Step 1: Select initial space-filling design

We start with a small space-filling initial design (or pilot sample), which is determined by an $n_0 \times k$ design matrix $\mathbf{D}$ with rows $\mathbf{d'}= (d_1, \ldots, d_k)$. Space-fillingness is a desirable property for the initial design, in order to capture as much information as possible on the behavior of objective and constraints throughout the search space (we assume that we have no advance information on this behavior). We opt for a maximin LHS design (as suggested in Driessen et al. 2006a, but other spacefilling designs are possible, see Cioppa and Lucas 2007 and Kleijnen 2008a). The rule of thumb given in the literature calls for a design size n_0 equal to $10k$; see Jones et al. (1998). However, as our design is sequentially updated, we propose to limit the desired initial design size to $n_0 = 5 + 2k$. Intuitively, it seems reasonable to estimate a smooth curve for the correlation functions (10) from at least four distances between points, so a basic

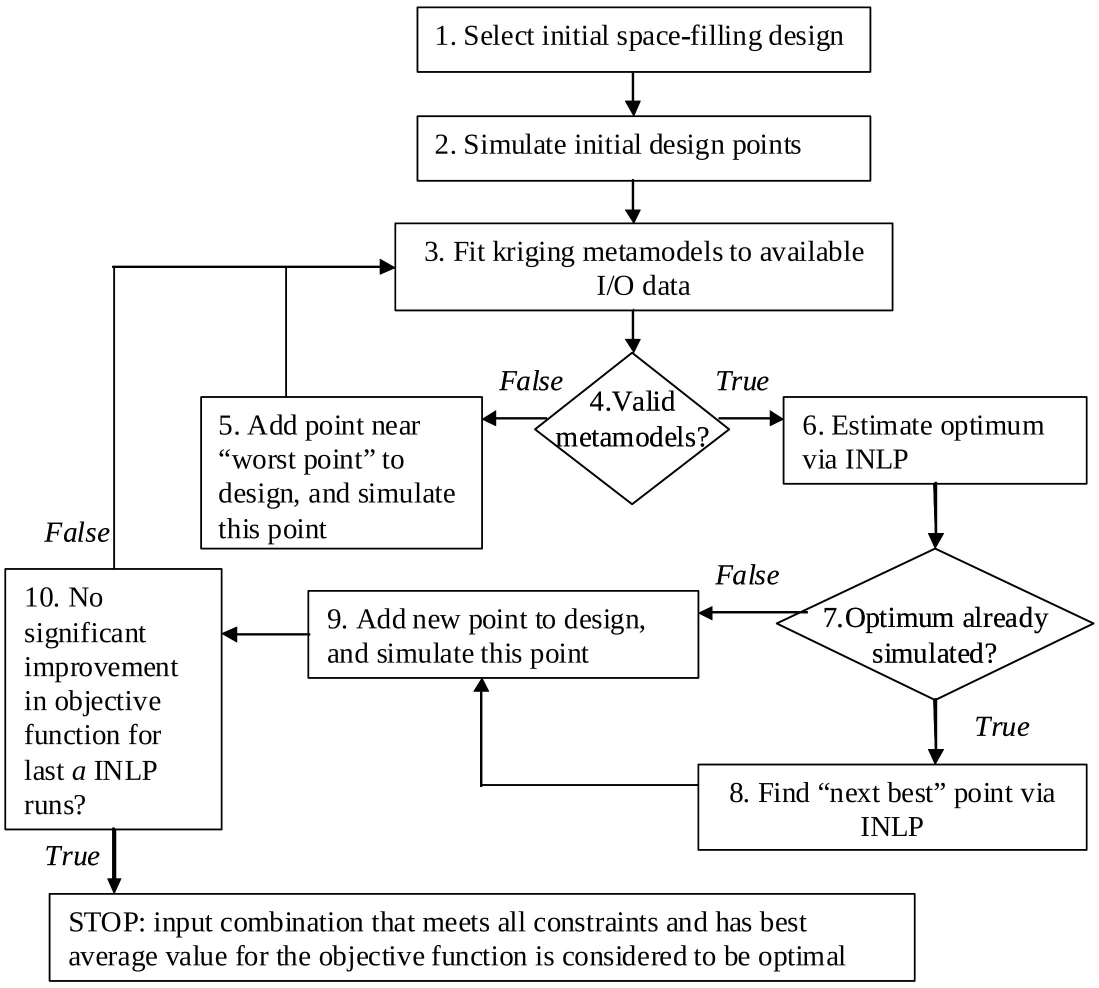
1. Select initial space-filling design
2. Simulate initial design points
3. Fit kriging metamodels to available I/O data
4.Valid metamodels?
False
True
5. Add point near "worst point" to design, and simulate this point
6. Estimate optimum via INLP
7.Optimum already simulated?
False
True
8. Find "next best" point via INLP
9. Add new point to design, and simulate this point
10. No significant improvement in objective function for last a INLP runs?
False
True
STOP: input combination that meets all constraints and has best average value for the objective function is considered to be optimal

Figure 1: Heuristic: stepwise approach.

rule-of-thumb could be $n_0 = 5$. The remaining $2k$ combinations are added to ensure sufficient observations for cross-validation (see Section 4.4 below).

The pilot sample includes only input combinations that satisfy the input constraints in (4). The standard maximin LHS design can take into account the box constraints in (4), by coding the inputs such that they are between 0 and 1, yielding real values $u_{i;j}$ (where $u_{i;j}$ denotes the value for input j in combination i, $0 \leq u_{i,j} \leq 1$). This standard design, however, needs adaptation for the integer constraints (3) and for any non-box input constraints (4). To account for the integer constraints (3), the $u_{i;j}$ are transformed into the integer input values

$$d_{i;j} = l_j + \lfloor (h_j - l_j + 1) u_{i;j} \rfloor \tag{12}$$

where $\lfloor x \rfloor$denotes the floor function of x so the real x is rounded downwards to the closest integer. To account for any non-box input constraints (4), we estimate the *feasible fraction* f, which is the fraction of the k-dimensional search space that satisfies these constraints. Because these constraints may be rather complicated, we use a simple Monte Carlo estimate $\widehat{f}$; i.e., we take a design of size n (with $n >> n_0$), apply the transformation in (12), and compute

$$\widehat{f} = \frac{\sum_{i=1}^{n} I[\forall h'' : f_{h''}(\mathbf{d}) \leq c_{h''}]}{n} \tag{13}$$

where the indicator function $I[.] = 1$ if the input combination $\mathbf{d}$ satisfies all non-box input constraints in (4); else $I[.] = 0$.

We then select a maximin LHS design with sample size $\widehat{n_0} = \lceil n_0/\widehat{f} \rceil$ where $\lceil . \rceil$ means rounding upwards to the next integer. If the realized acceptable number of combinations from the design is different from the targeted n_0, we repeat the procedure (with or without a change in $\widehat{n_0}$)—until an acceptable initial number of feasible input combinations is obtained.

Note that n_0 increases linearly in k; this implies that—for high-dimensional search spaces (large k)—a large set of initial simulations will be required. This is, in general, a drawback of using global metamodels (Driessen et al. 2006a). We therefore recommend to first reduce the number of input variables by adding a preceding screening step (Kleijnen 2008b). Moreover, a small value of k is desirable to make the resulting integer design *noncollapsing*; i.e., the projection of the design points in the k-dimensional space onto any of the k axes gives a uniform spacing of that axis without any points becoming identical (Driessen et al. 2006a). Collapsing occurs in integer designs whenever n_0 exceeds the number of integer values allowed for any input variable.

4.2 Step 2: Simulate initial design points

Once we have obtained an experimental design with size n_0, we simulate all combinations in this design. At this point, we need to decide on the desired number of replicates m_i for input combination i $(i = 1, \ldots, n)$. The choice of m_i reflects the magnitude of the noise we wish to accept. In our experiments, we use two options:

(i) simulate a fixed number of replications per input combination, so $m_i = m$;

(ii) select the number of replicates m_i based on a relative precision requirement; i.e., select m_i such that the halfwidth $l_h(m_i, \alpha)$ of the $(1-\alpha)$ confidence interval for the average simulation output $\overline{w_h(\mathbf{d}_i)}$ is within $\gamma\%$ of the true mean for all outputs h $(h = 0, .., r-1)$. Law (2007), pp. 500-503 implies that if

$$\forall h : \frac{l_h(m_i, \alpha)}{\left|\overline{w_h(\mathbf{d}_i)}\right|} \leq \frac{\gamma}{1+\gamma}, \tag{14}$$

then m_i realizes the relative precision γ with $0 < \gamma < 1$. The halfwidth of the $(1-\alpha)$ confidence interval is given by

$$l_h(m_i, \alpha) = t_{m_i-1;1-\alpha/2}\sqrt{\frac{\widehat{var(w_h(\mathbf{d_i}))}}{m_i}}$$

where $t_{m_i-1;1-\alpha/2}$ denotes the $(1-\alpha/2)$ quantile of the t_{m_i-1} distribution. The average simulation output is

$$\overline{w_h(\mathbf{d}_i)} = \frac{\sum_{l=1}^{m_i} w_{h;l}(\mathbf{d}_i)}{m_i} \tag{15}$$

with $w_{h;l}(\mathbf{d}_i)$ denoting simulation output h of replication l of input combination $\mathbf{d}_i$. The estimated variance is

$$\widehat{var(w_h(\mathbf{d}_i))} = \frac{\sum_{l=1}^{m_i}(w_{h;l}(\mathbf{d}_i) - \overline{w_h(\mathbf{d}_i)})^2}{m_i - 1} \text{ with } m_i > 1. \tag{16}$$

Because we use the I/O data to fit metamodels, we prefer a high signal/noise ratio. If the m_i are too small, then the signal/noise is too small and we fit Kriging models largely to noise. We therefore expect our heuristic to perform better with option (ii).

Another tactical issue is the possible use of a Variance Reduction Technique (VRT). We use Common Random Numbers (CRN) (see Law 2007) because we expect CRN to create positive correlation between outputs—so even outputs at input combinations that are relatively far away from the input combination to be predicted, still contain information (the parameters θ_j in (10) become smaller).

4.3 Step 3: Fit Kriging metamodels to available I/O data

In this step, we use the DACE toolbox to fit r univariate Kriging models to the available simulation I/O data. We fit the metamodels in the total experimental area; i.e., we also include those subareas that are either *clearly infeasible* (i.e., the input combinations located in that area violate the output constraints in (2)), or *clearly suboptimal* (feasible, but with a significantly higher goal output). The reason is that information regarding the location of infeasible or suboptimal regions is important for steering the INLP search in steps 6 and 8.

4.4 Step 4: Validate the Kriging metamodels

We apply *cross-validation*, similar to the procedure in Kleijnen (2008a). The procedure can be outlined roughly in the following six stages (we shall discuss each of these stages in further detail):

1. From the set of simulation I/O data, delete one input combination at a time—but avoid extrapolation.

2. Based on the remaining I/O data, compute the Kriging predictors for every output h ($h = 0, ..., r-1$) at the left-out combination.

3. Estimate the predictor variance for every output h ($h = 0, ..., r-1$) at the left-out combination $\mathbf{d}_i$.

4. Compute the Studentized prediction error $t^{h,i}_{m_i-1}$ for every output h ($h = 0, ..., r-1$) of the left-out combination i, which is replicated m_i times.

5. Repeat the preceding four steps, until all combinations available for cross-validation have been left out one-at-a-time.

6. Determine the highest absolute value of the observed $t^{h,i}_{m_i-1}$ over all h and i, and determine if this value is statistically significant. If this is the case, all r Kriging models are rejected (simulation gives estimates of all r outputs; the heuristic then moves to step 5). Else, the metamodels are accepted as being valid (in which case they are ready for use in the INLP procedure, step 6).

Now we discuss each of these stages in further detail.

Sub (1): To avoid extrapolation, the set of input combinations that can be left out in the cross-validation is restricted to those combinations $\mathbf{d}_i$ for which none of the input factors

(d_1,...,d_k) contains an extreme value (i.e., the highest or lowest value for that input factor, across the n points already simulated). The reason is that the literature suggests that Kriging models should not be used for extrapolation: for input combinations that fall in one of the extreme bins for a given input factor, Kriging will give bad predictions—even if the Kriging model is valid (see Kleijnen and Van Beers 2004). This implies that we cross-validate only $n_{cv} = n - 2k$ I/O combinations.

Sub (2): Deleting I/O combination i ($i = 1, \ldots, n_{cv}$) from the set of simulation I/O data yields a corresponding remaining I/O data set $(\mathbf{d}_{-i}, \overline{\mathbf{w}_{-i}})$. We call this data set the *cross-validation set*. Note that for any I/O combination i, the cross-validation set consists of $n - 1$ combinations. To predict output h at the left-out combination i from this cross-validation set, we re-estimate only the optimal Kriging weights (we may refer to these as λ_{-i}) through (9), using the current MLE estimates for the correlation function in (10). This yields a predictor $y_h^-(\mathbf{d}_i)$ for every output h. We do not re-estimate the correlation functions in (10), as the current estimates (based on all n observations) are more reliable. This approach is also followed by Jones et al. (1998) and Joseph et al. (2008).

Sub (3): To estimate the variance of the Kriging predictor, the literature often uses a biased estimator (see e.g. Lophaven et al. 2002, Jones et al. 1998). This literature does not account for the estimation of the parameters θ_j in (10), which makes the Kriging predictor a nonlinear estimator (Den Hertog et al. 2006). We use an unbiased estimator based on *distribution-free bootstrapping*, analogously to the procedure in Van Beers and Kleijnen (2008). In our procedure, however, we face three complications: firstly, every replicate l of a given input combination $\mathbf{d}_i$ gives a multivariate output vector $(w_{0,l}(\mathbf{d}_i), w_{1,l}(\mathbf{d_i}), \ldots, w_{r-1,l}(\mathbf{d}_i))^T$. For example, a relatively high realization of w_0 (namely, $w_{0,l}(\mathbf{d}_i) > \overline{w_0(\mathbf{d}_i)}$) tends to occur together with a relatively high realization of w_1 (namely, $w_{1,l}(\mathbf{d}_i) > \overline{w_1(\mathbf{d_i})}$) if these two outputs are positively correlated. Secondly, the use of *CRN* makes the output vectors of the same replicate l of the n simulated input combinations positively correlated (Kleijnen and Deflandre 2006). Thirdly, the number of replicates m_i may differ depending on the input combination $\mathbf{d}_i$, when the relative precision requirement is used to determine m_i (see Section 4.2). We account for these three complications in the following distribution-free bootstrapping procedure:

1. Define the maximum number of replicates across the different input combinations of the cross-validation set: $m = \max\{m_z | \, z \neq i\}$ $(i = 1, ..., n_{cv})$.

2. Bootstrap the replicate numbers from 1 to m (i.e., sample m replicate numbers l from the uniform distribution with constant probability $1/m$ for the m integers $[1, 2, \ldots, m]$). This

yields a set of m replicate numbers $\{l_1, l_2, ..., l_m\}$.

3. For each input combination $\mathbf{d}_z$ in the cross-validation set, include the output vectors $(w_{0,l_1}(\mathbf{d_z}),\ w_{1,l_1}(\mathbf{d}_z),\ \ldots,\ w_{r-1,l_1}(\mathbf{d}_z))$ to $(w_{0,l_m}(\mathbf{d_z}),\ w_{1,l_m}(\mathbf{d}_z),\ \ldots,\ w_{r-1,l_m}(\mathbf{d}_z))$ in the bootstrapped data set. This yields a bootstrapped data set containing m_z^* output vectors per combination z. Note that if $m_z < m$ for an input combination $\mathbf{d}_z$ (which is likely to be the case when the simulations were run according to the relative precision requirement), it may happen that some of the output vectors do not exist (so $m_z^* < m$). If $m_z^* = 0$ for any input combination $\mathbf{d}_z$, the bootstrap sample is rejected.

4. Compute the bootstrap outputs $\overline{w_h^*(\mathbf{d_z})}$ $(h = 0, ..., r-1)$ averaged over replications, for all combinations $\mathbf{d}_z$ in the cross-validation set:

$$\overline{w_h^*(\mathbf{d}_z)} = \frac{\sum_{j=1}^{m_z^*} w_{h,l_j}(\mathbf{d}_z)}{m_z^*}(h = 0, \ldots, r-1), \tag{17}$$

5. Calculate the bootstrapped estimated optimal Kriging weights λ_h^* and the corresponding bootstrapped Kriging predictor $y_h^*(\mathbf{d}_i)$ using the $n-1$ bootstrapped averages $\overline{w_h^*(\mathbf{d}_z)}$ for simulation output h in (17).

To decrease the sampling effects of bootstrapping, we repeat this whole procedure until we have B successful bootstrap samples (in our experiments, we select $B = 200$), yielding estimated optimal Kriging weights $\lambda_{h,b}^*$ and bootstrapped Kriging predictors $y_{h,b}^*(\mathbf{d}_i)$ with $b = 1, \ldots, B$. These B values enable estimation of the variance of the Kriging predictor, analogously to (16):

$$\widehat{var(y_h^*(\mathbf{d}_i))} = \frac{\Sigma_{b=1}^{B}(y_{h;b}^*(\mathbf{d}_i) - \overline{y_h^*(\mathbf{d}_i)})^2}{B-1} \tag{18}$$

where

$$\overline{y_h^*(\mathbf{d}_i)} = \frac{\Sigma_{b=1}^{B} y_{h;b}^*(\mathbf{d}_i)}{B}. \tag{19}$$

Sub (4): Compute the *Studentized prediction error* for every output h $(h = 0, ..., r-1)$ of the left-out combination $\mathbf{d_i}$:

$$t_{m_i-1}^{h,i} = \frac{\overline{w_h(\mathbf{d}_i)} - y_h^-(\mathbf{d}_i)}{\sqrt{\widehat{var(\overline{w_h(\mathbf{d}_i)})} + \widehat{var(y_h^*(\mathbf{d}_i))}}}\ (i = 1, \ldots, n_{cv}) \tag{20}$$

where $\widehat{var(\overline{w_h(\mathbf{d}_i)})} = \widehat{var(w_h(\mathbf{d}_i))}/m_i$ and $\widehat{var(w_h(\mathbf{d_i}))}$ follows from (16). Note that the numerator of (20) uses $y_h^-(\mathbf{d}_i)$, not $\overline{y_h^*(\mathbf{d}_i)}$; see (19). We conjecture that both $y_h^-(\mathbf{d}_i)$ and $\overline{y_h^*(\mathbf{d}_i)}$ have the same asymptotic mean as $B \uparrow \infty$ (where $B \uparrow \infty$ means B increases to infinity). However, had we used $\overline{y_h^*(\mathbf{d}_i)}$ in the numerator, then in the denominator of (20) we would use

$\widehat{var(\overline{y_h^*}(\mathbf{d}_i))} = \widehat{var(y_h^*(\mathbf{d}_i))}/B$. But the latter expression decreases to zero as $B \uparrow \infty$. Actually we use $\widehat{var(y_h^*(\mathbf{d}_i))}$ in the denominator, for which $\widehat{var(y_h^*(\mathbf{d}_i))} \longrightarrow \sigma^2 > 0$ as $B \uparrow \infty$, where σ^2 denotes the unknown predictor variance (see the numerator and denominator in (18)). Note that $\widehat{var(y_h^*(\mathbf{d}_i))}$ is an estimator of $\widehat{var(y_h^-(\mathbf{d}_i))}$, and this $\widehat{var(y_h^-(\mathbf{d}_i))} \downarrow 0$ as $n \uparrow \infty$ (not as $B \uparrow \infty$). Also see Efron and Tibshirani (1993), pp. 124-140.

Sub (5): Repeat the preceding four steps, until all n_{cv} combinations have been left out one-at-a-time. This results in n_{cv} Studentized prediction errors $t^{h,i}_{m_i-1}$ per output type h $(h = 0, ..., r-1)$.

Sub (6): Determine the highest absolute value of these Studentized prediction errors and determine if this value is statistically significant. The error is considered to be significant if

$$\max_h [\max_i \left| t^{h,i}_{m_i-1} \right|] > t_{m_{min}-1;1-[\alpha/(2n_{cv}r)]} \; (h = 0, \ldots, r-1; i = 1, \ldots, n_{cv}). \tag{21}$$

where the right-hand side uses degrees of freedom $m_{min} = \min\{m_i | \, i = 1, ..., n_{cv}\}$ and a significance level that follows from Bonferroni's inequality, which implies that the classic type-I error rate (in this case $\alpha/2$) is divided by the number of tests (in this case $n_{cv}r$)—resulting in the "experimentwise" or "familywise" type-I error rate. We choose $\alpha = 0.15$ in our experiments (higher values than the traditional 0.10 or 0.01 are acceptable in experimentwise testing; see Miller 1981). We call the input combination that causes the rejection of the metamodels the *worst point.*

4.5 Step 5: Add point near "worst point" to design

If the Kriging models are rejected, we augment the design with a new combination to improve the (global) Kriging models. Because the worst point gives the maximum Studentized prediction error (see (21)), we need extra information about the metamodels' behavior in the neighborhood of the worst point. Simulating a point too close to that point (or any other point in the current design), however, would give little new information, because Kriging assumes that input combinations near each other have outputs with high positive correlations. Therefore, we select the point halfway the worst point and its nearest neighbor in the current design, where distances between points are measured through Euclidean distances (as in the "distance list" of Cioppa and Lucas 2007). In case of ties (which certainly occur in the maximin pilot sample), we select the nearest neighbor with the minimum simulated value for the goal variable. To ensure that the new combination satisfies the integer constraints and any non-box constraints in (4), the coordinates of this point are rounded to the next lower integers. The simulation output data from the newly selected point are added to the I/O data, so all r Kriging models can be re-estimated in step 3. We keep adding new points in this way, until all r Kriging models are accepted as valid.

4.6 Steps 6-7: Estimate optimum via INLP, and check if it is new

Given the integer input constraints and the nonlinear behavior of both the objective and the constraints, we apply an INLP code to the Kriging metamodels, in order to estimate the optimum input combination. Many INLP codes are available (see e.g. Pintér 2007 and http://www.gamsworld.org/minlp/solvers.htm). In our experiments, we use the *bnb20.m* code, which is a branch-and-bound type of algorithm written for the Matlab environment and available for free download at http://www.mathworks.com/matlabcentral/fileexchange. A disadvantage of this solver is that it guarantees only local optimality. Hence, it needs to be used with multiple starting points; we chose to use three starting points in our experiments. In principle, the user may apply any INLP code in this step; the use of a more advanced solver is likely to further improve the effectiveness of the heuristic.

If the optimum input combination determined by the INLP code has already been simulated (and, hence, is part of the simulated design), the heuristic goes to step 8. Otherwise, it proceeds to step 9.

4.7 Step 8: Find "next best" point by INLP

In Step 6, INLP may give a previously simulated point as the optimum. In that case, we rerun the INLP code with the additional constraint that it should return a point that is not yet part of the design. This point is then referred to as the "next best point". Let $\mathbf{d}_t$ $(t = 1, \ldots, T)$ denote the (say) T points that have already been simulated during the previous iterations. To select a new point excluding these $\mathbf{d}_t$ requires the following T additional constraints

$$1 \leq \sum_{j=1}^{k} (d_{t;j} - d_j)^2 \ (t = 1, \ldots, T). \tag{22}$$

These constraints allow INLP to explore other promising points in a big neighborhood around the old optimum. The location of the next best point strongly depends on the behavior of both the objective function and the constraint functions; hence, it may be located far away from the old optimum.

4.8 Step 9: Add new point to design, and simulate

In this step we add the latest estimated optimal input combination —determined either in step 6 or in step 8—to the design, and simulate this point.

4.9 Step 10: No significant improvement for last a INLP runs?

This step embodies the stopping rule for our heuristic. After we have simulated the input combination determined by INLP, we check the following two conditions: (i) feasibility: the point needs to meet all output constraints, and (ii) optimality: if it is a feasible point, its corresponding objective value needs to be significantly better than the best point found so far. Only if both conditions are satisfied, the newly simulated point is considered to be the new best point.

Condition (i) is satisfied if the $(r-1)$ simulation estimates meet their corresponding thresholds a_h. For condition (ii), we compare the average objective value of the current combination (say) $\mathbf{d}_{curr}$ with that of the best point found so far (say) $\mathbf{d}_{best}$ using the following test statistic:

$$t_{\upsilon} = \frac{\overline{w_0(\mathbf{d}_{curr})} - \overline{w_0(\mathbf{d}_{best})}}{\sqrt{var(\widehat{\overline{w_0(\mathbf{d}_{curr})}}) + var(\widehat{\overline{w_0(\mathbf{d}_{best})}})}}$$

with $\nu = \min(m_{curr}, m_{best})$ where m still denotes the number of replicates. Only if $t_{\upsilon} < -t_{\upsilon;1-\alpha/2}$, the new input combination $\mathbf{d}_{curr}$ is significantly better than $\mathbf{d}_{best}$. We use $\alpha = 0.1$ in our experiments.

We use a counter to keep track of the number of consecutive times that at least one of the checks "fails"; i.e., INLP failed to come up with a statistically significantly better combination. We propose to stop the heuristic when this counter crosses a user-determined threshold a (we chose a=30 in our experiments). The simulated input combination that has the lowest average value for the objective, and simultaneously meets all constraints, is the final optimum.

Our heuristic ensures convergence to the true optimum as the expended computational effort grows (i.e., the number of replicates per input combination tends to infinity and the number of input combinations simulated approaches the—finite—size of the search space; see also Cezik and L' Ecuyer 2008). Increasing the number of replicates reduces the noise of the outputs, which ensures that the simulation estimates for the objective value and the constraints approach their true values. The number of possible input combinations is by definition finite in our setting because of the combination of integer and box constraints.

5 Computational results

Unfortunately, there are no standard testbeds for constrained random simulation optimization (in contrast to, e.g., Mathematical Programming). In the literature, it is common to test heuristics on known mathematical functions, augmented with additive white noise (e.g., Angün et al. 2002). The appendix to our article illustrates the application of our heuristic to such a "toy problem". In this section, we report the computational results of our heuristic for two test problems taken from the field of manufacturing and logistics.

Section 5.1 discusses the performance for a variant of the popular $(s,\ S)$ inventory problem by Bashyam and Fu (1998) (which was also used by Kleijnen and Wan 2007). Section 5.2 uses a call center staffing problem taken from Kelton et al. (2007). We compare the computational results of our heuristic with those of the popular simulation-optimization software OptQuest (provided by OptTek System Inc, and available as an add-on to simulation software such as Arena, CrystallBall, MicroSaint, ProModel, and Simul8; it also comes free of charge with the student version of the Arena software, which is provided by Rockwell Software). OptQuest uses a combination of tabu search, neural networks, and scatter search to estimate the optimum; for commercial reasons, however, the exact heuristic remains a black box.

Unfortunately, different versions of Arena with OptQuest give different results; we report results for two recent versions, namely the Arena versions 11 and 12. The heuristic is written in Matlab's version 7.4. For each test problem, we use ten macroreplicates for our heuristic and OptQuest. The number of replications per input combination was either fixed or determined by a relative precision requirement with $\gamma = 0.15$. For each macroreplicate, we present (i) the reported optimum, (ii) the total number of design points simulated by the heuristic (i_{max}), and (iii) the sequence number of the reported optimum in the list of points simulated (rank $\mathbf{d_{opt}}$). Though OptQuest allows the user to select a "relative precision requirement", we apply it only with a fixed number of replications per design point. The reason is that the interpretation of this "requirement" seems to differ; the User's Guide (version 11.00) indicates that it is used to determine whether a design point looks promising enough (in objective value) to continue additional replications. For each macroreplicate, we report the status of OptQuest's estimated optimum after having simulated the number of points that our heuristic required.

5.1 The $(s,\ S)$ inventory system

We consider an infinite-horizon periodic-review (s, S) inventory simulation model with full backlogging that was originally studied by Bashyam and Fu (1998). This model assumes that demand per time period is exponentially distributed with an average of 100 units. The inventory position (defined as stock on hand minus backorders plus any outstanding orders) is checked at the end of every time period; when the inventory position has dropped to a value smaller than or equal to the reorder level s, a replenishment order is placed to bring the inventory position back to S. Replenishment lead times are Poisson distributed with an average of 6 time periods, so replenishment orders may cross in time. Replenishment orders are received at the beginning of a period (Kleijnen and Wan 2007 study a variant of this model, where orders are received at the end of a previous period). The holding cost per period is 1, the fixed ordering cost is 36, and the variable ordering cost (per unit) is 2. The objective is to determine the values of s and $Q = S - s$ that minimize TC, which denotes the expected Total Costs (consisting of average ordering and holding cost) per period, subject to the constraint that the disservice rate δ (fraction of demand that is not met from stock on hand) is at most 10%. Because this problem is analytically intractable due to the order crossings, simulation is required. The simulation uses 30,000 periods per replication (as in Bashyam and Fu 1998). The search area is limited to $900 \leq s \leq 1250$ and $1 \leq Q \leq 500$.

The number of replications per input combination is either fixed at $m_i = m = 10$ (as in Bashyam and Fu 1998) or determined by a relative precision requirement with $\gamma = 0.15$; given the long replication length (30,000 periods), only 4 to 6 replications per design point suffice to reach this precision.

Table 1 shows the results of our heuristic (DOE-Kri-MP) for 10 macroreplicates. It also shows the OptQuest results (obtained with Arena versions 11 and 12) for the same 10 macroreplicates, for $m_i = m = 10$, starting from the initial solution (1075, 250). Table 2 summarizes the performance results of the different heuristics, in terms of the average, minimum and maximum of the objective and constraint values. Note that the OptQuest version embedded in ARENA 11 does not return the estimated value of the constraint outputs; hence, the corresponding entries in the table are set to "NA" (not available).

The detailed results from Table 1 show that the quality of the optimal solution found by our heuristic consistently outperforms the solutions provided by the Optquest version embedded in ARENA 11. The ARENA 12 version succeeds in finding a superior optimum (with lower objective value) in only 3 out of 10 macroreplicates (i.e., macroreplicates 2, 3, and 7). Table 2, however, shows that the quality of the solution found by Optquest in ARENA 12 varies more widely over the macroreplicates. The performance of our heuristic varies less over the macroreplicates; it clearly moves faster towards the constraint boundary.

As mentioned above, the use of a low relative precision criterion saves simulation time and effort, because it requires fewer replicates per design point. This does not seem to impact the performance of our heuristic substantially. Contrary to our expectations, it—in general—causes the heuristic to need fewer points to simulate before meeting the stopping criterion. While Table 2 seems to suggest that the quality of the optimum tends to be higher when lower precision is used (i.e., lower values for the objective are obtained), these results should be interpreted with caution, because the estimates for objective and constraint values are less reliable.

Table 1: Results for the (s, S) inventory model

macrorepl	Heuristic	$\mathbf{d_{opt}}$	E(TC)	E(δ)	i_{max}	rank $\mathbf{d_{opt}}$
1	OptQuest (Arena 12) $m_i = 10$	(1009,287)	716.16	0.0828	60	60
	OptQuest (Arena 11) $m_i = 10$	(1047,108)	660.91	NA	60	59
	DOE-Kri-MP $m_i = 10$	(1043,70)	638.34	0.0992	60	14
	DOE-Kri-MP $\gamma = 0.15$	(1021,114)	640.09	0.0991	54	41
2	OptQuest (Arena 12) $m_i = 10$	(1027,84)	632.42	0.0993	68	21
	OptQuest (Arena 11) $m_i = 10$	(1047,97)	656.79	NA	68	68
	DOE-Kri-MP $m_i = 10$	(1043,70)	638.34	0.0992	68	14
	DOE-Kri-MP $\gamma = 0.15$	(1061,31)	634.74	0.0999	61	27
3	OptQuest (Arena 12) $m_i = 10$	(1050,44)	632.77	0.0993	81	63
	OptQuest (Arena 11) $m_i = 10$	(1047,85)	650.60	NA	81	81
	DOE-Kri-MP $m_i = 10$	(1043,70)	638.34	0.0992	81	14
	DOE-Kri-MP $\gamma = 0.15$	(1062,29)	634.19	0.0999	58	40
4	OptQuest (Arena 12) $m_i = 10$	(1061,191)	713.74	0.0746	69	69
	OptQuest (Arena 11) $m_i = 10$	(1047,103)	657.35	NA	69	69
	DOE-Kri-MP $m_i = 10$	(1057,41)	636.36	0.0999	69	48
	DOE-Kri-MP $\gamma = 0.15$	(1076,12)	637.39	0.0993	61	40
5	OptQuest (Arena 12) $m_i = 10$	(1129,35)	700.19	0.0711	81	80
	OptQuest (Arena 11) $m_i = 10$	(1047,99)	657.90	NA	81	80
	DOE-Kri-MP $m_i = 10$	(1043,70)	638.34	0.0992	81	14
	DOE-Kri-MP $\gamma = 0.15$	(1041,73)	638.04	0.0983	66	12
6	OptQuest (Arena 12) $m_i = 10$	(1002,209)	671.28	0.0936	66	66
	OptQuest (Arena 11) $m_i = 10$	(1047,95)	655.28	NA	66	66
	DOE-Kri-MP $m_i = 10$	(1043,70)	638.34	0.0992	66	14
	DOE-Kri-MP $\gamma = 0.15$	(1047,58)	635.20	0.0998	62	22
7	OptQuest (Arena 12) $m_i = 10$	(1027,84)	632.02	0.0998	73	25
	OptQuest (Arena 11) $m_i = 10$	(1047,99)	657.86	NA	73	70
	DOE-Kri-MP $m_i = 10$	(1043,70)	638.34	0.0992	73	14
	DOE-Kri-MP $\gamma = 0.15$	(1041,73)	638.04	0.0983	54	12
8	OptQuest (Arena 12) $m_i = 10$	(1054,59)	642.44	0.0956	75	75
	OptQuest (Arena 11) $m_i = 10$	(1047,122)	667.34	NA	75	75
	DOE-Kri-MP $m_i = 10$	(1046,59)	635.63	0.0996	75	55
	DOE-Kri-MP $\gamma = 0.15$	(1057,40)	634.62	0.0990	55	27
9	OptQuest (Arena 12) $m_i = 10$	(1061,69)	714.79	0.0739	69	69
	OptQuest (Arena 11) $m_i = 10$	(1047,92)	652.85	NA	69	69
	DOE-Kri-MP $m_i = 10$	(1043,70)	638.34	0.0992	69	14
	DOE-Kri-MP $\gamma = 0.15$	(1039,73)	636.24	0.0992	71	38
10	OptQuest (Arena 12) $m_i = 10$	(999,185)	655.21	0.0972	63	63
	OptQuest (Arena 11) $m_i = 10$	(1047,109)	661.05	NA	63	63
	DOE-Kri-MP $m_i = 10$	(1043,70)	638.34	0.0992	63	14
	DOE-Kri-MP $\gamma = 0.15$	(1049,54)	636.14	0.0999	67	40

Table 2: Performance summary for the (s, S) inventory model

Heuristic	Total Costs TC			Disservice rate δ		
	average	max	min	average	max	min
OptQuest (Arena 12) $m_i = 10$	671.11	716.17	632.02	0.0887	0.0998	0.0711
OptQuest (Arena 11) $m_i = 10$	657.80	667.34	650.60	NA	NA	NA
DOE-Kri-MP $m_i = 10$	637.88	638.35	635.63	0.0993	0.0999	0.0992
DOE-Kri-MP $\gamma = 0.15$	636.47	640.10	634.19	0.0993	0.0999	0.0983

5.2 The call-center staffing problem

In this section, we summarize the results for the call-center simulation model used in Kelton et al. (2007). Other types of simulation models for call center staffing—applying different optimization heuristics—are presented by Avramidis et al. (2007) and Atlason et al. (2008), who also review related publications.

The telephone call-center handles three types of calls (Kelton et al. 2007, p. 195): (i) technical support, (ii) sales, and (iii) order-status checking. Calls arrive according to a nonstationary Poisson process. Customers call a central telephone number, which feeds a number of trunk lines. Next, customers choose from the three options listed under (i), (ii), and (iii) above. If customers select (i), then they must next select whether they want technical support for product 1, 2, or 3. If they select (iii), a computer handles their order-status checking; however, after receiving this computerized information, the customer may ask for a sales person—but then the customer gets lower priority than customers of type (ii). If customers call in when all trunk lines are busy, they get a busy signal and cannot enter the system.

The center opens at 8 AM and closes at 6 PM; after 6 PM a smaller staff serves any remaining calls. So each simulation run starts and ends with an empty system. Staffing levels vary over the day (Kelton et al. 2007, p. 226). Staff is specialized: most personnel can handle only one type of call, i.e., either sales and order-status checking calls (Sales staff), or technical calls for a given product type (Tech1, Tech2 or Tech3 staff); some operators are able to handle all technical calls (Tech All staff). Some technical calls are so special that they require handling by people outside the call center; when these people call back to a technical staff member, that member calls the customer—with priority over incoming customer calls.

Additional staff may be hired (Sales, Tech 1, Tech2, Tech3 or Tech All) for the time period between noon and 4 PM (the busiest period). Moreover, additional trunk lines may be obtained. Trunk lines are available throughout the day. Kelton et al. (2007) make each simulated day identically and independently distributed, so one day is one run.

In this example, the decision variables in our equation (1) are the total number of trunk lines to be hired (d_1), along with the additional staffing levels for each of the five staff types (d_2 = Tech1, d_3 = Tech2, d_4 = Tech3, d_5 = TechAll, d_6 = Sales), in order to minimize the Total Cost TC incurred per week (which includes the salaries of all staff, the cost for hiring all trunk lines, and a separate penalty cost per minute that customers of a specific type wait on hold). There is a single output constraint, namely the Percent Rejected Calls δ should be below 5%. The number of trunk lines d_1 must satisfy the box constraint

$$26 \leq d_1 \leq 50. \tag{23}$$

Moreover, the personnel inputs must satisfy the linear constraint

$$d_2 + d_3 + d_4 + d_5 + d_6 \leq 15. \tag{24}$$

All six input variables ($k = 6$) should be integers.

Table 3 shows the results of our heuristic for 10 macroreplicates. The number of replicates per input combination is either fixed at $m_i = m = 110$ or determined by a relative precision requirement with $\gamma = 0.15$. The table also shows the OptQuest results for the same 10 macroreplicates, for $m_i = m = 110$, starting from the initial solution (29, 3, 3, 3, 3, 3) (as suggested in Kelton et al. 2007). Table 4 summarizes the performance results of the different heuristics.

Table 3 shows that our heuristic outperforms OptQuest (both in ARENA 11 and ARENA 12) in every macroreplication. Table 4 shows that our heuristic moves faster towards the constraint boundary than OptQuest does, and obtains high-quality solutions with fewer design points.

We recommend the relative precision criterion because it controls the magnitude of the ”noise” relative to the ”signal”; moreover, it gives replication numbers that vary with the input combination (we assume that the noise does not remain constant as we move over the search area).

6 Conclusions and future research

In this article, we developed a heuristic for constrained optimization of random simulation models. One simulation output is selected as the objective to be minimized, while the other $(r - 1)$ outputs must satisfy prespecified target values. The (deterministic) simulation inputs must meet prespecified constraints, including integer constraints. The heuristic combines features from (i) design of experiments, (ii) Kriging modeling, to approximate the global I/O functions per output type implied by the underlying simulation model, and (iii) INLP, to estimate the

Table 3: Results for the call center model

macrorepl	Heuristic	$\mathbf{d_{opt}}$	E(TC)	E(δ)	i_{max}	rank $\mathbf{d_{opt}}$
1	OptQuest (Arena 12) $m_i = 110$	(28,1,2,1,0,2)	22034.52	3.94	62	49
	OptQuest (Arena 11) $m_i = 110$	(26,2,1,1,0,3)	21449.08	NA	62	56
	DOE-Kri-MP $m_i = 110$	(26,0,1,1,2,2)	21424.02	4.72	62	59
	DOE-Kri-MP $\gamma = 0.15$	(26,1,0,1,2,3)	21586.87	4.80	53	38
2	OptQuest (Arena 12) $m_i = 110$	(28,0,1,2,1,3)	21826.21	3.14	63	62
	OptQuest (Arena 11) $m_i = 110$	(26,2,1,1,0,3)	21621.16	NA	63	56
	DOE-Kri-MP $m_i = 110$	(26,0,0,0,3,2)	21323.31	4.96	63	47
	DOE-Kri-MP $\gamma = 0.15$	(26,1,2,1,0,3)	21562.87	4.77	53	38
3	OptQuest (Arena 12) $m_i = 110$	(28,0,2,2,0,2)	21930.79	3.70	62	51
	OptQuest (Arena 11) $m_i = 110$	(26,2,1,1,0,4)	21741.81	NA	62	55
	DOE-Kri-MP $m_i = 110$	(26,0,0,0,3,3)	21261.21	4.92	62	61
	DOE-Kri-MP $\gamma = 0.15$	(27,1,1,1,1,2)	21578.75	4.09	71	62
4	OptQuest (Arena 12) $m_i = 110$	(28,0,2,1,0,3)	22122.93	4.27	67	55
	OptQuest (Arena 11) $m_i = 110$	(26,2,2,1,0,2)	21633.66	NA	67	56
	DOE-Kri-MP $m_i = 110$	(27,0,1,1,1,2)	21194.63	4.47	67	59
	DOE-Kri-MP $\gamma = 0.15$	(27,0,1,0,2,3)	21412.04	4.36	63	61
5	OptQuest (Arena 12) $m_i = 110$	(28,1,2,1,0,2)	22113.72	3.85	66	51
	OptQuest (Arena 11) $m_i = 110$	(26,2,1,1,0,3)	21650.91	NA	66	56
	DOE-Kri-MP $m_i = 110$	(26,0,0,1,2,3)	21546.72	4.99	66	54
	DOE-Kri-MP $\gamma = 0.15$	(26,1,1,0,2,3)	21638.75	4.80	74	60
6	OptQuest (Arena 12) $m_i = 110$	(28,0,2,2,0,3)	22282.48	3.67	70	53
	OptQuest (Arena 11) $m_i = 110$	(26,2,1,1,0,3)	21682.38	NA	70	56
	DOE-Kri-MP $m_i = 110$	(26,1,1,1,1,2)	21443.62	4.71	70	64
	DOE-Kri-MP $\gamma = 0.15$	(26,0,2,1,1,7)	22736.19	4.60	57	27
7	OptQuest (Arena 12) $m_i = 110$	(27,1,2,1,0,3)	21976.32	4.17	93	79
	OptQuest (Arena 11) $m_i = 110$	(26,2,1,1,0,3)	21614.37	NA	93	56
	DOE-Kri-MP $m_i = 110$	(26,0,1,0,3,2)	21522.46	4.66	93	58
	DOE-Kri-MP $\gamma = 0.15$	(26,1,2,1,0,2)	20327.66	3.82	78	57
8	OptQuest (Arena 12) $m_i = 110$	(27,1,1,1,0,2)	21452.64	4.98	128	84
	OptQuest (Arena 11) $m_i = 110$	(26,1,2,1,0,2)	21398.62	NA	128	96
	DOE-Kri-MP $m_i = 110$	(26,1,1,0,2,2)	21299.95	4.54	128	96
	DOE-Kri-MP $\gamma = 0.15$	(26,0,0,1,2,3)	21020.93	4.85	60	52
9	OptQuest (Arena 12) $m_i = 110$	(28,1,1,1,0,2)	21933.02	4.27	87	79
	OptQuest (Arena 11) $m_i = 110$	(26,2,1,1,0,3)	21564.11	NA	87	56
	DOE-Kri-MP $m_i = 110$	(27,0,1,1,1,3)	21231.51	4.02	87	81
	DOE-Kri-MP $\gamma = 0.15$	(27,0,1,1,1,2)	20915.47	4.31	101	100
10	OptQuest (Arena 12) $m_i = 110$	(28,1,2,1,0,2)	22078.51	3.91	75	61
	OptQuest (Arena 11) $m_i = 110$	(26,2,2,1,0,2)	21854.64	NA	75	56
	DOE-Kri-MP $m_i = 110$	(26,0,1,2,1,3)	21620.26	4.76	75	57
	DOE-Kri-MP $\gamma = 0.15$	(27,0,1,0,2,3)	21200.80	4.24	54	47

Table 4: Performance summary for the call center model

Heuristic	TC			δ		
	average	max	min	average	max	min
OptQuest (Arena 12) $m_i = 10$	21975.11	22282.48	21452.64	3.99	4.98	3.14
OptQuest (Arena 11) $m_i = 10$	21621.07	21854.64	21398.62	NA	NA	NA
DOE-Kri-MP $m_i = 10$	21386.77	21620.26	21194.63	4.68	4.99	4.03
DOE-Kri-MP $\gamma = 0.15$	21398.04	22736.19	20327.67	4.47	4.85	3.83

optimal solution based on the Kriging models. The resulting stepwise procedure contains elements of both global and local search. We proposed solutions for a number of tactical problems, such as the validation of the Kriging metamodels, bootstrapping in case of CRN and unequal replication numbers, and statistically testing whether a newly obtained solution yields a significant improvement. We applied our heuristic to various simulation models, and compared its performance with OptQuest. The results suggest that our heuristic gives an optimum solution much faster than OptQuest does.

Though we selected two test cases from the fields of manufacturing and logistics, our heuristic is general in the sense that it is applicable to problem formulations that share the same characteristics: random objective, random output constraints, and integer input constraints. Our heuristic is flexible in the sense that it can be easily modified to represent other settings such as deterministic simulation. Moreover, the procedures implemented in steps 1 (initial space-filling design), 3 (fitting the metamodels), 6 and 8 (estimating optima through INLP) may be replaced by better procedures as the knowledge in each of the related fields (DOE, Kriging, INLP) evolves. Other types of global metamodels may also be considered: Classification And Regression Trees (CART), Generalized Linear Models (GLM), Multivariate Adaptive Regression Splines (MARS), Neural Networks (NN), nonlinear regression models, nonparametric regression analysis, radial functions, rational functions, splines, support vector regression, symbolic regression, and wavelets. If readers prefer such an alternative metamodel, then they can replace our Kriging metamodel by their favorite metamodel and adapt our heuristic.

Our future research will concentrate on further improvement of our heuristic. We do not recommend the use of a fixed number of replicates per point; i.e., we recommend the use of the relative precision criterion. The use of the signal/noise criterion in Kriging metamodeling requires more research. We may also study variations on the acceptance of constrained simulation outputs; i.e., we may accept a point with average outputs that give nonsignificant violations of the threshold values (currently, we accept only points with average outputs that do not exceed the threshold values). We might add a postprocessing step (to be run after our heuristic stops), aimed at ordering high-quality solutions (see Kelly 2002). Such a step may use Multiple Ranking and Selection procedures; see Kleijnen (2008a), p. 102. Selecting such a procedure is discussed by Branke et al. (2007). Moreover, the current heuristic is based on ordinary Kriging, which is univariate and was originally developed for deterministic simulation. Multivariate Kriging that accounts for stochastic simulation with replications and output variances that vary with the simulated input combinations, may further improve performance. Unfortunately, there is no software yet for multivariate stochastic Kriging—free and well-documented like DACE. Finally, we would like to adapt our heuristic for *continuous* inputs, replacing the INLP code by some NLP code; an advantage of Kriging is that in this case the metamodel also gives an estimate of the gradient, at no extra cost (see Biles et al. 2007).

Finally, we would like to see the simulation community develop standard testbeds for different heuristics for simulation optimization. Such testbeds are popular in Mathematical Programming, but they do not yet exist in our field. A start is made with the website http://www.simopt.org (based on Pasupathy and Henderson 2006), but its test problems are not available as code so they require recoding, which is a major source of errors.

References

Abspoel, S.J., L.F.P. Etman, J. Vervoort, R.A. van Rooij, A.J.G. Schoofs, J.E. Rooda. 2001. Simulation based optimization of stochastic systems with integer design variables by sequential multipoint linear approximation. *Struct. Multidiscip. Optim.* **22** 125–138.

Andradóttir, S. 1999. Accelerating the convergence of random search methods for discrete stochastic optimization. *ACM Transactions on Modeling and Computer Simulation* **9** 349–380.

Andradóttir, S. 2002. Simulation Optimization: Integrating research and practice. *INFORMS J. Comput.* **14** 216–219.

Angün, E., D. den Hertog, G. Gürkan, J.P.C. Kleijnen. 2002. Response surface methodology revisited. E. Yucesan, C.H. Chen, J.L. Snowdon and J.M. Charnes, eds. *Proc. 2002 Winter Simulation Conference* 377–383.

Ankenman, B., B.L. Nelson, J. Staum. 2008. Stochastic Kriging for simulation Metamodeling. *Oper.Res.* (conditionally accepted).

Atlason, J., M.A. Epelman, S.G. Henderson. 2008. Optimizing call center staffing using simulation and analytic center cutting-plane methods. *Management Sci.* **54** 295–309.

Avramidis, A.N., O. Pisacane, M. Gendreau, P. L' Ecuyer. 2007. Simulation-based optimization of agent scheduling in a multiskill call center. J. Ottjes, H. Veeke, eds. *5th International Industrial Simulation Conference, ISC'2007*, Delft, Netherlands, 255-263.

Bashyam, S., M. C. Fu. 1998. Optimization of (s, S) inventory systems with random lead times and a service level constraint. *Management Sci.* **44** 243–256.

Bates, R.A, R.S., Kenett, D.M. Steinberg, H.P. Wynn. 2006. Achieving robust design from computer simulations. *Quality Technology and Quantitative Management* **3** 161–17.

Beyer, H., B. Sendhoff. 2007. Robust optimization—a comprehensive survey. *Comput. Meth. Appl. Mech. Eng.* **196** 3190–3218.

Biles, W.E., J.P.C. Kleijnen, W.C.M. van Beers, I. van Nieuwenhuyse. 2007. Kriging metamodels in constrained simulation optimization: an explorative study. S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, R. R. Barton, eds. *Proc. 2007 Winter Simulation Conference* 355–362.

Branke, J., S.E. Chick, C. Schmidt. 2007. Selecting a selection procedure. *Management Sci.* **53** 1916–1932.

Cezik, M.T., P. L' Ecuyer. 2008. Staffing multiskill call centers via linear programming and simulation. *Management Sci.* **54** 310–323.

Chilès J-P., P. Delfiner. 1999. *Geostatistics: modeling spatial uncertainty.* Wiley, New York.

Cioppa, T.M., T.W. Lucas. 2007. Efficient nearly orthogonal and space-filling Latin hypercubes. *Technometrics* **49** 45–55.

Davis, E., M. Ierapetritou. 2007. A Kriging based method for the solution of mixed-integer nonlinear programs containing black-box functions. *J. Glob. Optim.* (in press)

Deflandre, D., J.P.C. Kleijnen. 2003. Statistical analysis of random simulations: bootstrap tutorial. *Simulation News Europe* **38/39** 29–34.

Den Hertog, D., J.P.C. Kleijnen, A.Y.D. Siem. 2006. The correct Kriging variance estimated by bootstrapping. *J. Oper. Res. Soc.* **57** 400–409.

Driessen, L., R.C.M. Brekelmans, M. Gerichhausen, H. Hamers, D. den Hertog. 2006a. Why methods for optimization problems with time consuming function evaluations and integer variables should use global approximation models. CentER Discussion Paper 2006-04, CentER, Tilburg University, Tilburg, the Netherlands.

Driessen, L., R. Brekelmans, H. Hamers, D. den Hertog. 2006b. On D-optimality based trust regions for black-box optimization problems. *Struct. Multidiscip. Optim.* **31** 40–48.

Efron, B., R.J. Tibshirani. 1993. *An introduction to the bootstrap.* Chapman & Hall, New York.

Fu, M.C. 2002. Optimization for Simulation: Theory vs. Practice, *Informs J. Comput.* **14** 192–215.

Fu, M.C. 2007. Are we there yet? The marriage between simulation & optimization. *OR/MS Today* **34** 16–17.

Fu, M.C., F.W. Glover, J. April. 2005. Simulation optimization: a review, new developments, and applications. M.E. Kuhl, N.M. Steiger, F.B. Armstrong, J.A. Joines, eds. *Proc 2005 Winter Simulation Conference.* Piscataway, New Jersey, 83–95.

Greenberg, H.J., T. Morisson. 2008. Robust optimization. A.R. Ravindran, ed. *Handbook of Operations Research and Management Science.* CRC Press, Boca Raton, Florida.

Huang, D., T.T. Allen, W. Notz, N. Zheng. 2006. Global optimization of stochastic black-box systems via sequential Kriging meta-models. *J. Glob. Optim.* **34** 441–466.

Jones, D.R., M. Schonlau, W.J. Welch. 1998. Efficient global optimization of expensive black-box functions. *J. Glob. Optim.* **13** 455–492.

Joseph, V. R., Hung, Y., and Sudjianto, A. 2008. Blind Kriging: a new method for developing metamodels. *J. Mech. Des.* , **130**, in press.

Kelly, J.P. 2002. Simulation optimization is evolving, *Informs J. Comput.* **14** 223–225.

Kelton, W.D., R.P. Sadowski, D.T. Sturrock. 2007. *Simulation with Arena.* 4th ed. McGraw-Hill, Boston.

Kenett, R., D. Steinberg. 2006. New frontiers in design of experiments. *Quality Progress* 61–65.

Kleijnen, J.P.C. 2008a. *Design and analysis of simulation experiments.* Springer Science + Business Media.

Kleijnen, J.P.C. 2008b. Factor screening in simulation experiments: review of sequential bifurcation. Working Paper, Tilburg University, Tilburg, Netherlands.

Kleijnen, J.P.C. 2009. Kriging metamodeling in simulation: a review. *Eur. J. Oper. Res.* **192** 707–716.

Kleijnen, J.P.C., D. Deflandre. 2006. Validation of regression metamodels in simulation: bootstrap approach. *Eur. J. Oper. Res.* **170** 120–131.

Kleijnen, J.P.C., W.C.M. van Beers. 2004. Application-driven sequential designs for simulation experiments: Kriging metamodeling. *J. Oper. Res. Soc.* **55** 876–883.

Kleijnen, J.P.C., J. Wan. 2007. Optimization of simulated systems: OptQuest and alternatives. *Simul. Model. Pract. Theory* **15** 354–362.

Law, A.M. 2007. *Simulation modeling and analysis.* 4th ed. McGraw-Hill, Boston.

Lophaven, S.N., H.B. Nielsen, J. Sondergaard. 2002. DACE: a Matlab Kriging toolbox, version 2.0. IMM Technical University of Denmark, Lyngby.

Martin, J.D., T.W. Simpson. 2004. A Monte Carlo simulation of the Kriging model. *10th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA-2004-4483.

Martin, J.D., T.W. Simpson. 2005. Use of Kriging models to approximate deterministic computer models. *AIAA J.* **43**(4) 853–863.

Miller, R.G. 1981. *Simultaneous statistical inference.* Rev. 2nd ed. Springer, New York.

Oden, J.T. 2006. *Revolutionizing engineering science through simulation.* National Science Foundation (NSF), Blue Ribbon Panel on Simulation-Based Engineering Science.

Pasupathy, R., S. G. Henderson. 2006. A testbed of simulation-optimization problems. L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, R. M. Fujimoto, eds. *Proc. 2006 Winter Simulation Conference*. 255–263.

Pintér, J.D. 2007. Computational global optimization: state-of-the-art and perspectives. *OR/MS Today* **34**(5) 18–19.

Rosen, S.C., C.M. Harmonosky, M.T. Traband. 2007. Optimization of systems with multiple performance measures via simulation: survey and recommendations. *Comput. Ind. Eng.* **54**(2) 327–339.

Van Beers, W.C.M., J.P.C. Kleijnen. 2008. Customized sequential designs for random simulation experiments: Kriging metamodeling and bootstrapping. *Eur. J. Oper. Res.* **186**(3) 1099–1113.

A Computational results for the "toy problem"

The outputs of the so-called toy problem are generated by second-order polynomials augmented with noise. The problem is based on Angün et al. (2002); however, to increase the integer-valued search area, we transform the original decision variables z_1 and z_2 into d_1 and d_2 as follows:

$$z_1 = \frac{d_1}{10},\ z_2 = \frac{d_2}{10} - 2,\ 0 \leq d_1 \leq 30,\ 0 \leq d_2 \leq 30. \tag{25}$$

This gives the following optimization problem:

$$\begin{array}{ll} \text{minimize} & E(w_0) = E[5(\frac{d_1}{10} - 1)^2 + (\frac{d_2}{10} - 7)^2 + 4\frac{d_1}{10}(\frac{d_2}{10} - 2) + e_0] \\ \text{subject to} & E(w_1) = E[(\frac{d_1}{10} - 3)^2 + (\frac{d_2}{10} - 2)^2 + \frac{d_1}{10}(\frac{d_2}{10} - 2) + e_1] \leq 4 \\ & E(w_2) = E[\left(\frac{d_1}{10}\right)^2 + 3\left(\frac{d_2}{10} - 0.939\right)^2 + e_2] \leq 9 \\ & 0 \leq d_1 \leq 30,\ 0 \leq d_2 \leq 30;\ d_1, d_2 \in \mathbb{N}. \end{array} \tag{26}$$

The random errors e_0, e_1, and e_2 are normally distributed with means 0, variances $\sigma_{0,0} = 0.7500$ (so $\sigma_0 = \sqrt{0.75} = 0.87$), $\sigma_{1,1} = 0.0169$, $\sigma_{2,2} = 0.1200$, and correlations $\rho_{0,1} = 0.82$, $\rho_{0,2} = 0.30$, $\rho_{1,2} = -0.07$. Because the simulation noise is purely additive in this problem, we do not use CRN to sample the multivariate normal noise at different input combinations. The true optimum for this integer problem is (12, 24), with $E(w_0) = 23.28$, $E(w_1) = 3.88$, $E(w_2) = 7.8436$.

Table 5 shows the results of our heuristic for 10 macroreplicates. The number of replicates per input combination is either fixed at $m_i = m = 110$ or determined by a relative precision requirement with $\gamma = 0.15$. This table also shows the OptQuest results for the same 10 macroreplicates with $m_i = m = 110$, starting from the initial solution (15, 15).

ARENA Version 12 finds the optimum after only 9 points in each macroreplicate. Strangely enough, Version 11 gives very different results; it requires between 55 and 90 points! Our heuristic finds the optimum after 12 to 30 points; our heuristic uses a pilot sample with 9 points, which explains why it needs more than 9 points to find the optimum.

Table 5: Results for the toy problem obtained by the heuristic

macrorepl	Heuristic	$\mathbf{d_{opt}}$	$E(w_0)$	$E(w_1)$	$E(w_2)$	i_{max}	rank $\mathbf{d_{opt}}$
1	OptQuest (Arena 12) $m_i = 110$	(12,24)	23.31	3.89	7.82	105	9
	OptQuest (Arena 11) $m_i = 110$	(13,24)	23.504	NA	NA	105	55
	DOE-Kri-MP $m_i = 110$	(12,24)	23.31	3.89	7.82	105	12
	DOE-Kri-MP $\gamma = 0.15$	(11,23)	22.64	3.93	6.72	56	14
2	OptQuest (Arena 12) $m_i = 110$	(12,24)	23.41	3.89	7.85	103	9
	OptQuest (Arena 11) $m_i = 110$	(12,24)	23.41	NA	NA	103	90
	DOE-Kri-MP $m_i = 110$	(12,24)	23.41	3.89	7.85	103	13
	DOE-Kri-MP $\gamma = 0.15$	(13,24)	22.83	3.44	7.97	62	30
3	OptQuest (Arena 12) $m_i = 110$	(12,24)	23.34	3.89	7.85	93	9
	OptQuest (Arena 11) $m_i = 110$	(12,24)	23.34	NA	NA	93	90
	DOE-Kri-MP $m_i = 110$	(12,24)	23.34	3.89	7.85	93	15
	DOE-Kri-MP $\gamma = 0.15$	(12,24)	22.49	3.79	7.74	61	13
4	OptQuest (Arena 12) $m_i = 110$	(12,24)	23.31	3.88	7.81	104	9
	OptQuest (Arena 11) $m_i = 110$	(13,24)	23.84	NA	NA	104	55
	DOE-Kri-MP $m_i = 110$	(12,24)	23.31	3.88	7.81	104	12
	DOE-Kri-MP $\gamma = 0.15$	(12,24)	23.05	3.82	7.86	60	12
5	OptQuest (Arena 12) $m_i = 110$	(12,24)	23.28	3.89	7.79	105	9
	OptQuest (Arena 11) $m_i = 110$	(13,24)	23.67	NA	NA	105	55
	DOE-Kri-MP $m_i = 110$	(12,24)	23.28	3.89	7.79	105	13
	DOE-Kri-MP $\gamma = 0.15$	(11,23)	22.67	3.93	6.95	47	18
6	OptQuest (Arena 12) $m_i = 110$	(12,24)	23.22	3.87	7.81	71	9
	OptQuest (Arena 11) $m_i = 110$	(13,24)	23.65	NA	NA	71	55
	DOE-Kri-MP $m_i = 110$	(12,24)	23.22	3.87	7.81	71	12
	DOE-Kri-MP $\gamma = 0.15$	(12,24)	23.08	3.85	7.86	49	12
7	OptQuest (Arena 12) $m_i = 110$	(12,24)	23.28	3.88	7.83	91	9
	OptQuest (Arena 11) $m_i = 110$	(13,24)	23.64	NA	NA	91	55
	DOE-Kri-MP $m_i = 110$	(12,24)	23.28	3.88	7.83	91	16
	DOE-Kri-MP $\gamma = 0.15$	(12,24)	23.21	3.86	7.66	63	15
8	OptQuest (Arena 12) $m_i = 110$	(12,24)	23.29	3.88	7.84	98	9
	OptQuest (Arena 11) $m_i = 110$	(13,25)	23.15	NA	NA	98	78
	DOE-Kri-MP $m_i = 110$	(13,25)	23.15	3.77	8.97	98	16
	DOE-Kri-MP $\gamma = 0.15$	(12,23)	23.22	3.65	6.96	56	17
9	OptQuest (Arena 12) $m_i = 110$	(12,24)	23.27	3.87	7.89	75	9
	OptQuest (Arena 11) $m_i = 110$	(13,24)	23.55	NA	NA	75	55
	DOE-Kri-MP $m_i = 110$	(12,24)	23.27	3.87	7.89	75	11
	DOE-Kri-MP $\gamma = 0.15$	(11,23)	23.08	3.94	6.81	55	12
10	OptQuest (Arena 12) $m_i = 110$	(12,24)	23.20	3.86	7.82	98	9
	OptQuest (Arena 11) $m_i = 110$	(13,25)	23.33	NA	NA	98	78
	DOE-Kri-MP $m_i = 110$	(12,24)	23.20	3.86	7.82	98	11
	DOE-Kri-MP $\gamma = 0.15$	(12,24)	23.07	3.79	8.03	54	14

Table 6: Performance summary for the toy problem

Heuristic	w_0			w_1			w_2		
	average	max	min	average	max	min	average	max	min
OptQuest (Arena 12) $m_i = 10$	23.29	23.41	23.20	3.88	3.89	3.86	7.83	7.89	7.79
OptQuest (Arena 11) $m_i = 10$	23.39	23.84	22.26	NA	NA	NA	NA	NA	NA
DOE-Kri-MP $m_i = 10$	23.28	23.41	23.15	3.87	3.89	3.77	7.94	8.97	7.79
DOE-Kri-MP $\gamma = 0.15$	22.93	23.22	22.49	3.80	3.94	3.44	7.46	8.03	6.72