Constrained Optimization in Expensive Simulation: Novel Approach

Jack P.C. Kleijnen a), Wim van Beers b) and Inneke van Nieuwenhuyse c)

*a) Department of Information Management, Tilburg University, Postbox 90153,
5000 LE Tilburg, Netherlands, e-mail: kleijnen@uvt.nl,*

<http://center.uvt.nl/staff/kleijnen/>; phone 31-13-466-2029; fax: 31-13-466-3069

*b) Department of Information Management, Tilburg University, Postbox 90153,
5000 LE Tilburg, Netherlands, e-mail: W.C.M.vanBeers@uvt.nl*

*c) Research Center for Operations Management, Department of Decision Sciences
and Information Management, K.U. Leuven, Leuven, Belgium, email :
inneke.vannieuwenhuyse@econ.kuleuven.be*

Version 2: 20 April 2009

Abstract

This article presents a novel heuristic for constrained optimization of computationally expensive random simulation models. One output is selected as objective to be minimized, while other outputs must satisfy given threshold values. Moreover, the simulation inputs must be integer and satisfy linear or nonlinear constraints. The heuristic combines (i) sequentialized experimental designs to specify the simulation input combinations, (ii) Kriging (or Gaussian process or spatial correlation modeling) to analyze the global simulation input/output data resulting from these designs, 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 call-center simulation, and compared with the popular commercial heuristic OptQuest embedded in the Arena versions 11 and 12. In these two applications the novel heuristic outperforms OptQuest in terms of number of simulated input combinations and quality of the estimated optimum.

Key words: Simulation; Global Optimization; Heuristics; Kriging; Bootstrap

1 Introduction

In this article, we present a novel heuristic – which we call "DOE-Kri-MP" ("Design Of Experiments-Kriging-Mathematical Programming", pronounced

as "doe-krimp") – for constrained optimization of random simulation models. The importance of optimization, and the crucial role of simulation are widely recognized – especially in engineering (see Oden, 2006). Though many other fields (e.g., logistics, supply chain management) have also embraced simulation, its use in these areas has focused on system evaluation and what-if analysis. The interest in simulation for optimization purposes has undoubtedly been stimulated by the availability of optimization tools in commercial software packages, such as OptQuest in the Arena simulation package (Kelton et al., 2007).

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 large. Running the simulation model may be computationally expensive (e.g., a single run of a crash simulation at Ford took ten days); in such applications, the computational time of our heuristic is negligible. Simulation models for logistics systems present some additional problems; typically, these are discrete-event models where the inputs are non-negative integers. Historically, research efforts in simulation optimization have focused on continuous inputs (Fu, 2002). Moreover, many simulation models give multiple outputs; the optimization problem needs to reflect the trade-offs among these outputs. This problem can be approached in various ways (see the recent surveys by Beyer and Sendhoff, 2007; Greenberg and Morrison, 2008; Rosen et al., 2007). We follow the classic *mathematical programming approach* that selects one output as "the" objective, while requiring the other outputs to satisfy given threshold values. Good algorithms for handling random constraints are scarce (Fu, 2002).

Our DOE-Kri-MP heuristic combines methodologies from three research fields: (i) Design of Experiments (DOE), (ii) Kriging metamodeling, and (iii) Mathematical Programming (MP). We extend classic DOE with sequential design, and allow for possibly non-linear input constraints. We extend Kriging with validation of the Kriging metamodel through bootstrapping allowing for Common Random Numbers (CRN) and unequal replication numbers per design point. We extend MP, forcing it to give a point that is not yet part of the design (so-called "next best point"). We apply our heuristic to a well-known academic (s, S) inventory simulation and a realistic call-center simulation model, and compare its performance with the most popular commercial heuristic in simulation optimization, namely OptQuest. Our empirical results show that our heuristic outperforms OptQuest in terms of number of simulated input combinations and quality of the estimated optimum. Such performance is desirable in case of computationally expensive simulation experiments (Kelly, 2002).

The remainder of this article is organized as follows. Section 2 presents an Integer Non-Linear Programming (INLP) formalization of our type of optimization

problems. Section 3 summarizes the basics of Kriging metamodeling. Section 4 details our heuristic. Section 5 compares computational results for DOE-Kri-MP and OptQuest in two test problems, namely an (s, S) inventory optimization model, and a call-center staffing problem (the online appendix gives results for a "toy problem", namely second-order polynomial response functions augmented with multivariate normal noise). Finally, 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*:

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

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

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

where the right-hand coefficients $a_{h'}$ are given. The expected values (means) in the objective function and the constraint functions are estimated through simulation; the simulation estimates are denoted by the sample averages $\overline{w_h}(\mathbf{d})$ ($h = 0, \dots, r - 1$) (see (14) below). The k deterministic inputs d_j ($j = 1, \dots, k$) are required to be non-negative integers:

$$d_j \in \mathbb{N} \quad (3)$$

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

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

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 j ($j = 1, \dots, k$). 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 the expected total cost, which typically consists of the sum of expected inventory carrying costs and expected ordering costs. In practice, stockout costs are hard to quantify, so the stockout fraction is commonly treated as a constraint; e.g., the expected

stockout fraction (complement of the fill rate) should not exceed 5% so in (2) $a_1 = 0.05$ (the stockout fraction is related to the stockout probability, used in many analytical studies). The search area for s and S is typically limited by predetermined upper and lower bounds that form the box constraints, and the constraint $s < S$.

The research literature on stochastic simulation optimization focuses on continuous inputs. 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, 2007). Research on integer constrained optimization of stochastic simulations is scarce. Abspoel et al. (2001) use local linear approximations, so they may encounter problems when either the objective or the constraints are highly nonlinear. Moreover, Driessen et al. (2006a) demonstrate that the use of local approximations for solving integer optimization problems may easily give inferior solutions. Atlason et al. (2008) present an algorithm for constrained integer simulation optimization based on the cutting-plane method; their method, however, requires that the constraint functions be pseudoconcave. Cezik and L'Ecuyer (2008)'s algorithm runs into similar problems when exploring regions of the search area where the constraint functions are nonconcave. Davis and Ierapetritou (2007) combine Kriging metamodels for global exploration and Response Surface Modelling for local exploration, to optimize process synthesis; however, their uncertainty results only from the noise in the input variables, whereas both objective and constraints are explicit functions of the input variables.

3 Kriging metamodels

Kriging is a type of metamodel that enables adequate approximation 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 Cressie, 1993, Sacks et al., 1989, Santner et al., 2003, and the many more references in Kleijnen, 2008, 2009). Ordinary Kriging – from now on, briefly called Kriging – assumes a single deterministic output. Our objective is to apply this Kriging to a random simulation model with multiple (multivariate) outputs; i.e., we apply Kriging to the average output per simulated input combination, and we do so for each of the r types of output. Because of this randomness, we select the number of replicates such that the averages are "precise enough"; moreover, we apply CRN to obtain acceptable

signal/noise.

This Kriging makes the following two assumptions:

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

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

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

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

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

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

It can be proven that the solution of the constrained minimization problem implied by the BLUP criterion implies $\sum_{i=1}^n \lambda_i = 1$. Furthermore, the *optimal* weights are

$$\lambda = \mathbf{\Gamma}^{-1}(\gamma + \mathbf{1} \frac{1 - \mathbf{1}'\mathbf{\Gamma}^{-1}\gamma}{\mathbf{1}'\mathbf{\Gamma}^{-1}\mathbf{1}}) \quad (8)$$

where $\mathbf{\Gamma} = \text{cov}(\overline{w(\mathbf{d}_i)}, \overline{w(\mathbf{d}_{i'})})$ with $i, i' = 1, \dots, n$ is the $n \times n$ (symmetric and positive semi-definite) matrix with the covariances between the n old outputs; $\gamma = \text{cov}(\overline{w(\mathbf{d}_i)}, \overline{w(\mathbf{d}_c)})$ is the n -dimensional vector with the covariances between the n old outputs and the new output to be predicted.

Consequently, the optimal weights (8) depend on the covariances collected in $\mathbf{\Gamma}$ and γ . The model assumption (5) implies 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

$$\rho = \exp\left(-\sum_{j=1}^k \theta_j |d_{j;i} - d_{j;c}|^{p_j}\right) = \prod_{j=1}^k \exp(-\theta_j |d_{j;i} - d_{j;c}|^{p_j}) \quad (9)$$

where $|d_{j;i} - d_{j;c}|$ denotes the distance between the j^{th} input of combination i and combination c ; θ_j denotes the importance of input j (the higher θ_j is, the faster the correlation decreases with the distance); $0 < p_j \leq 2$ 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 $\mathbf{d}_c$.

Unfortunately, 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 old data. 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. (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 the predictor is an *exact interpolator*; i.e., it equals the observed average simulation output: $y(\mathbf{d}_i) = \overline{w}(\mathbf{d}_i)$ ($i = 1, \dots, n$) (which implies that all weights are zero except the weight of the old output, which has the value one).

4 The DOE-Kri-MP heuristic

Our approach is iterative; for ease of reference, we summarize the different steps in Figure 1. We shall detail each step in the following subsections. Note that our approach falls in the general framework of global metamodel-based optimization discussed by Barton and Meckesheimer (2006, Figure 18.4, right-hand side) but we consider a *constrained* optimization problem and refit the global metamodel more than once.

We start by selecting an initial space-filling design (step 1), and simulate all input combinations – called points in MP – of this design (step 2). This yields the multiple simulation outputs $\overline{w}_h(\mathbf{d}_i)$ ($h = 0, \dots, r - 1$) 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 our statistical testing proclaims one or more metamodels to be invalid, we update the design by simulating an additional input combination (step 5) from the global search area, in order to further fine-tune the r metamodels. Our heuristic then loops

Fig. 1. Overview of "DOE-Kri-MP"

back to step 3, to refit the Kriging models using all available I/O data. When we have proclaimed all r Kriging models to be valid, we use an INLP solver to estimate the location of the optimum based on the Kriging models (step 6). We check whether we have already simulated the indicated optimum (step 7). If this is indeed the case, then 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), which is the point that is optimal when all already simulated points are excluded from the optimization. We then simulate the newly found point (resulting from either step 6 or step 8), and we add its I/O data to the design (step 9). Step 10 embodies the stopping rule for our heuristic. If the last a INLP runs were unable to point out a point yielding a statistically significant improvement in the objective, then we stop our heuristic search and accept the point that so far is the best. The parameter a can be chosen by the user; we set it equal to 30 in our experiments (OptQuest uses 100). As long as the stopping condition is not met, we update the Kriging metamodels with the newly available I/O information, and continue the search.

We point out that we sequentially update the initial design: we add additional points in steps 5 and 9 respectively (a similar sequential approach is proposed in Bates et al., 2006 and Kenett and Steinberg, 2006). We select points in step 5 to improve the metamodel, which Driessen et al. (2006b) call "geometry improving"; we select points in step 9 to find the optimum. These two reasons are also related to "exploration" and "exploitation" respectively, in classic simulation optimization (Fu, 2007).

Furthermore, metamodeling approaches (unlike approaches such as OptQuest) require that the global metamodel be accurate enough to enable the INLP solver to identify clearly infeasible points (which violate the constraints on the random simulation outputs) and suboptimal points (for which the goal output values are too high). Consequently, our heuristic's design may add points throughout the entire input-feasible area, and our metamodels use all I/O values. In order to guide the INLP search, our heuristic simulates each point with a specific precision to be reasonably certain of the objective values and the (non)violation of the constraints; i.e., the essence of metamodeling in random simulation is to obtain "acceptable signal/noise".

Step 1: Select initial space-filling design

We start with a small space-filling initial design (or pilot sample), namely an $n_0 \times k$ design matrix $\mathbf{D}$ with rows $\mathbf{d}' = (d_1, \dots, d_k)$. Space-filling is a desirable property for the initial design, in order to capture information 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 (Cioppa and Lucas, 2007; Kleijnen, 2008). Jones et al. (1998)'s rule of thumb calls for a design size n_0 equal to $10k$. However, because we sequentially update our design, we limit the desired initial design size n_0 to $5 + 2k$. It seems reasonable to estimate a smooth curve for the correlation functions (9) from at least four distances between points, so a basic rule-of-thumb could be $n_0 = 5$. However, we add $2k$ combinations to ensure sufficient observations for cross-validation (see Step 4 below).

The pilot sample includes only input combinations that satisfy the input constraints in (4). The standard maximin LHS design does account for the box constraints in (4); i.e., it codes the inputs such that they are between 0 and 1, yielding real values $g_{i,j}$ so $g_{i,j}$ denotes the value for input j in combination i , $0 \leq g_{i,j} \leq 1$. However, we need to adapt this standard design for the integer constraints (3) and any non-box input constraints (4). To account for the integer constraints (3), we transform $g_{i,j}$ into integer values:

$$d_{i,j} = l_j + \lfloor (u_j - l_j + 1)g_{i,j} \rfloor \quad (10)$$

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 box-constrained search space that also satisfies these non-box constraints. Because the latter constraints may be rather complicated, we use a simple Monte Carlo estimate $\hat{f}$; i.e., we take a maximin LHS design of size N with $N \gg n_0$, evenly spread over this box-constrained space, apply the transformation in (10), and compute

$$\hat{f} = \frac{\sum_{i=1}^N I[\forall h'' : f_{h''}(\mathbf{d}) \leq c_{h''}]}{N} \quad (11)$$

with indicator function $I[\cdot] = 1$ if the input combination $\mathbf{d}$ also satisfies all non-box input constraints in (4); else $I[\cdot] = 0$. Next we select a maximin LHS design with sample size $\hat{n}_0 = \lceil n_0 / \hat{f} \rceil$ where $\lceil \cdot \rceil$ 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 $\hat{n}_0$) – until an acceptable initial number of feasible input combinations results.

Step 2: Simulate initial design points

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

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

$$\forall h : \frac{l_h(m_i, \alpha)}{\overline{w_h(\mathbf{d}_i)}} \leq \frac{\gamma}{1 + \gamma} \text{ with } h = 0, \dots, r - 1, \quad (12)$$

then m_i realizes the relative precision γ with $0 < \gamma < 1$. This halfwidth is

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

where $t_{m_i-1; 1-\alpha/2}$ denotes the $(1 - \alpha/2)$ quantile of the t_{m_i-1} distribution, and $\widehat{\text{var}}(w_h(\mathbf{d}_i))$ denotes the estimated variance

$$\widehat{\text{var}}(w_h(\mathbf{d}_i)) = \frac{\sum_{l=1}^{m_i} (w_{h;l}(\mathbf{d}_i) - \overline{w_h(\mathbf{d}_i)})^2}{m_i - 1} \text{ with } m_i > 1 \quad (13)$$

with $\overline{w_h(\mathbf{d}_i)}$ denoting the average simulation output

$$\overline{w_h(\mathbf{d}_i)} = \frac{\sum_{l=1}^{m_i} w_{h;l}(\mathbf{d}_i)}{m_i} \quad (14)$$

where $w_{h;l}(\mathbf{d}_i)$ denotes simulation output h of replication l of input combination $\mathbf{d}_i$.

Because we use the I/O simulation 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 *CRN* (see Law, 2007) because we expect CRN to create positive correlations between outputs – so even outputs at input combinations that are relatively far away from the input combination to be predicted, still contain information (the Kriging parameters θ_j in (9) become smaller).

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

We use DACE to fit r univariate Kriging models to the available simulation I/O data. We fit these 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* (significantly higher goal output). Our reason is that information regarding the location of infeasible or suboptimal regions is important for steering the INLP search in steps 6 and 8.

Step 4: Validate the Kriging metamodels

We apply *cross-validation*, similar to the procedure in Kleijnen (2008) (who gives more references). First we outline this procedure in the following six stages (next, we shall detail each stage):

- (1) From the set of simulation I/O data, delete one input combination at a time – but avoid extrapolation (as explained below).
- (2) Based on the remaining I/O data, compute the Kriging predictors for every output h ($h = 0, \dots, r - 1$) at the left-out combination.
- (3) Estimate the predictor variance for every output h at the left-out combination $\mathbf{d}_i$.
- (4) Compute the Studentized prediction error (say) $t_{m_i-1}^{h,i}$ for every output h 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 $t_{m_i-1}^{h,i}$ over all out-

puts h and input combinations i , and determine if this value is statistically significant. If this is the case, then all r Kriging models are rejected (simulation gives estimates of all r outputs; the heuristic then moves to step 5). Else, the metamodels are proclaimed to be valid (in which case they are ready for use in the INLP procedure, step 6).

Sub (1): To avoid extrapolation, we restrict the set of input combinations that can be left out in the cross-validation to those combinations $\mathbf{d}_i$ for which none of the input factors $(d_1, \dots, d_k)$ 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 (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 ($i = 1, \dots, n_{cv}$) from the set of simulation I/O data yields a corresponding remaining I/O data set $(\mathbf{d}_{-i}, \mathbf{w}_{-i})$. We call this data set the *cross-validation set* (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 (8), using the current MLE estimates for the correlation function in (9). This yields a predictor $y_h^-(\mathbf{d}_i)$ for every output h . We do not re-estimate the correlation functions in (9), 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 (9), 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 $(w_{0,l}(\mathbf{d}_i), w_{1,l}(\mathbf{d}_i), \dots, w_{r-1,l}(\mathbf{d}_i))^T$. 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 (see Step 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: $m = \max\{m_z | z \neq i\}$ ($i = 1, \dots, n_{cv}$).

- (2) Bootstrap the replicate numbers from 1 to m ; i.e., sample m replicate numbers (say) l from the uniform distribution with constant probability $1/m$ for the m integers $[1, 2, \dots, m]$. This yields a set of m replicate numbers $\{l_1, l_2, \dots, l_m\}$.
- (3) For each input combination $\mathbf{d}_z$ in the cross-validation set, include the output vectors $(w_{0,l_1}(\mathbf{d}_z), w_{1,l_1}(\mathbf{d}_z), \dots, w_{r-1,l_1}(\mathbf{d}_z))$ to $(w_{0,l_m}(\mathbf{d}_z), w_{1,l_m}(\mathbf{d}_z), \dots, w_{r-1,l_m}(\mathbf{d}_z))$ 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 $\overline{w_h^*(\mathbf{d}_z)}$ ($h = 0, \dots, r-1$) averaged over the bootstrapped replicates, for all combinations $\mathbf{d}_z$ in the cross-validation set:

$$\overline{w_h^*(\mathbf{d}_z)} = \frac{\sum_{j=1}^{m_z^*} w_{h,l_j}(\mathbf{d}_z)}{m_z^*} (h = 0, \dots, r-1). \quad (15)$$

- (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 (15).

To decrease the sampling effects of bootstrapping, we repeat this whole procedure until we have B successful bootstrap samples (in our experiments, we select $B = 200$). This yields estimated optimal Kriging weights $\lambda_{h,b}^*$ and bootstrapped Kriging predictors $y_{h,b}^*(\mathbf{d}_i)$ with $b = 1, \dots, B$. These B values enable estimation of the variance of the Kriging predictor, analogously to (13):

$$\widehat{\text{var}}(y_h^*(\mathbf{d}_i)) = \frac{\sum_{b=1}^B (y_{h,b}^*(\mathbf{d}_i) - \overline{y_h^*(\mathbf{d}_i)})^2}{B-1} \quad (16)$$

where

$$\overline{y_h^*(\mathbf{d}_i)} = \frac{\sum_{b=1}^B y_{h,b}^*(\mathbf{d}_i)}{B}. \quad (17)$$

Sub (4): Compute the *Studentized prediction error* for every output h of the left-out combination $\mathbf{d}_i$:

$$t_{m_i-1}^{h,i} = \frac{\overline{w_h(\mathbf{d}_i)} - y_h^-(\mathbf{d}_i)}{\sqrt{\widehat{\text{var}}(w_h(\mathbf{d}_i)) + \widehat{\text{var}}(y_h^*(\mathbf{d}_i))}} (h = 0, \dots, r-1) (i = 1, \dots, n_{cv}) \quad (18)$$

where $\widehat{\text{var}}(w_h(\mathbf{d}_i)) = \widehat{\text{var}}(w_h(\mathbf{d}_i))/m_i$ with $\widehat{\text{var}}(w_h(\mathbf{d}_i))$ following from (13).

Note: The numerator of (18) uses $y_h^-(\mathbf{d}_i)$, not $\overline{y_h^*(\mathbf{d}_i)}$; see (17). 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 (18) we would use $\widehat{var(y_h^*(\mathbf{d}_i))} = \widehat{var(y_h^*(\mathbf{d}_i))}/B$. 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 $\widehat{var(y_h^*(\mathbf{d}_i))} \rightarrow \sigma^2 > 0$ as $B \uparrow \infty$, where σ^2 denotes the unknown predictor variance (see the numerator and denominator in (16)). The denominator has the term $\widehat{var(y_h^*(\mathbf{d}_i))}$, which 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 $t_{m_i-1}^{h,i}$ per output type h .

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

$$\max_h [\max_i |t_{m_i-1}^{h,i}|] > t_{m_{min}-1; 1-\lceil \alpha/(2n_{cv}r) \rceil} \quad (h = 0, \dots, r-1) \quad (i = 1, \dots, n_{cv}) \quad (19)$$

where the right-hand side uses degrees of freedom $m_{min} = \min\{m_i | i = 1, \dots, n_{cv}\}$ 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*.

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 (19)), 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; see again (9). We therefore select the point halfway the worst point and its nearest neighbor in the current design – measuring distances between points 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 proclaimed valid.

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 $(r - 1)$ constraint functions, we apply an INLP solver to the Kriging metamodels, in order to estimate the optimum input combination. INLP problems are notoriously difficult; many INLP solvers are available; see (Grossmann, 2002; Pintér, 2007) and

<http://www.gamsworld.org/minlp/solvers.htm>.

In our experiments, we use the *bnb20.m* solver, 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 solver 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 solver has already been simulated (and, hence, is part of the simulated design), the heuristic goes to step 8. Otherwise, it proceeds to step 9.

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 solver 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 $\mathbf{d}_t$ ($t = 1, \dots, T$) denote the (say) T points that have already been simulated during the previous iterations. To select a new point excluding these $\mathbf{d}_t$, we impose the following T additional constraints

$$1 \leq \sum_{j=1}^k (d_{t;j} - d_j)^2 \quad (t = 1, \dots, T). \quad (20)$$

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.

Step 9: Add new point to design, and simulate

Now we add the latest estimated optimal input combination – determined in either step 6 or step 8 – to the design, and simulate this point. (As Figure 1 shows, we re-estimate the correlation parameters and the corresponding optimal Kriging weights in Step 3, and we validate the resulting Kriging meta-model in Step 4).

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

This step is 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 $(r - 1)$ output constraints, and (ii) significant improvement: if it is a feasible point, its corresponding objective value needs to be significantly better than the best point found so far.

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:

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

with $\nu = \min(m_{curr}, m_{best})$ where m still denotes the number of replicates. Only if $t_v < -t_{v;1-\alpha/2}$, 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 fails to give 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.

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). In the online appendix, we illustrate 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.

First we discuss a variant of the popular (s, S) inventory problem specified by (Bashyam and Fu, 1998) (this variant was also used by Kleijnen and Wan, 2007). Next we discuss a call center staffing problem specified by (Kelton et al., 2007). For these test problems we compare the computational results of our heuristic with those of OptQuest. We wrote our own heuristic in Matlab’s version 7.4. For commercial reasons, OptQuest is a black box heuristic. Unfortunately, different versions of Arena combined with OptQuest give different results; we report results for two recent versions, namely 11 and 12.

For each test problem, we use ten macroreplicates for OptQuest and our heuristic DOE-Kri-MP. We either select the number of replicates per point as a constant or we determine m_i through the relative precision requirement with $\gamma = 0.15$; see again (12). We present the reported optimum input $\mathbf{d}_{\text{opt}}$, the estimated goal output, and the estimated constrained output. We also present the total number of design points simulated by the heuristic, i_{max} , and the sequence number or ”rank” of the reported optimum in the list of points simulated. Though OptQuest allows the user to select a ”relative precision requirement”, we apply it only with a fixed number of replicates per design point because OptQuest applies this criterion only to the goal output – not the constrained outputs. Though the OptQuest user documentation claims that selecting the Automatic Stop condition will stop the search after 100 non-improving points, in our experiments this option does not work properly; i.e., OptQuest continued its search after reaching this threshold. Consequently, we assign OptQuest a fixed number of search points; we select this number equal to the number of points that our heuristic requires to meet its stopping criterion. For each macroreplicate, we report OptQuest’s estimated optimum after it has simulated the same number of points as our heuristic required. The results indicate that our heuristic requires fewer points to reach the optimum than OptQuest – as we shall see next.

An (s, S) inventory system

Like (Bashyam and Fu, 1998) we consider an infinite-horizon periodic-review (s, S) inventory simulation model with full backlogging. 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. If the inventory position has dropped to a value smaller than or equal to the reorder level s , then 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, namely orders arrive 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 $E(TC)$, which denotes the expected Total Costs – consisting of average ordering and holding cost – per period, subject to the constraint that the stockout rate δ – fraction of demand not supplied from stock on hand – is at most 10%. Order crossing makes this problem analytically intractable so simulation is applied. The simulation uses 30,000 periods per replicate. The search area is limited to $900 \leq s \leq 1250$ and $1 \leq Q \leq 500$.

We either fix the number of replicates per point at 10 like (Bashyam and Fu, 1998) or we determine this number through the relative precision requirement with $\gamma = 0.15$; the latter requirement requires only 4 to 6 replicates per point because of the long replication length (namely, 30,000 periods).

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) with $m_i = m = 10$, for the same 10 macroreplicates; OptQuest starts from the initial solution (1075, 250). OptQuest embedded in Arena 11 does not return the estimated value of the constraint outputs, so the corresponding entries in the table are set to "NA" (not available). Table 2 summarizes the performance results of the different heuristics, in terms of the average, minimum, and maximum of the estimated goal and constraint values over the ten macroreplicates. Table 1 shows that the quality of the optimal solution found by our heuristic is consistently better than the solutions provided by the OptQuest version embedded in Arena 11. The Arena 12 version succeeds in finding a better optimum (lower objective value) in only 3 out of 10 macroreplicates, namely the 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 10 macroreplicates than our heuristic's solution. Table 1 shows that our heuristic requires fewer points to move towards the constraint boundary (where $\delta = 0.10$)

As mentioned above, in this example the use of a relative precision criterion $\gamma = 0.15$ saves simulation time because it requires fewer replicates per point. This does not seem to impact the performance of our heuristic substantially. Contrary to our expectations, it – in general – causes our heuristic to simulate fewer points before satisfying the stopping criterion. Table 2 suggests that the quality of the optimum tends to be higher with the relative precision (i.e., lower values for the objective are obtained), but these results should be interpreted with caution because the estimates of the objective and the constraint are less reliable.

A call-center staffing problem

We use the call-center simulation model specified by (Kelton et al. 2007). (Other simulation models for call center staffing – applying different opti-

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

Macrorep.	Heuristic	$\mathbf{d}_{\text{opt}}$	TC	δ	i_{max}	Rank
1	OptQuest (Arena 12) $m_i = 10$	(1009, 287)	716.16	8.28	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	9.92	60	14
	DOE-Kri-MP $\gamma = 0.15$	(1021, 114)	640.09	9.91	54	41
2	OptQuest (Arena 12) $m_i = 10$	(1027, 84)	632.42	9.93	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	9.92	68	14
	DOE-Kri-MP $\gamma = 0.15$	(1061, 31)	634.74	9.99	61	27
3	OptQuest (Arena 12) $m_i = 10$	(1050, 44)	632.77	9.93	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	9.92	81	14
	DOE-Kri-MP $\gamma = 0.15$	(1062, 29)	634.19	9.99	58	40
4	OptQuest (Arena 12) $m_i = 10$	(1061, 191)	713.74	7.46	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	9.99	69	48
	DOE-Kri-MP $\gamma = 0.15$	(1076, 12)	637.39	9.93	61	40
5	OptQuest (Arena 12) $m_i = 10$	(1129, 35)	700.19	7.11	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	9.92	81	14
	DOE-Kri-MP $\gamma = 0.15$	(1041, 73)	638.04	9.83	66	12
6	OptQuest (Arena 12) $m_i = 10$	(1002, 209)	671.28	9.36	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	9.92	66	14
	DOE-Kri-MP $\gamma = 0.15$	(1047, 58)	635.20	9.98	62	22
7	OptQuest (Arena 12) $m_i = 10$	(1027, 84)	632.02	9.98	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	9.92	73	14
	DOE-Kri-MP $\gamma = 0.15$	(1041, 73)	638.04	9.83	54	12
8	OptQuest (Arena 12) $m_i = 10$	(1054, 59)	642.44	9.56	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	9.96	75	55
	DOE-Kri-MP $\gamma = 0.15$	(1057, 40)	634.62	9.90	55	27
9	OptQuest (Arena 12) $m_i = 10$	(1061, 69)	714.79	7.39	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	9.92	69	14
	DOE-Kri-MP $\gamma = 0.15$	(1039, 73)	636.24	9.92	71	38
10	OptQuest (Arena 12) $m_i = 10$	(999, 185)	655.21	9.72	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	9.92	63	14
	DOE-Kri-MP $\gamma = 0.15$	(1049, 54)	636.14	9.99	67	40

Table 2

Performance summary based on 10 macroreplicates for the (s, S) inventory model

Heuristic	Total Costs TC			Stockout rate δ		
	average	max	min	average	max	min
OptQuest (Arena 12) $m_i = 10$	671.11	716.17	632.02	8.87	9.98	7.11
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	9.93	9.99	9.92
DOE-Kri-MP $\gamma = 0.15$	636.47	640.10	634.19	9.93	9.99	9.83

mization heuristics – are presented by (Avramidis et al. 2007) and (Atlason et al. 2008), who also review related publications.) Our telephone call-center handles three types of calls (Kelton et al. 2007, p. 195): (i) technical support, (ii) sales, and (iii) order-status checking. Customers call a central telephone number, which feeds a number of trunk lines. If customers call in when all trunk lines are busy, they get a busy signal and cannot enter the 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). Additional staff may be hired (Sales, Tech 1, Tech2, Tech3 or Tech All). Moreover, additional trunk lines may be obtained.

The decision variables in (1) now 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, $d_2 = \text{Tech1}$, $d_3 = \text{Tech2}$, $d_4 = \text{Tech3}$, $d_5 = \text{TechAll}$, and $d_6 = \text{Sales}$. The goal function to be minimized is denoted by TC , Total Cost per week (salaries of all staff, cost for hiring all trunk lines, and a 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 , which should be below 5%. The number of trunk lines d_1 must satisfy the box constraint $26 \leq d_1 \leq 50$; moreover, the personnel inputs must satisfy the linear constraint $d_2 + d_3 + d_4 + d_5 + d_6 \leq 15$. All six input variables ($k = 6$) should be integers. Kelton et al., 2007 use this simulation model to illustrate the use of OptQuest.

Table 3 shows the results of our heuristic for 10 macroreplicates. The number of replicates per point is either fixed at $m_i = m = 110$ or determined by the relative precision requirement with $\gamma = 0.15$. In this example, $m_i = m = 110$ gives a lower relative precision than $\gamma = 0.15$ in general, in particular for the Percent Rejected Calls δ . OptQuest uses $m_i = m = 110$, starting from the initial solution (29, 3, 3, 3, 3, 3) that was suggested by (Kelton et al., 2007). Table 4 summarizes the results over the ten macroreplicates for the various 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

Table 3
Results for the call center model

Macrorep.	Heuristic	$\mathbf{d}_{opt}$	Total Costs TC	Percent Rejected Calls δ	i_{max}	Rank
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 based on 10 macroreplicates for the call center model

Heuristic	Total Costs TC			Percent Rejected Calls δ		
	average	max	min	average	max	min
OptQuest (Arena 12) $m_i = 110$	21975.11	22282.48	21452.64	3.99	4.98	3.14
OptQuest (Arena 11) $m_i = 110$	21621.07	21854.64	21398.62	NA	NA	NA
DOE-Kri-MP $m_i = 110$	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

faster towards the constraint boundary than OptQuest does, and obtains high-quality solutions with fewer design points.

Though the results seem to indicate that the heuristic with $\gamma = 0.15$ does not obtain the same performance as with a fixed number of replicates (in some cases, it settles at a worse objective function value), these results should again be interpreted with caution because the estimates for the macroreplicates with $\gamma = 0.15$ are more reliable (higher precision). We recommend the use of the relative precision criterion γ because it controls the magnitude of the "noise" relative to the "signal". It gives replication numbers that vary with the input combination (we assume that the noise does not remain constant as we move through 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 the objective to be minimized, while the other $(r - 1)$ outputs must satisfy prespecified threshold values. The deterministic simulation inputs must meet prespecified constraints, including integer constraints. The heuristic combines (i) DOE, (ii) Kriging metamodeling to approximate the global I/O functions per output type implied by the underlying simulation model, and (iii) INLP to estimate the optimal solution based on the Kriging models. DOE includes sequential design, and allows for possibly non-linear input constraints. Kriging includes validation of the Kriging metamodel through bootstrapping allowing for CRN and unequal replication numbers per design point. MP is forced to give a point that is not yet part of the design. The resulting stepwise procedure contains elements of both global and local search.

We applied our heuristic to various simulation models, and compared its performance with OptQuest. The results suggest that our heuristic requires fewer simulated points to find an optimum solution than OptQuest does. We do not compare computational times required by the alternative heuristics, because (i) these times strongly depend on the coding of the heuristics; e.g., it is well-known that cross-validation in linear regression may avoid repeated computa-

tion by using the so-called "hat" matrix, and similar tricks may be possible in Kriging; (ii) we assume computationally expensive simulation models so the computer time needed by the metamodeling may be negligible.

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; namely, random objective, random output constraints, and integer input constraints. Our heuristic is flexible (robust). For example, our heuristic 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. If analysts prefer other types of global metamodels, then they can replace our Kriging metamodel by their favorite metamodel and adapt our heuristic. Practitioners may appreciate the fact that our heuristic is composed of modules that use off-the-shelf software.

We propose the following *future research* agenda:

1. Replace Ordinary Kriging by Kriging that accounts for *random* simulation models with *multivariate* output with variance heterogeneity, possibly using CRN. Only recently, (Ankenman et al., 2008) and (Yin et al. 2008)) developed a Kriging model for random simulation; (Chilès and Delfiner, 1999), (Santner et al., 2003, pp. 101-116), and (Wackernagel, 2003) discuss multivariate Kriging—but these publications do not use an optimization context.
2. Try to prove *convergence* of our heuristic to the true optimum, as the number of replicates per point tends to infinity and the number of points 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 points is by definition finite in our setting because of the combination of integer and box constraints.
3. Collect additional *empirical evidence* of the performance of our heuristic compared with alternative heuristics (e.g., OptQuest) for constrained random simulation optimization in various cases, including high k values (high-dimensional search space). The simulation community should develop standard testbeds for different heuristics for simulation optimization. Such testbeds are popular in Mathematical Programming, but they do not yet exist in the simulation field. A start is made with the website

<http://www.simopt.org>

which is based on (Pasupathy and Henderson, 2006), but its test problems are not available as computer code so they require recoding, which is a major source of errors. When comparing heuristics, we point out that heuristics using global metamodels require the simulation of points in the whole area, with adequate precision.

4. Extend the problem formulation from non-negative integer inputs to *mixed integer and continuous inputs*. This generalization implies that the design is not restricted to integer combinations, and that the INLP solver is replaced by a Mixed INLP (denoted as MINLP) solver. The latter might take advantage of the gradients that are estimated by the Kriging model for the continuous inputs (Biles et al., 2007). Different INLP or MINLP solvers may be tried.
5. Replace Matlab’s maximin LHS by a *non-random space-filling* design; many references and websites are given by Kleijnen (2008, pp. 126-130).
6. Experiment with different *sizes* for the initial (pilot) design in Step 1.
7. Accept points 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.
8. Add a *postprocessing* step (to be run after our heuristic stops), aimed at ordering high-quality solutions (Kelly, 2002). Such a step may use Multiple Ranking and Selection procedures (Kleijnen, 2008, p. 102); selecting such a procedure is discussed by (Branke et al., 2007).

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.
- 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. Nelson, and J. Staum (2008), Stochastic kriging for simulation metamodeling, *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. R. Hill, L. Moench, O. Rose, pp. 362-370
- 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.
- Barton, R.R. and M. Meckesheimer (2006), Metamodel-based simulation optimization. *Handbooks in Operations Research and Management Science, Volume 13*, edited by S.G. Henderson and B.L. Nelson, Elsevier/North Holland, pp. 535-574
- 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** 161–177.
- 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.
- Cressie, N.A.C. (1993), *Statistics for spatial data: revised edition*. Wiley, New York
- 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)
- 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.
- Egea, J.A., E. Vazquez, J.R. Banga, R. Martí 2009, Improved scatter search for the global optimization of computationally expensive dynamic models. *Journal of Global Optimization*, **43** 175-190.
- Fu, M.C. 2002. Optimization for Simulation: Theory vs. Practice, *Inform. J. Comput.* **14** 192–215.
- Fu, M.C. 2007. Are we there yet? The marriage between simulation & optimization. *OR/MS Today* **34** 16–17.
- Greenberg, H.J., T. Morisson. 2008. Robust optimization. A.R. Ravindran, ed. *Handbook of Operations Research and Management Science*. CRC Press, Boca Raton, Florida.
- Grossmann, I.E. 2002. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering* **3** 227-252
- 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, *Inform. 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. 2008. *Design and analysis of simulation experiments*. Springer Science + Business Media.

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 recommenda-

tions. *Comput. Ind. Eng.* **54**(2) 327–339.

Sacks, J., W.J. Welch, T.J. Mitchell, and H.P. Wynn (1989), Design and analysis of computer experiments (includes Comments and Rejoinder). *Statistical Science*, 4, no. 4, pp. 409–435

Santner, T.J., B.J. Williams, and W.I. Notz (2003), *The design and analysis of computer experiments*. Springer-Verlag, New York

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.

Wackernagel, H. (2003), *Multivariate geostatistics: an introduction with applications; third edition*. Springer-Verlag, Berlin

Yin, J., S.H. Ng, and K.M. Ng (2008), Kriging model with modified nugget effect. *Proceedings of the 2008 IEEE International Conference on Industrial Engineering and Engineering Management*, to appear.

Acknowledgment

We thank Bert Bettonvil (Tilburg University) for bringing (Avramidis et al., 2007) to our attention, William (‘Bill’) Biles (University of Louisville, Kentucky) for his contribution to an earlier version, two anonymous referees, Jose Egea (Instituto de Investigaciones Marinas, Spain), and Dick den Hertog (Tilburg University) for their detailed comments on earlier versions of this article, Gabriella Dellino and Carlo Meloni (Politecnico di Bari) for the many helpful discussions, and Hans Blanc (Tilburg University) and David Kelton (University of Cincinnati) for their help with OptQuest.