Comparison of Kriging-Based Methods for Simulation Optimization with Heterogeneous Noise

Hamed Jalali, Inneke van Nieuwenhuyse, Victor Picheny

Comparison of Kriging-based methods for simulation optimization with heterogeneous noise

Hamed Jalali*[1], Inneke Van Nieuwenhuyse†[1] and Victor Picheny‡[2]

[1]Research Center for Operations Management, Department of Decision Sciences and Information Management, KU Leuven, Naamsestraat 69, 3000 Leuven, Belgium

[2]INRA-French National Institute for Agricultural Research, 31326 Castanet-Tolosan, France

Abstract

In recent years, several algorithms have been proposed which extend the traditional Kriging-based simulation optimization methods (assuming deterministic outputs) to problems with noise. Our objective in this paper is to compare the relative performance of a number of these algorithms on a set of well-known analytical test functions, assuming different patterns of heterogeneous noise. We also apply the algorithms to a popular inventory test problem. The conclusions and insights obtained may serve as a useful guideline for researchers aiming to apply Kriging-based methods to solve engineering and/or business problems, and may be useful in the development of future Kriging-based algorithms.

Keywords: Simulation, Stochastic Kriging, Heterogeneous noise, Ranking and Selection, Optimization via simulation

1 Introduction

Assume that we would like to find the solution that minimizes some goal function $f(\mathbf{x})$:

$$\min_{\mathbf{x}\in\Theta} f(\mathbf{x}), \tag{1}$$

where $f : \Theta \to \mathbb{R}$ and $\mathbf{x} = (x_1, x_2, ..., x_d)$ (with d the dimension of the solution space). This goal function cannot be directly observed, and must be estimated through a stochastic simulation model: we thus only have access to noisy observations $\tilde{f}_j(\mathbf{x}^i) = f(\mathbf{x}^i) + \varepsilon_j(\mathbf{x}^i)$, where $\tilde{f}_j(\mathbf{x}^i)$ represents the observed goal value in the jth simulation replication at point $\mathbf{x}^i$. The noise $\varepsilon_j(\mathbf{x}^i)$ has mean zero, and its variance depends on $\mathbf{x}^i$ (we thus have heterogeneous noise). We usually estimate the value of $f(\mathbf{x}^i)$ by performing n^i simulation replications: $\bar{f}(\mathbf{x}^i) = \sum_{j=1}^{n^i} \tilde{f}_j(\mathbf{x}^i)/n^i$.

This type of problem is very common in engineering and business applications since in many practical problems, the objective function is analytically intractable (Law, 2015). In inventory management problems, for instance, we often need to optimize a discrete event simulation model to obtain the optimal stocking quantities (Jalali and Van Nieuwenhuyse, 2015). Examples in other fields include supply chain design (Saif and Elhedhli, 2015), pump scheduling (Naoum-Sawaya et al., 2015), and ambulance fleet allocation (McCormack and Coates, 2015).

*Corresponding author: Telephone: +3216379064, Fax: +3216326624, E-mail: hamed.jalali@kuleuven.be

†inneke.vannieuwenhuyse@kuleuven.be

‡Victor.Picheny@toulouse.inra.fr

In this paper, we focus on problems where Θ is continuous; for reasons explained in Section 3.1, we discretize the function domain to obtain a finite set of points (as in Frazier et al. (2009)). Many methods are available for solving problem (1) with finite set of points (see Hong et al. (2015) for a review of Ranking and Selection methods), but most of them are unsuitable when (1) the number of feasible solutions is large, or (2) simulation replications are time-consuming (Xu, 2012). Kriging-based or Bayesian optimization is among the few techniques that can handle this problem with low problem dimensionality ($d \leq 20$) (Brochu et al., 2010; Preuss et al., 2012). Based on the observed $\bar{f}(\mathbf{x}^i)$ for several $\mathbf{x}^i$, Kriging provides an approximation (or *metamodel*) for $f(\mathbf{x})$. The traditional *Efficient global optimization* (EGO) approach of Jones et al. (1998) is one of the most popular Kriging-based techniques for optimizing *noiseless* simulation; in this case, the fitted metamodel is a deterministic Kriging model. In stochastic simulation (e.g., discrete event simulation), however, EGO might not be very appropriate since it ignores the noise in the observations, assuming they were sampled with infinite precision (Quan et al., 2013). Realizing this deficiency, researchers have tried to extend EGO for stochastic simulation. Most of the approaches assume *homogeneous* simulation noise, meaning that the variance of the noise does not depend on $\mathbf{x}$ (Picheny et al., 2013a); Picheny et al. (2013b) compare several Kriging-based techniques for optimizing functions with this type of noise.

In practice, however, the noise is *heterogeneous* (Kleijnen and Van Beers, 2005; Yin et al., 2011; Kim and Nelson, 2006). In recent years, a number of Kriging-based optimization algorithms have been proposed that can handle heterogeneous noise, based on stochastic Kriging models (see Cressie (1993), Ankenman et al. (2010) and Yin et al. (2011)). To the best of our knowledge, the performance of these algorithms has not yet been compared. Our objective in this paper is to evaluate the effectiveness and the relative performance of these algorithms for simulation optimization problems with heterogeneous noise. More specifically, we evaluate the performance of six methods: correlated knowledge-gradient (CKG, Frazier et al. (2009)), two-stage stochastic optimization (TSSO, Quan et al. (2013)), extended two-stage stochastic optimization (eTSSO, Liu et al. (2014)), expected quantile improvement (EQI, Picheny et al. (2013a)), the minimum quantile criterion (MQ, Picheny et al. (2013b), which is similar to the Gaussian process upper confidence bound (GP-UCB) method of Auer et al. (2002) and Jones (2001)), and an adapted version of the sequential Kriging optimization approach (SKO, Huang et al. (2006)). To this end, we apply these algorithms to minimize three well-known analytical test functions (Rescaled Branin, Six-hump camel-back, and Hartmann-6), perturbed with heterogeneous Gaussian noise, over a finite set of solutions. We also apply the algorithms to optimize the (s, S) inventory system of Fu and Healy (1997) which has been a popular test problem in the simulation optimization literature (Jalali and Van Nieuwenhuyse, 2015).

Though our work is closely related to Picheny et al. (2013b), it clearly differs in the following respects: (1) we consider heterogeneous noise, (2) we add two recent Kriging-based algorithms (TSSO and eTSSO), (3) we adapt the SKO algorithm (Huang et al., 2006) to settings with heterogeneous noise, (4) we explicitly consider the *replication strategy* used by the different algorithms. In Picheny et al. (2013b), a single replication is used for estimating $f(\mathbf{x})$ (they use homogeneous noise with known magnitude). In our case, algorithms need to smartly allocate the limited replication budget to obtain reasonably good estimates (which is referred to as the

replication strategy). While Picheny et al. (2013b) compare the algorithms only in terms of how they perform the search for interesting alternatives in Θ, we also study the effectiveness and importance of the replication strategies and their influence on the choice of the final solution (i.e., identification step).

The complexity and computational requirements vary widely across the algorithms. As the MQ method is the most straightforward, we consider it as a benchmark in our experiments, checking whether it is outperformed by the more sophisticated (and, thus, more computationally expensive) approaches. Additionally, some algorithms (i.e., CKG, SKO and EQI, see Section 2) require information on the noise variance function $\tau^2(\mathbf{x})$ at any $\mathbf{x} \in \Theta$. This may limit the applicability of these methods in practice, as an unknown noise variance function is "a fact of life in system simulation problems"(Kim and Nelson, 2006) so it needs to be estimated. In our experiments, we test whether these "informed" algorithms outperform the ones that don't need this information.

Our conclusions may serve as a useful guideline for researchers who want to apply Kriging-based methods to solve engineering and/or business problems, and may be useful in the development of future Kriging-based algorithms. Our main conclusions are as follows:

1. not only the magnitude but also the structure of noise can have a large impact on the performance of the algorithms;

2. among the methods that require an estimate of $\tau^2(\mathbf{x})$, SKO and CKG usually outperform MQ while EQI does not;

3. two-stage stochastic optimization (TSSO) simulates the most promising alternatives but usually fails to identify them at the end. If the identification is improved, TSSO will outperform MQ and will provide very competitive results, making it preferable over SKO or CKG since it does not require an estimate of the noise variance function. Also, eTSSO does not seem to provide better results than TSSO;

4. none of the existing methods have an appropriate replication strategy. While the focus of the literature has been primarily on designing better search methods, our results show that there is a need for smarter replication strategies (e.g., by adding an appropriate ranking and selection procedure). This can likely lead to significant improvements in the performance of all algorithms.

The remainder of this article is organized as follows. Section 2 provides a brief explanation of the Kriging-based methods studied, Section 3 details the experiments, and Section 4 contains the main results (additional results are provided in appendix). We present conclusions in Section 5.

2 Overview of Kriging-based methods

Section 2.1 provides a brief introduction to Kriging-based simulation optimization, and explains the general steps involved in this process. Section 2.2 highlights the similarities and differences between the methods studied in this paper.

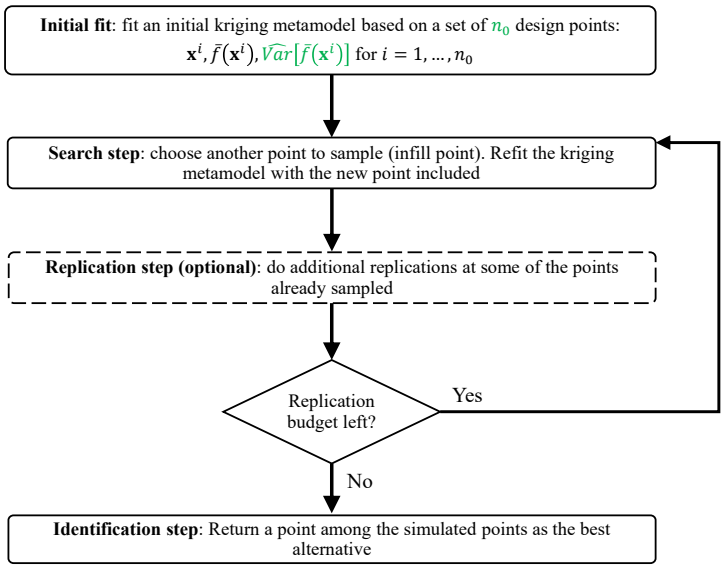


Figure 1: Typical steps in Kriging-based optimization

2.1 Kriging-based simulation optimization

As noted in Section 1, we need a stochastic simulation model to estimate the objective value at a point within Θ (see Equation 1). As the simulation output is noisy, we use the sample mean of the simulation output across several replications to obtain an estimate. This estimate is denoted by $\bar{f}(\mathbf{x}^i)$ (with estimated variance $\widehat{\text{Var}}[\bar{f}(\mathbf{x}^i)]$) at any arbitrary point $\mathbf{x}^i$.

As shown in Figure 1, the Kriging-based methods studied in this article are *sequential*. In the first step, they fit an initial Kriging metamodel using n_0 initial points. Space filling designs (e.g., Latin hypercube sampling) are often used for sampling these initial points (Kleijnen, 2009); simulation replications are used to estimate the objective value at these points. Since our observations are noisy, we use stochastic Kriging (Ankenman et al., 2010) to fit an initial Kriging metamodel. This metamodel provides us with (1) $\hat{f}(\mathbf{x})$: a prediction for the objective function value at any $\mathbf{x} \in \boldsymbol{\Theta}$ and (2) $\sigma^2(\mathbf{x})$: the variance of $\hat{f}(\mathbf{x})$, which provides a measure for the Kriging prediction uncertainty or error (Kleijnen, 2014); the estimate of this Kriging variance is denoted by $s^2(\mathbf{x})$, see Appendix A for related expressions. Table 1 provides an overview of the most important notations in this paper. We refer to Quan et al. (2013), Ankenman et al. (2010), Yin et al. (2011), and Cressie (1993) for further details on stochastic Kriging.

In the *search step*, the algorithm iteratively chooses a point, simulates it for some number of replications to obtain $\bar{f}(\mathbf{x})$ and $\widehat{\text{Var}}[\bar{f}(\mathbf{x})]$, and refits the Kriging metamodel with the new point included. The algorithm uses an *infill criterion* to choose these points (known as *infill points*): the purpose of the infill criterion is to discover the good alternatives in Θ, and guide the search to regions with promising objective values (Forrester et al., 2008).

Optionally, the *replication step* allows us to perform extra replications at some of the points already sampled (either initial design points or infill points), to increase the accuracy of $\bar{f}(\mathbf{x})$ in those points. Such a replication step is present in TSSO, eTSSO, and EQI (see section 2.2), but the actual choice of points and the number of replications allocated to this step differ between

Table 1: Overview of notations

Notation	Description
$f(\mathbf{x}^i)$	The true objective value at point $\mathbf{x}^i$, which is assumed to be unknown.
$\bar{f}(\mathbf{x}^i)$	Estimate of the objective value at point $\mathbf{x}^i$: $\bar{f}(\mathbf{x}^i) = \sum_{j=1}^{n} \tilde{f}_j(\mathbf{x}^i)/n^i$.
$\hat{f}(\mathbf{x}^i)$	Kriging estimate: Kriging prediction of the objective value at point $\mathbf{x}^i$.
$s^2(\mathbf{x}^i)$	Estimate of Kriging prediction variance at point $\mathbf{x}^i$.

these algorithms.

Each time we execute the search and replication steps, we consume part of our finite replication budget. Hence, after some iterations, the replication budget will end and we enter the *identification step*: the algorithm looks back at all simulated or tested (T) alternatives (Θ^T) and returns one of them as the best solution to the optimization problem. In noiseless simulation, this is trivial since $f(\mathbf{x})$ is available for all the simulated points: $\mathbf{x}^r = \arg\min_{\mathbf{x}\in\Theta^T} f(\mathbf{x})$, where $\mathbf{x}^r$ denotes the proposed solution. In the presence of noise, only $\hat{f}(\mathbf{x})$ is available, along with $s^2(\mathbf{x})$ and the observed $\bar{f}(\mathbf{x})$ at each alternative $\mathbf{x} \in \Theta^T$. Consequently, the success of an algorithm is impacted by the infill criterion (only alternatives that have been simulated are considered in the identification step), by the criterion used to choose $\mathbf{x}^r$ in the identification step (i.e., the *identification criterion*), and by the replication strategy (more accurate estimates can help to distinguish superior alternatives in the identification step).

2.2 Description of the algorithms

Due to the limited budget, all algorithms must make a trade-off between the total number of infill points and their accuracy (Picheny et al., 2013a). If the replication strategy performs only a few replications at each point, we can sample many infill points but the estimates $\bar{f}(\mathbf{x})$ may remain very noisy. This may result in inaccurate Kriging models, which in turn affects the search step (i.e., we might sample points with bad objective values). Additionally, it may cause problems in the identification step, as it becomes more difficult to differentiate between good and inferior solutions. Allocating more replications to the points mitigates these issues, but decreases the number of infill points, implying that we might not get the chance to discover (some) interesting alternatives.

Table 2 summarizes the main characteristics of the methods (infill criterion, replication strategy, identification criterion). As evident from this table, the identification criterion varies between the algorithms: yet, most methods rely on $\hat{f}(\mathbf{x})$ and $s^2(\mathbf{x})$ for proposing the final solution. The remainder of this section briefly explains the search and the replication strategy for each method.

2.2.1 MQ algorithm

MQ is a straightforward method and acts as a benchmark for other algorithms: it simply chooses the point with minimum Kriging quantile ($q(\mathbf{x}) = \hat{f}(\mathbf{x}) + \Phi^{-1}(\beta)s(\mathbf{x})$ with $\beta \in (0, 0.5]$) as the infill point in each iteration k: $\mathbf{x}^k = \arg\min_{\mathbf{x}\in\Theta} q(\mathbf{x})$.

Table 2: Kriging-based algorithms

Method	Source	Infill criterion	Replication strategy	Identification criterion	Info on $\tau^2(\mathbf{x})$ required?
1) MQ	Picheny et al. (2013b)	Min quantile	Revisits	$\min_{\mathbf{x}\in\Theta^T} q(\mathbf{x})$	No
2) SKO	Huang et al. (2006)	AEI	Revisits	$\min_{\mathbf{x}\in\Theta^T} q(\mathbf{x})$	Yes
3) CKG	Frazier et al. (2009)	CKG	Revisits	$\min_{\mathbf{x}\in\Theta^T} \hat{f}(\mathbf{x})$	Yes
4) EQI	Picheny et al. (2013a)	EQI	Replication step & Revisits	$\min_{\mathbf{x}\in\Theta^T} q(\mathbf{x})$	Yes
5) TSSO	Quan et al. (2013)	MEI	Replication step	$\min_{\mathbf{x}\in\Theta^T} \bar{f}(\mathbf{x})$	No
6) eTSSO	Liu et al. (2014)	MEI	Replication step	$\min_{\mathbf{x}\in\Theta^T} \bar{f}(\mathbf{x})$	No

Note: $q(\mathbf{x})$ is the Kriging quantile, $q(\mathbf{x}) = \hat{f}(\mathbf{x}) + \Phi^{-1}(\beta)s(\mathbf{x})$ where $\Phi^{-1}(.)$ is the inverse CDF of standard normal distribution and $\beta \in (0, 0.5]$ in MQ and $\beta \in [0.5, 1]$ in SKO and EQI.

As evident from the above equation, MQ does not require information on $\tau^2(\mathbf{x})$. The algorithm allocates a fixed number of replications per iteration (B): the analyst needs to decide on B prior to running the algorithm. However, MQ allows for revisits: at any iteration k, we might sample a point that has been simulated previously, and add B additional replications. The sampled points can thus receive a different total number of replications depending on how often they are revisited.

2.2.2 Adapted sequential Kriging optimization (SKO)

The SKO method of Huang et al. (2006) chooses the alternative with maximum *augmented expected improvement* (AEI) as the next infill point:

$$AEI(\mathbf{x}) = E\left[\max\left(\hat{f}(\mathbf{x}^{**}) - \hat{f}(\mathbf{x}), 0\right)\right]\left(1 - \frac{\tau(\mathbf{x})}{\sqrt{s^2(\mathbf{x}) + \tau^2(\mathbf{x})}}\right), \tag{2}$$

where $\hat{f}(\mathbf{x}^{**})$ is the Kriging prediction at the current *effective best solution* $\mathbf{x}^{**}$: i.e., the point with minimum $q(\mathbf{x})$ among the simulated points ($\beta \in [0.5, 1]$). Thus, the first term of the above equation represents how much we expect the cost value at $\mathbf{x}$ to be lower than $\hat{f}(\mathbf{x}^{**})$, see Huang et al. (2006) for details. This method was originally designed for homogeneous noise: in the second term of Equation 2, we introduce $\tau^2(\mathbf{x})$ instead of τ^2 to reflect the presence of heterogeneous noise. SKO also uses a fixed number of replications per iteration (B) and allows for revisits. Note that the second term in equation (2) prevents the algorithm from getting trapped in a given point $\mathbf{x}$, as continued resampling causes $s^2(\mathbf{x})$ to gradually approach zero, which in turn pushes $AEI(\mathbf{x})$ towards zero (Huang et al., 2006).

2.2.3 Correlated knowledge-gradient (CKG)

The idea behind the correlated knowledge-gradient (CKG) infill criterion is that in noisy environments, the Kriging prediction $\hat{f}(\mathbf{x})$ may be closer to $f(\mathbf{x})$ than the sample mean $\bar{f}(\mathbf{x})$; therefore, points are selected based on their effect on the Kriging prediction (Picheny and Ginsbourger, 2014; Frazier et al., 2009). More specifically, at iteration $k+1$, the improvement that would

result from sampling alternative $\mathbf{x}^{k+1}$ is defined as:

$$I(\mathbf{x}^{k+1}) = \min_{\mathbf{x}\in\Theta^k\cup\mathbf{x}^{k+1}} \hat{f}_k(\mathbf{x}) - \min_{\mathbf{x}\in\Theta^k\cup\mathbf{x}^{k+1}} \hat{f}_{k+1}(\mathbf{x}), \tag{3}$$

where Θ^k is the set of simulated points at the end of iteration k, $\hat{f}_k(\mathbf{x})$ is the current Kriging prediction and $\hat{f}_{k+1}(\mathbf{x})$ is the Kriging prediction after refitting the Kriging metamodel with $\mathbf{x}^{k+1}$ included. We choose the point with highest $\text{CKG}(\mathbf{x}) = E[I(\mathbf{x})]$ as the next point. As with SKO, we need to know $\tau^2(\mathbf{x})$ (or its estimate) for calculating this expectation (see Frazier et al. (2009) for further details on this calculation). Analogous to MQ and SKO, the CKG algorithm allows for revisits, and uses a fixed number of replications B per iteration.

2.2.4 Expected quantile improvement (EQI)

In EQI, at iteration $k+1$, simulating an alternative $\mathbf{x}^{k+1}$ is beneficial if its quantile using the updated Kriging model is lower than the current minimum quantile:

$$I(\mathbf{x}^{k+1}) = \left(\min_{\mathbf{x}\in\Theta^k} q_k(\mathbf{x}) - q_{k+1}(\mathbf{x}^{k+1})\right)^+, \tag{4}$$

where $q_{k+1}(\mathbf{x}^{k+1})$ is the quantile of the updated Kriging metamodel at $\mathbf{x}^{k+1}$ ($\beta \in [0.5, 1]$). The next point is the point with maximum $\text{EQI}(\mathbf{x}) = E[I(\mathbf{x})]$; Picheny et al. (2013a) explain the calculation of this expectation. Again, this calculation requires an estimate of $\tau^2(\mathbf{x})$.

EQI allows for revisits, and additionally includes a replication step after each search step (see Figure 1). The number of replications per iteration is no longer fixed to a constant B. In the search step of iteration $k+1$, EQI samples the infill point $\mathbf{x}^{k+1}$ with n_{inc} replications and refits the Kriging metamodel. In the replication step, it checks whether performing another n_{inc} replications at $\mathbf{x}^{k+1}$ is beneficial, by looking at the evolution of $\text{EQI}(\mathbf{x}^{k+1})$ (see Picheny et al. (2013a) for details). If so, the algorithm adds another n_{inc} replications, updates $\bar{f}(\mathbf{x}^{k+1})$ and $\widehat{\text{Var}}[\bar{f}(\mathbf{x}^{k+1})]$, and refits the Kriging metamodel. This is repeated until adding extra replications is no longer beneficial: at this moment EQI leaves the replication step.

2.2.5 Two-stage sequential optimization (TSSO)

Analogous to EQI, TSSO has a search and a replication step (called *allocation stage* in Quan et al. (2013)). In the search step of iteration $k+1$, TSSO looks at all *unvisited* alternatives ($\mathbf{x} \in \Theta \backslash \Theta^k$) and simulates the one with maximum modified expected improvement (MEI):

$$\text{MEI}(\mathbf{x}) = E\left[\max\left(\hat{f}(\mathbf{x}_{min}) - f_p^*(\mathbf{x}), 0\right)\right], \tag{5}$$

where $\hat{f}(\mathbf{x}_{min})$ is the stochastic Kriging prediction at $\mathbf{x}_{min} = \arg\min_{\mathbf{x}\in\Theta^k} \bar{f}(\mathbf{x})$ (i.e., the alternative with the lowest sample mean among the already simulated points) and $f_p^*(\mathbf{x})$ is a normal random variable: $f_p^*(\mathbf{x}) \sim N(\hat{f}(\mathbf{x}), s_D(\mathbf{x}))$ where the mean $\hat{f}(\mathbf{x})$ is the stochastic Kriging prediction at solution $\mathbf{x}$, and $s_D^2(\mathbf{x})$ is the estimate of *deterministic* Kriging prediction error. TSSO does not require $\tau^2(\mathbf{x})$ for calculating $\text{MEI}(\mathbf{x})$, see Quan et al. (2013) for details. Note that, as TSSO only considers unvisited alternatives in the search step, revisits are not allowed.

Table 3: Analytical test functions

Function	Source	Description
Six-hump Camelback	Dixon and Szegö (1978)	dimension=2 $f(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + x_1^6/3 + x_1x_2 - 4x_2^2 + 4x_2^4$ $-2 \leqslant x_1 \leqslant 2$ and $-1 \leqslant x_2 \leqslant 1$ $x^* = (0.0977, -0.6973)$ $f(\mathbf{x}^*) = -1.0294, R_f \simeq 7.3$
Rescaled Branin	Dixon and Szegö (1978)	dimension=2 $f(x_1, x_2) = \frac{1}{51.95}\left[\left(\bar{x}_2 - \frac{5.1\bar{x}_1^2}{4\pi^2} + \frac{5\bar{x}_1}{\pi} - 6\right)^2 + \left(10 - \frac{10}{8\pi}\right)\cos(\bar{x}_1) - 44.81\right], \bar{x}_1 = 15x_1 - 5, \bar{x}_2 = 15x_2$ $0 \leqslant x_1 \leqslant 1$ and $0 \leqslant x_2 \leqslant 1$ $x^* = (0.541, 0.1348)$ $f(x^*) = -1.0459, R_f \simeq 6$
Hartmann-6	Dixon and Szegö (1978)	dimension=6 $f(x_1, ..., x_6) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} \alpha_{ij}(x_j - p_{ij})^2\right]$ c_i, α_{ij}, p_{ij} are parameters, see Dixon and Szegö (1978) $0 \leqslant x_i \leqslant 1$ for $i = 1, ..., 6$ $x^* = (0.2382, 0.1391, 0.3665, 0.3286, 0.3519, 0.7018)$ $f(\mathbf{x}^*) = -3.02, R_f \simeq 3.3$

Note: R_f is the range of the response ($\max_{\mathbf{x}\in\Theta} f(\mathbf{x}) - \min_{\mathbf{x}\in\Theta} f(\mathbf{x})$) in the region of interest.

The number of replications per iteration is fixed to a predetermined value B. This number is shared between the search and replication steps according to a heuristic (Quan et al., 2013). In the replication step, the algorithm uses the optimal computing budget allocation (OCBA) which is a Bayesian ranking and selection (R&S) procedure (Chen et al., 2000). OCBA allocates most of the replication step budget to points with low $\bar{f}(\mathbf{x})$ and high $\widehat{\text{Var}}[\bar{f}(\mathbf{x})]$ (Quan et al., 2013), in view of maximizing the probability of selecting the best point among the sampled ones.

2.2.6 Extended two-stage sequential optimization (eTSSO)

The eTSSO algorithm differs from the TSSO approach in the number of replications per iteration; while this was fixed to a given value B in TSSO, it now differs for each iteration k:

$$B_k = B_{k-1}\left(1 + \frac{\widehat{\text{Var}}[\bar{f}(\mathbf{x}_{OCBA})]}{\widehat{\text{Var}}[\bar{f}(\mathbf{x}_{OCBA})] + s_D^2(\mathbf{x}^k)}\right), \tag{6}$$

where $\widehat{\text{Var}}[\bar{f}(\mathbf{x}_{OCBA})]$ is the estimate of variance of the sample mean at the point where OCBA allocates the highest number of replications ($\mathbf{x}_{OCBA}$), and $s_D^2(\mathbf{x}^k)$ is the estimate of deterministic Kriging prediction error at the point that maximizes MEI at iteration k. In eTSSO, the search step budget of each iteration is fixed to a value (n_s) prior to running the algorithm: at iteration k, the replication step budget of eTSSO is thus $B_k - n_s$.

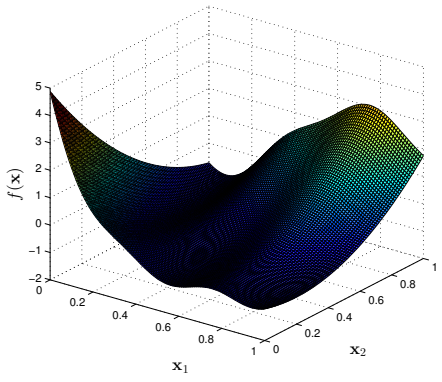
3 Benchmark design

In this section, we explain our experiments in detail. We apply the Kriging-based methods to three well-known analytical test functions; Section 3.1 provides the details. We also test our algorithms on the inventory problem of Fu and Healy (1997), see Section 3.2. Section 3.3 explains the method used for sampling the initial design and Section 3.4 discusses the different scenarios tested in our experiments. We discuss the implementation of the algorithms in Section 3.5. Finally, Section 3.6 explains the performance measures used to compare the algorithms.

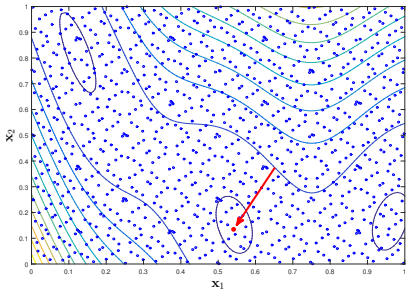
3.1 Analytical test functions and candidate points

The algorithms are applied to minimize three well-known analytical test functions (six-hump camelback, rescaled Branin, and Hartmann-6) (Dixon and Szegö, 1978), perturbed with heterogeneous Gaussian noise. Hence, we obtain noisy observations $\tilde{f}_j(\mathbf{x}^i) = f(\mathbf{x}^i) + \varepsilon_j(\mathbf{x}^i)$, with $\varepsilon_j(\mathbf{x}^i) \sim N(0, \tau(\mathbf{x}^i))$. Table 3 gives further details on these analytical test functions. All test functions are continuous; however, we discretize the domain to obtain a large but finite set of solutions. In this way, we avoid the challenging problem of maximizing the expected improvement criteria in continuous space; also, some of the criteria might be more difficult to maximize than others. We discretize the domains using Faure low discrepancy sequences (a quasi-Monte Carlo sampling method with good space-filling properties, see chapter 5 of Lemieux (2009)) and we take 1000 points for the camelback and Branin functions ($d = 2$), and 10000 points for the Hartmann-6 function ($d = 6$). These points represent our set of candidate points Θ; our objective is to find the global minimum among these alternatives (referred to as $\mathbf{x}^*$, having function value $f(\mathbf{x}^*)$) using our Kriging-based methods.

Figures 2 and 3 show an illustration of the rescaled Branin and the camel-back functions, along with their respective candidate points. As evident from the figures, the global minimum of camel-back lies within a small valley, while Branin has a large flat valley. Although the exact shape of Hartmann-6 is unknown, we know that it is a multimodal function.

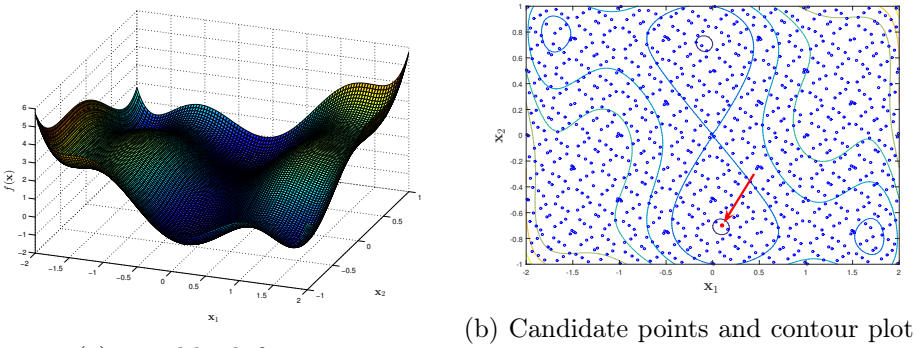


(a) Rescaled Branin function



(b) Candidate points and contour plot of Branin function

Figure 2: Rescaled Branin function and candidate points; $(\mathbf{x}^*, f(\mathbf{x}^*))$ is denoted by an arrow.



(a) camel-back function

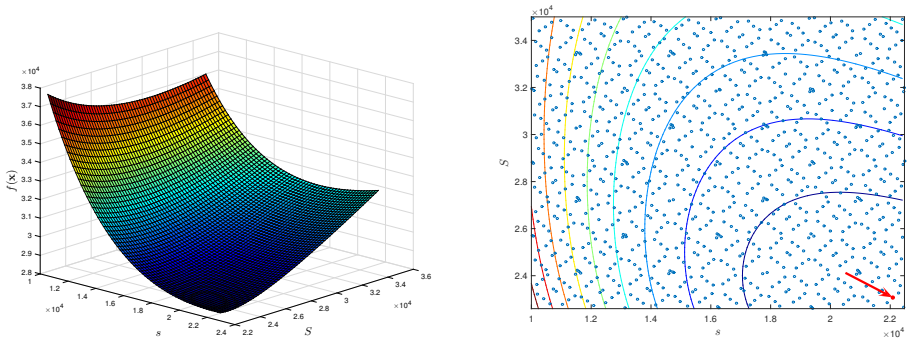
(b) Candidate points and contour plot of the camel-back function

Figure 3: Six-hump camel-back function and candidate points; $(\mathbf{x}^*, f(\mathbf{x}^*))$ is denoted by an arrow.

3.2 The (s, S) inventory problem and candidate points

In the inventory control problem of Fu and Healy (1997), a firm employs a periodic review (s, S) policy to manage the inventory of a single product. At regular intervals (e.g., each week), the firm checks the inventory. If the inventory position (i.e., inventory on-hand + on-order − backorder) is above the level s, the firm does not order. If it is below s, the firm orders the difference between the order-up-to level S and the inventory position. Demand is stochastic and continuous and is i.i.d across the periods. The goal is to determine the optimal s and S in view of minimizing the long-run expected cost per period. The costs include the fixed ordering (K), per-unit ordering (c), holding (h), and backorder (b) cost. Zero replenishment lead time and full backlogging are assumed.

To simulate this problem, we use a replication length of 1000 periods with a warm-up length of 100 periods. The inventory parameters (s, S) are continuous; we again use the Faure sequence to take 1000 space filling solutions (Figure 4). With exponential demand, the expected cost function has a closed form and the optimal parameters can be calculated analytically; we can thus evaluate the performance of our Kriging-based methods (see Table 4 which gives details about the parameters used in our experiment; based on Table 1 in Fu and Healy (1997)).



(a) expected cost function

(b) Candidate points and contour plot of the expected cost function

Figure 4: Expected cost function of the (s, S) inventory problem with exponential(λ) demand; $(\mathbf{x}^*, f(\mathbf{x}^*))$ is denoted by an arrow.

Table 4: (s, S) inventory problem with exponential(λ) demand

	Source	Description
periodic review (s, S) problem	Fu and Healy (1997)	dimension=2 $f(s,S) = \frac{c}{\lambda} +$ $[K + h(s - \frac{1}{\lambda} + 0.5\lambda(S^2 - s^2)) + \frac{h+b}{\lambda} e^{-\lambda s}]/[1 + \lambda(S - s)]$ $K = b = 100, c = h = 1, \lambda = 0.0002$ $10000 \leqslant s \leqslant 22500$ and $22600 \leqslant S \leqslant 35000$ $x^* = (22084.9609, 23060.1563)$ $f(\mathbf{x}^*) = 28165.0049, R_f \simeq 8584$

Note: R_f is the range of the response ($\max_{\mathbf{x}\in\Theta} f(\mathbf{x}) - \min_{\mathbf{x}\in\Theta} f(\mathbf{x})$) in the region of interest.

3.3 Initial design and Kriging metamodeling

For the initial design, we follow the suggestion of Jones et al. (1998): $n_0 = 10d$ (so 20 initial points for the camel-back and rescaled Branin functions, as well as for the (s, S) inventory problem; 60 initial design points for the Hartmann-6 function). A maximin Latin hypercube sample (LHS) is used to obtain these points (as is common in Kriging metamodelling, see Picheny et al. (2013b)); we determine $\bar{f}(\mathbf{x}^i)$ for each $i = 1, ..., n_0$ using 55 replications (in view of controlling the noise, see Section 3.4). As prevalent in the literature, we only consider an intercept term in the Kriging metamodel (i.e., no trend term). We also don't use common random numbers (CRN): as shown by Chen et al. (2012), CRN increases the stochastic Kriging variance and variance of the intercept estimation. We fit the Kriging metamodel by means of maximum likelihood; the classical Matern 2.5 is used as the covariance function (see Rasmussen and Williams (2006) chapter 4).

3.4 Test Scenarios

There are many factors that can influence the performance of the methods: some factors are *internal* or method-specific parameters (e.g., β in SKO), while others are *external* (such as the replication budget, the noise magnitude and noise structure). Table 5 provides an overview.

We test our algorithms using low and high replication budgets; this refers to the replication budget available *after* simulating the n_0 initial points, for performing the search and replication steps. We focus on problems for which a very small number of solution evaluations are possible, e.g., because the simulation is very time consuming. The number of replications per iteration (B) is equal to 55 for SKO, CKG, TSSO, and MQ, implying that with low (resp. high) budget we simulate 10 (resp. 50) infill points for these methods. In TSSO, the number of replications per infill point is usually less than 55, since 55 replications are shared between the search and replication steps (see Section 2.2.5). In EQI and eTSSO, B is not fixed; the number of infill points is thus not known prior to running these algorithms.

In real-life problems, the noise can follow any type of structure. In the (s,S) inventory problem this structure is unknown. For the 3 analytical test functions, we assume that the standard deviation of the noise (i.e., $\tau(\mathbf{x})$) varies linearly in terms of the objective value with its maximum and minimum linked to R_f, i.e., the range of the objective value in the region of

Table 5: External Factors considered

Factors	**levels**
Replication budget	Low: 550 replications High: 2750 replications
Number of replications per iteration (for SKO, CKG, TSSO, and MQ)	$B = 55$
Noise structure for the 3 analytical test functions	**Best case** $\min_{x\in\Theta} \tau(\mathbf{x})$ at $\min_{x\in\Theta} f(\mathbf{x})$ $\max_{x\in\Theta} \tau(\mathbf{x})$ at $\max_{x\in\Theta} f(\mathbf{x})$ **Worst case** $\min_{x\in\Theta} \tau(\mathbf{x})$ at $\max_{x\in\Theta} f(\mathbf{x})$ $\max_{x\in\Theta} \tau(\mathbf{x})$ at $\min_{x\in\Theta} f(\mathbf{x})$
Noise magnitude for the 3 analytical test functions	**Light noise** $\min_{x\in\Theta} \tau(\mathbf{x}) = 0.15R_f$, $\max_{x\in\Theta} \tau(\mathbf{x}) = 0.6R_f$ **Heavy noise** $\min_{x\in\Theta} \tau(\mathbf{x}) = 1.5R_f$, $\max_{x\in\Theta} \tau(\mathbf{x}) = 6R_f$

Note: R_f is the range of the objective value in the region of interest.

interest (as in Huang et al. (2006) and Picheny and Ginsbourger (2014)). With light noise, $\tau(\mathbf{x})$ varies between 15% and 60% of R_f. With heavy noise, these numbers increase to 150% and 600% of R_f. We sample the initial designs with 55 replications: for these points the noise on $\bar{f}(\mathbf{x})$ varies between $1.5/\sqrt{55}R_f = 0.2R_f$ and $6/\sqrt{55}R_f = 0.8R_f$ with heavy noise. The noise structure may be in general very case-dependent and adopt a large variety of shapes, here we focus on two cases:

1. **Best case**: the noise standard deviation decreases linearly as the objective value decreases; therefore we have minimum noise at the global minimum.

2. **Worst case**: the noise standard deviation increases linearly as the objective value decreases; we then have maximum noise at the global minimum.

We refer to appendix B for details on the linear functions used to implement the different noise structures.

As mentioned earlier, CKG, EQI, and SKO need an estimate of $\tau^2(\mathbf{x})$ at all the candidate points. For the inventory test problem, the noise structure is unknown. Analogous to Section 3.1 in Ankenman et al. (2010), we fit a *deterministic* Kriging model to the sampled variances of the simulated function values and use it to predict the noise variance at any $\mathbf{x} \in \Theta$. In the analytical test functions, however, the noise structure is known: as mentioned above, we assume a linear relation between the noise magnitude and the function value ($\tau(\mathbf{x}) = af(\mathbf{x}) + a \times b$). We then estimate the noise at $\mathbf{x} \in \Theta$ by using the Kriging prediction $\hat{f}(\mathbf{x})$ instead of the unknown true function value: $\hat{\tau}(\mathbf{x}) = a\hat{f}(\mathbf{x}) + a \times b$.

Figure 5 illustrates the structure of heavy worst case and heavy best case noise for the Branin function, at $x_2 = 0$. Here $R_f = 6$, so the maximum and minimum standard deviation of noise

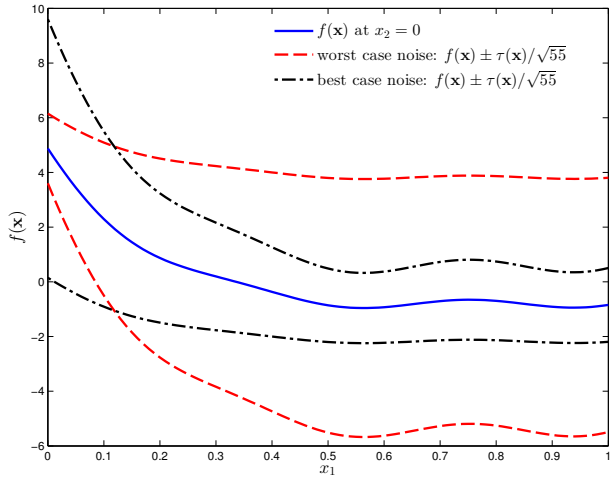
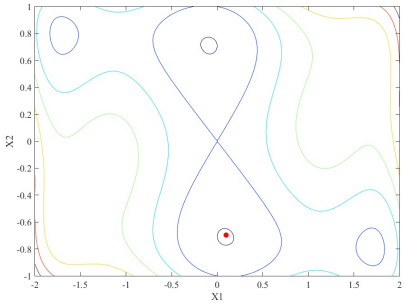
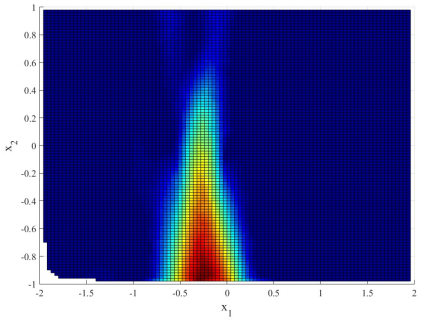


Figure 5: Heavy worst case and best case noise for rescaled Branin function at $x_2 = 0$ with 55 replications

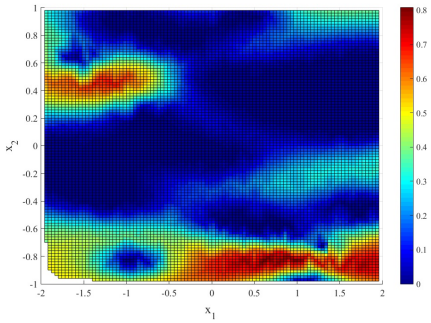
with 55 replications are $0.8 \times 6 = 4.8$ and $0.2 \times 6 = 1.2$, respectively. As shown in the figure, with best case noise, this maximum occurs at $(x_1 = 0, x_2 = 0)$, and the noise drops linearly as $f(\mathbf{x})$ decreases. With worst case noise, we have the opposite behavior.



(a) Camel-back function, the location of the optimum is indicated in red.



(b) MEI for best case heavy noise



(c) MEI for worst case heavy noise

Figure 6: MEI of camel-back function for the first iteration of a given run, with best and worst case noise

The noise structure may substantially impact the performance of the algorithms, as it impacts the search pattern in step 2. As an illustration, Figure 6 shows the structure of MEI for the

Table 6: Parameter settings for different algorithms

Method	**Factors**
MQ	$\beta = 0.1$
SKO	$\beta = 0.84$
EQI	$\beta = 0.5, n_{inc} = 10, \alpha = 0.5$
TSSO	$n_{min} = 2$
eTSSO	$n_s = 10$

Note: CKG does not require any parameters

camel-back function, with best and worst case noise, at the first iteration of an arbitrary run (so only the n_0 design points have been simulated). With best case noise, the maximum MEI is near the optimal solution; with worst case noise, the search may be driven to non-interesting areas of the search space.

Table 6 provides the parameter settings of the different algorithms. Huang et al. (2006) recommend $\beta = 0.84$ for SKO ($\Phi^{-1}(0.84) = 1$); for the β values in MQ and EQI we follow the suggestions of Picheny et al. (2013b). We follow the recommendation of Picheny et al. (2013a) for α in EQI; we set n_{inc}=10. In TSSO, n_{min} (i.e., the minimum number of replications for simulating an infill point) only affects the way B is shared between the search and replication steps (see Quan et al. (2013)). In eTSSO, we chose $n_s = 10$ to match n_{inc} of EQI.

3.5 Implementation details

For best case as well as for worst case noise, we have the following 4 settings: 1) Low budget – Light noise 2) Low budget – Heavy noise 3) High budget – Light noise 4) High budget – Heavy noise. These are applied to each of the 3 analytical test functions, yielding a total of $2 \times 4 \times 3 = 24$ scenarios. For the inventory problem, we have 2 scenarios: low budget and high budget. Recall that the budget is for the search and replication steps (steps 2 and 3 in Figure 1). To evaluate the performance of an algorithm for a given scenario, we use 100 macroreplications: we run the algorithm 100 times, each time using a different initial design. In a given macroreplication, all methods start with the same set of initial points (same $\mathbf{x}^i$, $\bar{f}(\mathbf{x}^i)$, and $\widehat{\text{Var}}[\bar{f}(\mathbf{x}^i)]$) and, hence, the same Kriging model (analogous to Picheny et al. (2013b)).

All the methods were coded in MATLAB. To fit the stochastic Kriging models, we used the code provided on http://stochastickriging.net/, for CKG we used the code on http://people.orie.cornell.edu/pfrazier/src.html, the TSSO code was based on http://www.ise.nus.edu.sg/staff/ngsh/download/matlabdocs/SOK/ and the eTSSO code was obtained from the authors of Liu et al. (2014). The remaining algorithms were coded by the authors themselves and compared with the DiceOptim package in R (Picheny and Ginsbourger, 2014) to ensure correctness.

As most of the methods allow for revisits, the covariance matrix of the metamodel may become ill-conditioned; we follow the guidelines in Section 4.3 of Picheny and Ginsbourger (2014) for handling this issue. Table 7 shows the average running time of different algorithms for one macroreplication in all the test problems (on a Dell desktop with 3.4 GHz Core i7-2600 processor and 8GB of RAM). For the analytical test functions, we focused on heavy worst case noise; the structure or magnitude of noise had only minor effects on the running times.

Table 7: Average running time of different algorithms for one macroreplication (high budget) in seconds.

Method	(s, S) **problem**	**Camel-back**	**Branin**	**Hartmann-6**
MQ	17	20	19	61
SKO	31	24	31	91
CKG	385	384	379	88734
EQI	133	109	73	256
TSSO	26	24	27	121
eTSSO	4	6	4	17

As evident from the table, CKG and then EQI have the longest running time, while eTSSO requires the smallest computation time followed by MQ. In eTSSO, the number of replications per iteration quickly increases (see Equation 6) and the budget is depleted after a relatively small number of iterations: this explains its short running time. CKG requires a lot of time to calculate the expected improvement of all 10000 feasible alternatives in Hartmann-6 and thus, as shown in the table, takes around 24 hours per macroreplication. For the experiments in Section 4, we implemented this algorithm on a server with 5 cores and 100 threads, running each macroreplication on a different thread. However, in many problems, the feasible region contains millions of points (Xu, 2012); the computational requirements may limit the applicability of CKG in these cases.

3.6 Performance measures

We use three performance measures to compare the algorithms in each scenario. Let $\mathbf{x}_m^r$ be the point returned by the algorithm in the identification step at macroreplication $m = 1, 2, ..., 100$. $GAP_m = f(\mathbf{x}_m^r) - f(\mathbf{x}^*)$ then represents the difference in objective value between the alternative suggested by the algorithm and the optimal solution. The distribution of $GAP = (GAP_1, GAP_2, ..., GAP_{100})$ is our first performance measure, visualized by means of a boxplot for each of the algorithms.

A bad GAP performance can occur because (1) the algorithm fails to discover promising alternatives in the search step, so Θ^T does not contain interesting points; or (2) promising alternatives are present in Θ^T, but the algorithm fails to identify them in the identification step. To investigate this, we introduce two additional performance measures. The first measure (NV_χ) counts the number of macroreplications in which the algorithm *visited* at least one point that has a goal function value within the $(1 - \chi) \times 100\%$ range from the true optimum $f(\mathbf{x}^*)$. A low NV_χ for a given algorithm thus indicates that the algorithm's search criterion failed to locate promising points for that scenario. In this case, GAP results will also be disappointing. The second measure (NR_χ) counts the number of macroreplications for which the *returned* solution is within the $(1 - \chi) \times 100\%$ range of $f(\mathbf{x}^*)$. Evidently, we always have: $NR_\chi \leq NV_\chi$. When NV_χ is high but NR_χ is low, this indicates that the algorithm has visited good alternatives in the search step, but fails to recognize them in the identification step (i.e., the algorithm has an identification problem), which in turn causes bad GAP results. For camel-back and Branin, we use $\chi = 0.95$. For the Hartmann-6 function, we used $NR_{0.8}$ and $NV_{0.8}$ since $NV_{0.95}$ was usually

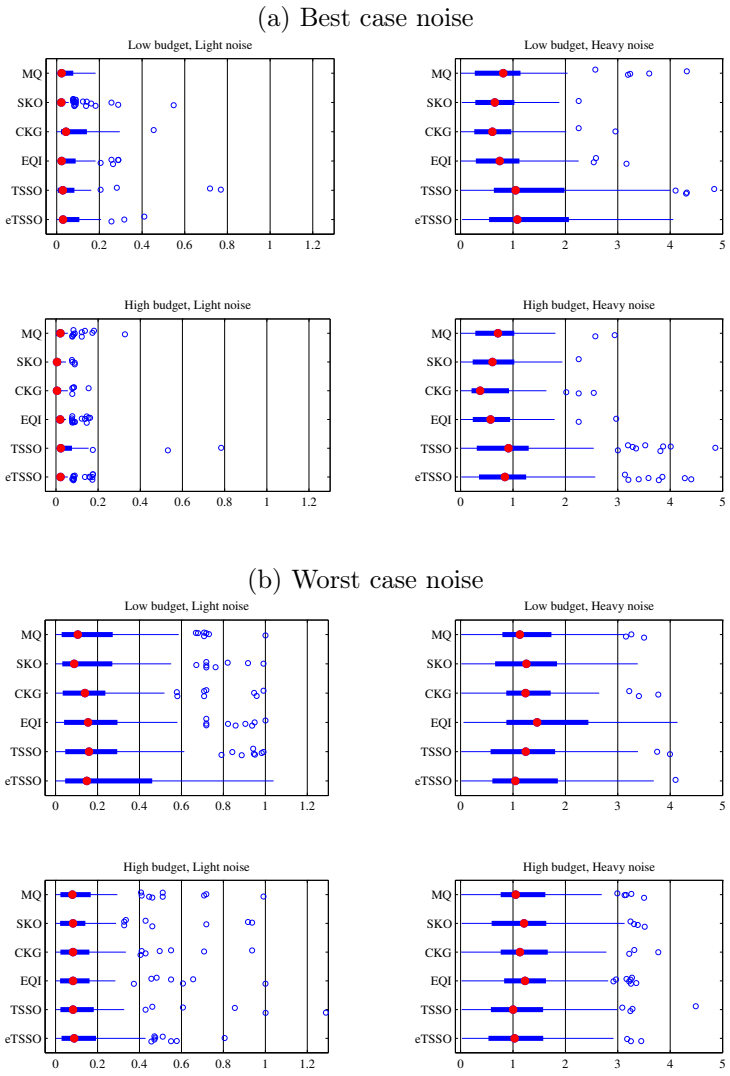


Figure 7: *GAP* for the camel-back function ($R_f = 7.3$). At each scenario, the boxplots show the distribution of *GAP* for each distinct method. The median is indicated in red; the edges of the boxes are the 25th and 75th percentiles.

too low (≤ 10) for this function. In the (s, S) inventory problem, we focused on $NR_{0.999}$ and $NV_{0.999}$ as all methods returned a solution within 1% of $f(\mathbf{x}^*) = 28165$ in all macroreplications (this has to do with both the convexity of the problem, and the fact that the objective value of many points in the search space is very close to $f(\mathbf{x}^*)$).

4 Results

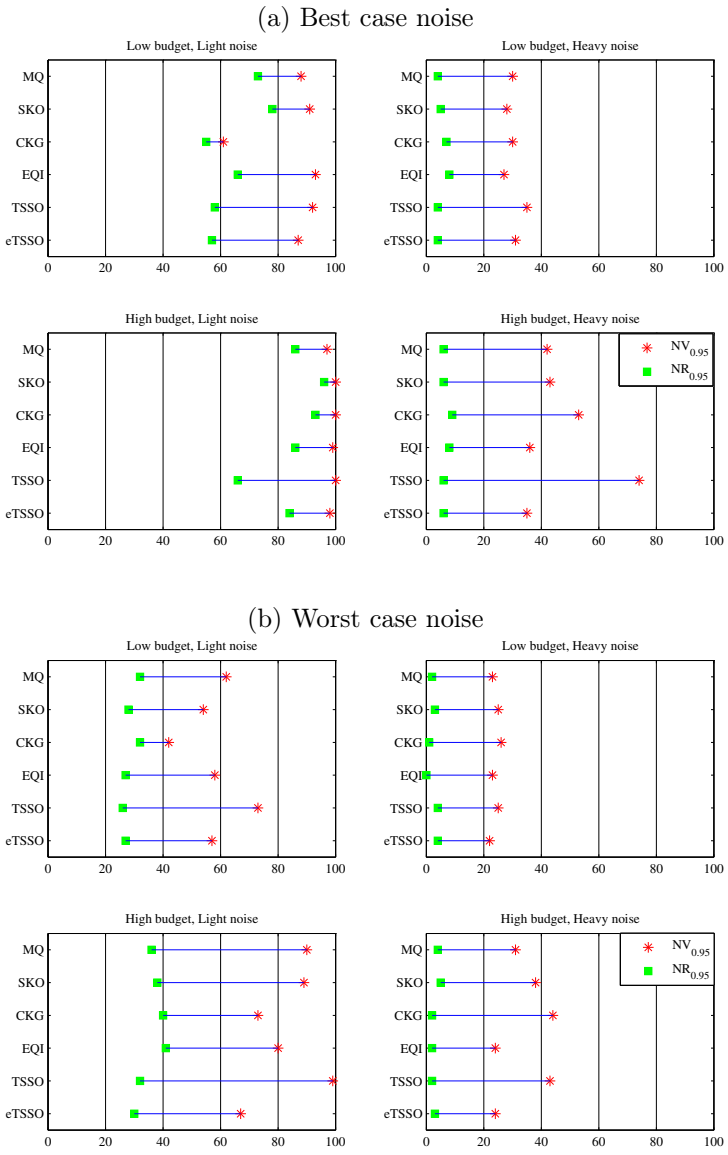
In this section, we provide the results of our analysis. As outlined above, we study the following research questions:

1. Does the structure and magnitude of the noise and the replication budget affect the performance of the algorithms?

2. Do the algorithms that require an estimate of the noise variance function in step 2 (CKG, SKO, EQI) outperform the others that don't need this information? How large is the difference?

3. How do the algorithms perform in comparison to our benchmark method (MQ)?

Figure 8: $NV_{0.95}$ (indicated in red) and $NR_{0.95}$ (indicated in green) for the camel-back function.



4. How do the algorithms compare in terms of their search strategies? Do they manage to locate good solutions among Θ (high NV_χ)?

5. Are the replication strategies of the algorithms effective? Do they help the algorithms to distinguish the superior solutions in the identification step, or do algorithms experience identification problem (large $NV_\chi - NR_\chi$)?

In Section 4.1, we discuss the effect of the external factors proposed in Table 5. Section 4.2 details the performance of the algorithms with respect to our measures: GAP, NV_χ, and NR_χ. As the results were similar for the camel-back, Branin, and the (s, S) problem, we moved the results of the two latter ones to appendix C and appendix D, respectively; only the results for camel-back (Figures 7, 8, and 11) and Hartmann-6 (Figures 9, 10, and 12) are discussed in the text.

4.1 Effect of external factors

The effect of the external factors on the performance of the algorithms is largely the same for the different test problems. From figures 7–10, we can derive the following effects:

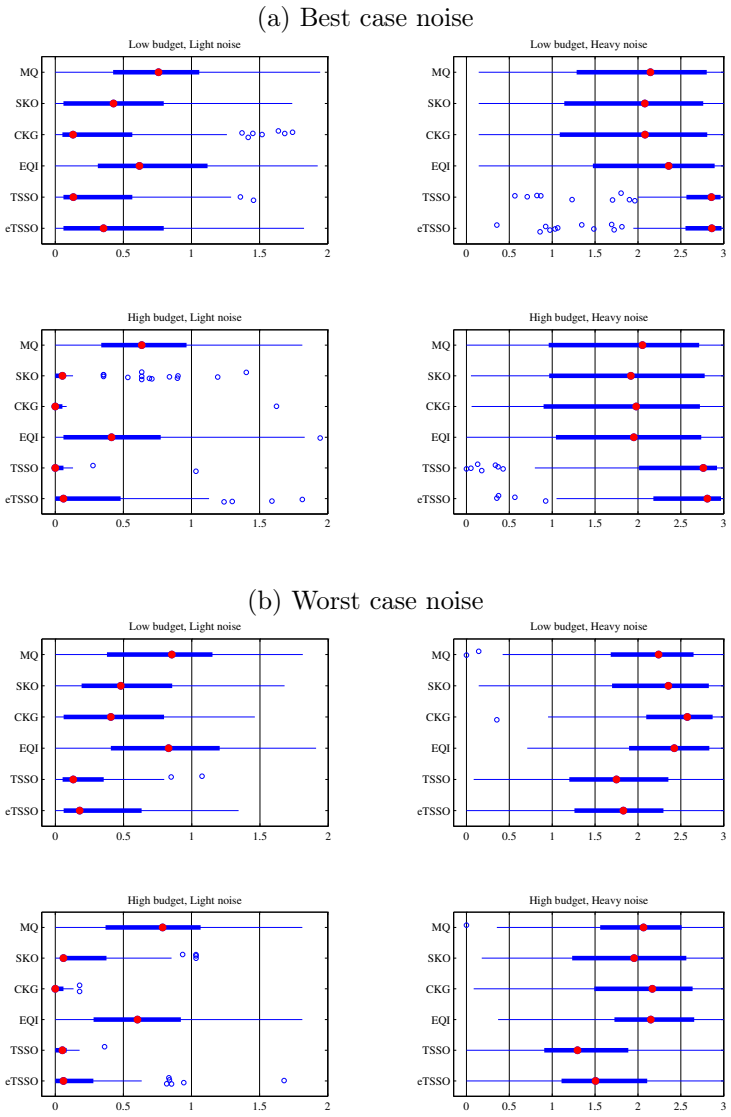
- **Replication budget**: comparing low and high budget figures reveals that a higher replication budget enhances the performance of all algorithms with respect to all measures; for the analytical test functions, though, the difference is smaller with heavy noise. While we see some improvements in $NV_{0.95}$, a higher budget has a minor impact on $NR_{0.95}$ and GAP with heavy noise (i.e., simulating more infill points does not increase $NR_{0.95}$ or improve GAP with heavy noise).
- **Noise structure** (for the analytical test functions): Figures 7–10 show that the performance of all methods usually degrades with worst case noise comparing to best case noise, for all scenarios.
- **Noise magnitude** (for the analytical test functions): Comparing light and heavy noise figures, we see that stronger noise has a substantial negative impact on the performance of all algorithms, with respect to all measures. Moreover, noise magnitude has a stronger influence on the performance of the methods than noise structure.

4.2 Comparison with respect to the performance measures

To compare the Kriging-based methods, we focus on the three measures one at a time and then combine our observations:

- GAP: There is no method that clearly outperforms the others in all the tested scenarios. The benchmark method MQ is very competitive in camel-back, Branin, and the (s, S) problem: only CKG and SKO tend to outperform MQ. TSSO and eTSSO give the largest GAP on average in the analytical test functions, especially with best case noise. Their performance is also disappointing in the (s, S) problem (see Appendix D). The results for Hartmann-6 are somewhat different: as evident from Figure 9, MQ is less competitive (especially with light noise); together with EQI, it usually gives the largest GAP. TSSO usually provides the best outcome (except with heavy best case noise), and is similar in performance to CKG.
- NV_χ: In the (s, S) inventory problem, $NV_{0.999} \simeq 100$ for all algorithms except eTSSO. In the analytical test functions, TSSO tends to locate better alternatives than other methods, while EQI usually yields a low NV_χ. MQ is also competitive for this measure for the camel-back and Branin functions; for Hartmann-6, though, MQ performs as badly as EQI.
- NR_χ: the large difference between NV_χ and NR_χ, shows that all algorithms have an identification problem (see Figure 8 for the camel-back function, Figure C.2 for the Branin function, and Figure D.1 for the inventory example). This problem seems to be most severe for TSSO. Figure 10 indicates that for the Hartmann-6 function, the identification problem tends to be less severe: i.e., $NV_{0.8} - NR_{0.8}$ tends to be smaller for all methods (except for TSSO with heavy noise).

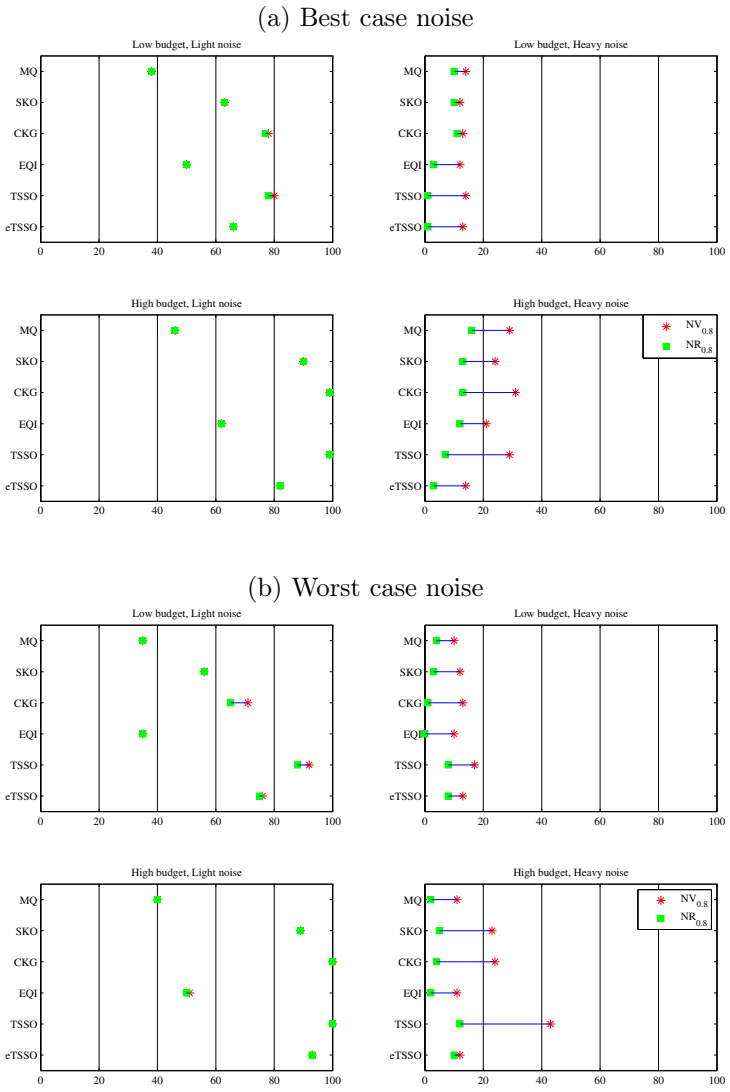
Figure 9: GAP for the Hartmann-6 function ($R_f = 3.3$). At each scenario, the boxplots show the distribution of GAP for each method. The median is indicated in red; the edges of the boxes are the 25th and 75th percentiles.



The above observations show that, among the methods that require $\tau^2(\mathbf{x})$ in step 2, CKG and SKO usually outperform the benchmark MQ method, while EQI does not. This holds even in the (s, S) inventory problem where $\tau^2(\mathbf{x})$ is estimated via a metamodel, see Appendix D. Analogous to MQ, TSSO and eTSSO don't require an estimate for $\tau^2(\mathbf{x})$ in step 2. Among these methods, TSSO is most successful in discovering very good alternatives (i.e., it has a good search strategy), but it usually fails to identify them at the end of the algorithm (especially in the camel-back, Branin, and the inventory problem). This also explains its relatively bad GAP performance for these examples. It thus seems that the replication strategy of TSSO is not very successful: the algorithm usually can't distinguish between the inferior and good solutions in the identification step. In Hartmann-6, identification is less of an issue: we see that TSSO performs as well as CKG. Comparing to TSSO, eTSSO does slightly better in the identification stage (i.e., lower $NV_\chi - NR_\chi$) but that comes at the cost of lower NV_χ: Figures 7 and 9 show that TSSO usually provides a better GAP than eTSSO (the same holds for the Branin function and the inventory problem, see Figures C.1 and D.1).

The replication strategy of the algorithms also affects their performance in the search step.

Figure 10: $NV_{0.8}$ (indicated in red) and $NR_{0.8}$ (indicated in green) for Hartmann-6 function



Appendix F shows the number of *distinct* infill points (i.e., without counting the revisits) sampled by the algorithms, for the high budget scenarios. As evident from these figures, TSSO always samples 50 distinct infill points as it does not allow for revisits. All other algorithms sample less distinct points. This can also contribute to the relatively high NV_χ for TSSO. EQI and eTSSO usually yield the lowest number of distinct infill points. Especially with heavy noise, EQI becomes very "conservative", and spends a lot of replications to get more accurate estimates for the objective value: the replication budget is depleted after sampling a few infill points. In eTSSO, the number of replications per iteration quickly increases (see Equation 6), implying that the budget is also depleted after a relatively small number of iterations.

The above observations lead us to conjecture that if the identification problem of TSSO could be solved, this algorithm would probably provide much better GAP results, and might even be preferred over CKG or SKO (as it does not require $\tau^2(\mathbf{x})$ in step 2). This could be done by either changing the replication strategy used by the algorithm (for instance, using other ranking and selection procedures instead of OCBA), or by modifying the identification criterion. Figures C.3, C.4, E.1, and E.2 in Appendix show, for instance, the change in performance when the identification criterion of TSSO is changed to choose the point with minimum Kriging

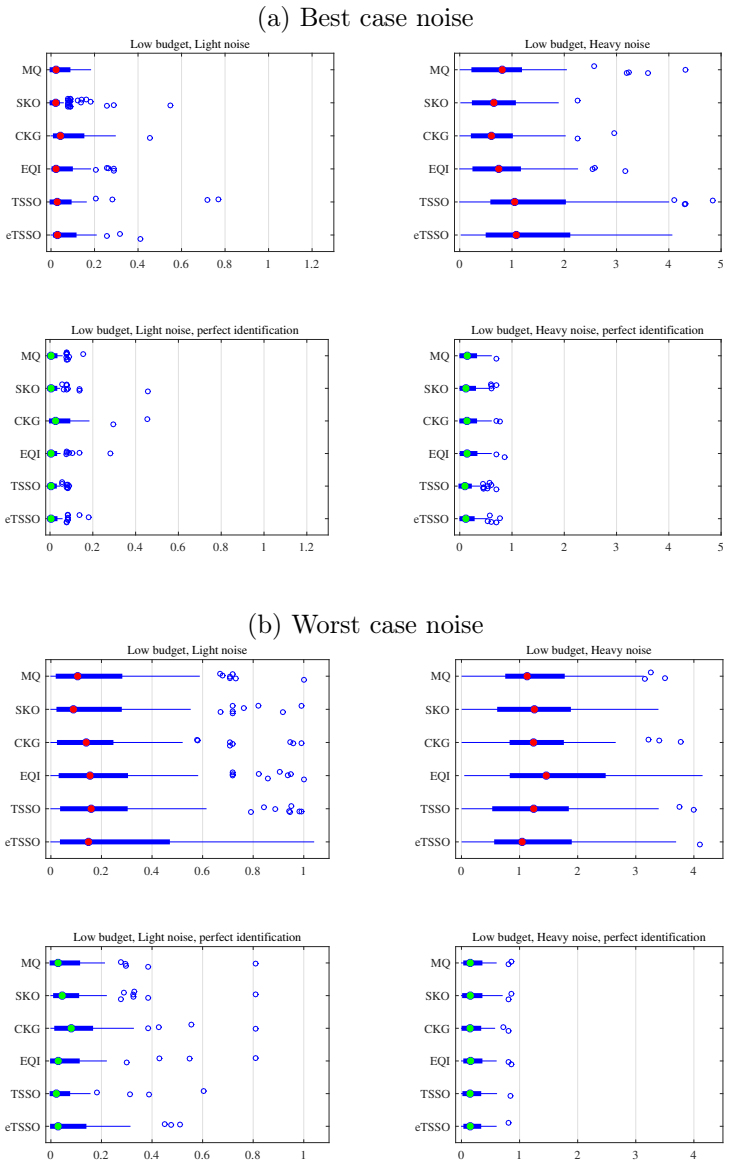


Figure 11: Comparing previous GAP results of the camel-back function ($R_f = 7.3$) with GAP results using perfect identification, for all algorithms. At each scenario, the boxplots show the distribution of GAP for each method. The median of the previous GAP is indicated in red; the median of the new GAP (with perfect identification) is indicated in green. The edges of the boxes are the 25th and 75th percentiles.

prediction (as in CKG) instead of the point with minimum sample mean. This modified TSSO (MTSSO) almost always gives a better GAP performance than TSSO, and has a performance similar to MQ. This illustrates that the identification criterion has a significant effect on the outcome of Kriging-based algorithms.

Although it is more acute in TSSO, *all* methods have identification issues in the camel-back and Branin functions (except when we have best case light noise with high budget), and in the inventory problem. The replication strategies of the algorithms are thus, in general, not helpful in the identification step. While increasing B (replication budget per iteration) might, at first sight, seem the obvious way to mitigate this problem, it is in general not easy to find the right B in practical problems where the optimal solution is unknown. Moreover, a higher B will cause us to sample less infill points. While some algorithms determine B dynamically during the run of the algorithm (i.e., eTSSO and EQI), our results reveal that these are not very successful.

We thus believe that it is essential to focus on smarter replication strategies. In our opinion, a ranking and selection (R&S) procedure that acts as a "clean-up" in the identification step may

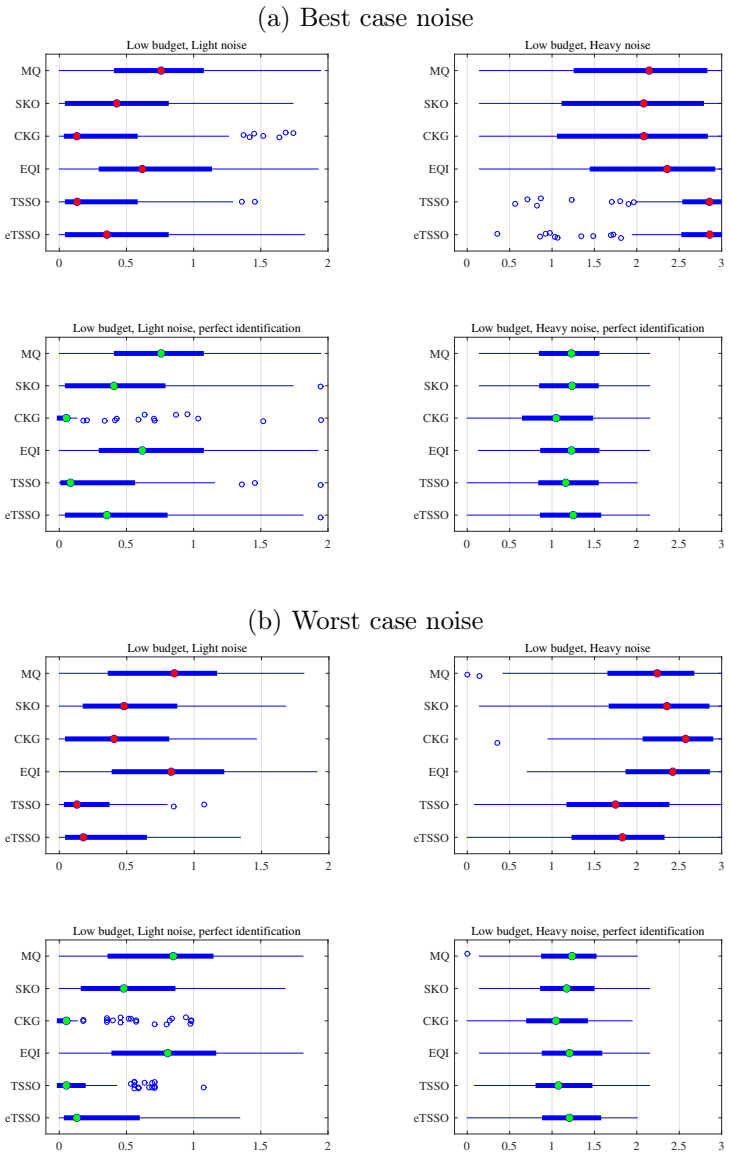


Figure 12: Comparing previous GAP results of Hartmann-6 ($R_f = 3.3$) with GAP results using perfect identification, for all algorithms. At each scenario, the boxplots show the distribution of GAP for each method. The median of the previous GAP is indicated in red; the median of the new GAP results (with perfect identification) is indicated in green. The edges of the boxes are the 25th and 75th percentiles.

provide an appropriate technique. The objective of this R&S procedure would be to discard the bad alternatives in Θ^T and find a sufficiently good one using as few replications as possible (Boesel et al. (2003) provide a good candidate procedure). Alternatively, R&S could be used in the optional replication step (see Figure 1); Hong and Nelson (2007), for instance, propose a R&S procedure to identify the best alternative among the sampled ones in each iteration. In addition to mitigating the identification problem, this approach might help the algorithm to choose better infill points in the search step (since, with a certain probability, it enables the algorithm to identify the true current best point). While the OCBA technique (Chen et al., 2000) used in TSSO tries to achieve a similar goal, our results indicate that it is not very effective in resolving the identification problem.

For the camel-back function, Figure 11 shows that we would have obtained much better results for all algorithms if we had achieved *perfect* identification in the last step (the search strategies are unchanged). Although perfect identification is unlikely to result from any given R&S method, these results illustrate the best possible improvement in GAP that we could hope to obtain.

Evidently, R&S consumes replication budget. Note, however, that in our case, all algorithms provide better GAP performance in low budget scenarios with perfect identification (green boxplots in Figure 11) than in their corresponding high budget scenarios in Figure 7. Therefore, reducing the budget of the search step to allow for a suitable R&S could be beneficial (the same observations hold for the Branin function and the inventory problem, see Figures C.5 and D.2). In Hartmann-6, R&S is less useful since the criteria employed by the algorithms usually achieve good identification. Only with heavy noise, R&S might still be beneficial (Figure 12).

Further research is needed on how to combine Kriging-based algorithms with R&S to achieve an appropriate replication strategy. In addition, deciding on how much replication budget to allocate to the R&S step is not straightforward. For instance, when using R&S in a clean-up phase, we should decide on when to stop simulating infill points and move to the R&S stage.

5 Conclusions and future research

In this paper, we have compared six recent Kriging-based methods for continuous optimization via simulation with heterogeneous noise. Using Kriging-based algorithms for optimizing stochastic simulation models (especially with heterogeneous noise) is relatively new, and provides an active research area: we think that the insights obtained in this article can be useful in improving these methods, and give useful suggestions for researchers who wish to employ these algorithms for solving engineering/business problems.

Our results showed that not only the magnitude, but also the shape of the noise can affect the performance of the algorithms. More specifically, we observed that with worst case noise (highest noise in the global minimum region) all algorithms performed worse than with best case noise (lowest noise in the global minimum region). We also showed that increasing the replication budget could improve the performance of the methods, especially when the noise magnitude was low.

CKG, SKO, and EQI require an estimate for the noise variance function but in practice, finding a reasonable estimate can be challenging. Among these methods, CKG and SKO tend to outperform our benchmark method (MQ), while EQI does not. CKG is computationally very intensive, especially when the number of candidate points is high (e.g., 10000 feasible alternatives in the Hartmann-6 function). As, in some optimization problems, the feasible space includes millions of solutions, applying CKG to these problems might be impractical. In these cases, one can use the approximate knowledge gradient policy which employs an approximation of the expected improvement function to speed up the algorithm (Scott et al., 2011; these authors showed that this policy performs similarly to SKO, at least in a setting with homogeneous noise).

Analogous to MQ, TSSO and eTSSO do not require information about the noise variance function. We did not observe a big difference between TSSO and eTSSO; TSSO usually even gave better results. The quality of the solutions returned by TSSO was strongly affected by its inability to identify good solutions among the simulated ones in the identification step: if it were successful in identifying the best alternatives, it would give significantly better results than MQ, and would perform as well as CKG or SKO. It would likely be preferred over CKG or SKO, though, since it does not require information with regard to the noise variance function.

Our results indicate that, especially with heavy noise, *all* algorithms have problems with the identification step. Combining the algorithms with smarter replication strategies (e.g., using appropriate ranking and selection methods) is, therefore, essential to improve their performance. In our opinion, this is a key area for future research.

Acknowledgments

This research was supported by the Research Foundation-Flanders (FWO)(grant no. G.0822.12). We would like to thank Professor Szu Hui Ng, Department of Industrial & Systems Engineering, National University of Singapore, for sharing eTSSO code with us. Some of the computational resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Hercules Foundation and the Flemish Government – department EWI.

Appendix A Kriging expressions

In this appendix we provide formulas for the stochastic Kriging prediction $\hat{f}(\mathbf{x})$ and its estimated variance $s^2(\mathbf{x})$. Stochastic Kriging (without trend term) assumes that the unknown true function can be represented by $f(\mathbf{x}) = \beta + \mathrm{M}(\mathbf{x})$, where $\beta \in \mathbb{R}$ is an unknown constant term and M is a realization of a mean zero Gaussian random field (Ankenman et al., 2010). We assume *spatial correlation*: $\mathrm{M}(\mathbf{x}^h)$ and $\mathrm{M}(\mathbf{x}^l)$ tend to be close if the distance $\|\mathbf{x}^h - \mathbf{x}^l\|$ is small. As this distance goes to infinity, $\mathrm{Corr}[M(\mathbf{x}^h), M(\mathbf{x}^l)] \to 0$ and if the distance is zero, $\mathrm{Corr}[M(\mathbf{x}^h), M(\mathbf{x}^l)] = 1$ (the implied covariance is denoted by $\mathrm{Cov}[M(\mathbf{x}^h), M(\mathbf{x}^l)]$). The way the spatial correlation changes with distance $\|\mathbf{x}^h - \mathbf{x}^l\|$ depends on the employed correlation function (As noted in Section 3.3, we used Matern 2.5 in our experiments). The observed goal value in the jth simulation replication at design point $\mathbf{x}^k$, can thus be written as:

$$\tilde{f}_j(\mathbf{x}^k) = \beta + \mathrm{M}(\mathbf{x}^k) + \varepsilon_j(\mathbf{x}^k), \tag{7}$$

where, as mentioned in Section 1, $\varepsilon_j(\mathbf{x}^k)$ has mean zero and variance $\tau^2(\mathbf{x}^k)$ and represents the noise inherent in a stochastic simulation model. This noise is independent and identically distributed across replications and since we don't consider common random numbers (CRN), $\mathrm{Corr}[\varepsilon_j(\mathbf{x}^h), \varepsilon_j(\mathbf{x}^l)] = 0$.

Assume we have a Kriging metamodel fitted using k design points $(\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^k)$ with estimated function values $\bar{\mathbf{f}} = (\bar{f}(\mathbf{x}^1), \bar{f}(\mathbf{x}^2), ..., \bar{f}(\mathbf{x}^k))^{\mathrm{T}}$. These estimates are sample means of the simulation output across $(n^1, n^2, ..., n^k)$ replications. If the Kriging metamodel only has an intercept term, we obtain the Kriging prediction at an arbitrary point $\mathbf{x}^i$ as follows (Ankenman et al., 2010):

$$\hat{f}(\mathbf{x}^i) = \beta + \boldsymbol{\Sigma}_{\mathrm{M}}(\mathbf{x}^i, .)^{\mathrm{T}}[\boldsymbol{\Sigma}_{\mathrm{M}} + \boldsymbol{\Sigma}_{\varepsilon}]^{-1}(\bar{\mathbf{f}} - \beta \mathbf{1}_k), \tag{8}$$

where $\mathbf{1}_k$ is a $k \times 1$ vector of ones, $\boldsymbol{\Sigma}_{\mathrm{M}}(\mathbf{x}^i, .) = (\mathrm{Cov}[M(\mathbf{x}^i), M(\mathbf{x}^1)], ..., \mathrm{Cov}[M(\mathbf{x}^i), M(\mathbf{x}^k)])^{\mathrm{T}}$, and $\boldsymbol{\Sigma}_{\mathrm{M}}$ is the spatial variance covariance matrix of the k design points (with size $k \times k$). $\boldsymbol{\Sigma}_{\varepsilon}$ is a $k \times k$ variance covariance matrix implied by the sample average simulation noise. Since we don't consider CRN, this matrix is diagonal: $\boldsymbol{\Sigma}_{\varepsilon} = \mathrm{Diag}\{\tau^2(\mathbf{x}^1)/n^1, ..., \tau^2(\mathbf{x}^k)/n^k\}$.

To be able to use Equation (8), β, $\boldsymbol{\Sigma}_{\mathrm{M}}$, and $\boldsymbol{\Sigma}_{\varepsilon}$ need to be estimated (the estimates are denoted by $\hat{\beta}$, $\widehat{\boldsymbol{\Sigma}}_{\mathrm{M}}$, and $\widehat{\boldsymbol{\Sigma}}_{\varepsilon}$). For instance, since the noise variance function $\tau^2(\mathbf{x})$ is usually unknown in practice (Kim and Nelson, 2006), we use $\widehat{\boldsymbol{\Sigma}}_{\varepsilon} = \mathrm{Diag}\{\widehat{\mathrm{Var}}[\bar{f}(\mathbf{x}^1)], ..., \widehat{\mathrm{Var}}[\bar{f}(\mathbf{x}^k)]\}$. See Ankenman et al. (2010) for estimation of the other two parameters.

The mean squared error (MSE) of the Kriging predictor in Equation (8) is also known as the *Kriging*

variance and it is estimated as follows (Chen and Kim, 2014):

$$\widehat{\mathrm{MSE}}(\hat{f}(\mathbf{x}^i)) = s^2(\mathbf{x}^i) = \widehat{\mathbf{\Sigma}}_{\mathrm{M}}(\mathbf{x}^i, \mathbf{x}^i) - \widehat{\mathbf{\Sigma}}_{\mathrm{M}}(\mathbf{x}^i, .)^{\mathrm{T}}[\widehat{\mathbf{\Sigma}}_{\mathrm{M}} + \widehat{\mathbf{\Sigma}}_{\varepsilon}]^{-1}\widehat{\mathbf{\Sigma}}_{\mathrm{M}}(\mathbf{x}^i, .) + \frac{\boldsymbol{\delta}^{\mathrm{T}}\boldsymbol{\delta}}{\mathbf{1}_k^{\mathrm{T}}[\widehat{\mathbf{\Sigma}}_{\mathrm{M}} + \widehat{\mathbf{\Sigma}}_{\varepsilon}]^{-1}\mathbf{1}_k}, \tag{9}$$

where $\boldsymbol{\delta} = 1 - \mathbf{1}_k^{\mathrm{T}}[\widehat{\mathbf{\Sigma}}_{\mathrm{M}} + \widehat{\mathbf{\Sigma}}_{\varepsilon}]^{-1}\widehat{\mathbf{\Sigma}}_{\mathrm{M}}(\mathbf{x}^i, .)$.

Appendix B Relation between $\tau(\mathbf{x})$ and $f(x)$ in different scenarios

For the 3 analytical test functions, we assume a linear relationship between the function value and the standard deviation of noise (i.e., noise magnitude). For best case noise, we have:

$$\begin{aligned}(\max_{\mathbf{x}\in\Theta} f(\mathbf{x}) + b) \times a &= \max_{\mathbf{x}\in\Theta} \tau(\mathbf{x}) = 0.6R_f \text{ or } 6R_f \text{ (light noise or heavy noise)} \\ (\min_{\mathbf{x}\in\Theta} f(\mathbf{x}) + b) \times a &= \min_{\mathbf{x}\in\Theta} \tau(\mathbf{x}) = 0.15R_f \text{ or } 1.5R_f \text{ (light noise or heavy noise)},\end{aligned} \tag{10}$$

where $\tau(\mathbf{x})$ is the standard deviation of the noise on $\bar{f}(\mathbf{x})$ with one replication. From the above equations we can easily obtain a and b. For worst case noise, we have a similar relation:

$$\begin{aligned}(\min_{\mathbf{x}\in\Theta} f(\mathbf{x}) + b) \times a &= \max_{\mathbf{x}\in\Theta} \tau(\mathbf{x}) = 0.6R_f \text{ or } 6R_f \text{ (light noise or heavy noise)} \\ (\max_{\mathbf{x}\in\Theta} f(\mathbf{x}) + b) \times a &= \min_{\mathbf{x}\in\Theta} \tau(\mathbf{x}) = 0.15R_f \text{ or } 1.5R_f \text{ (light noise or heavy noise)}\end{aligned} \tag{11}$$

Table 8: Parameters for the relation between $f(\mathbf{x})$ and $\tau(\mathbf{x})$

		Camel-back	**Rescaled Branin**	**Hartmann-6**
Best case	Low noise	$a = 0.45, b = 3.46$	$a = 0.45, b = 3.05$	$a = 0.45, b = 4.12$
	High noise	$a = 4.5, b = 3.46$	$a = 4.5, b = 3.05$	$a = 4.5, b = 4.12$
Worst case	Low noise	$a = -0.45, b = -8.704$	$a = -0.45, b = -6.95$	$a = -0.45, b = -1.38$
	High noise	$a = -4.5, b = -8.704$	$a = -4.5, b = -6.95$	$a = -4.5, b = -1.38$

Recall that SKO, CKG, and EQI need an estimate of $\tau(\mathbf{x})$: we use $\hat{\tau}(\mathbf{x}) = (\hat{f}(\mathbf{x}) + b) \times a$, as the goal function $f(\mathbf{x})$ is assumed to be unknown.

Appendix C Results for Branin function

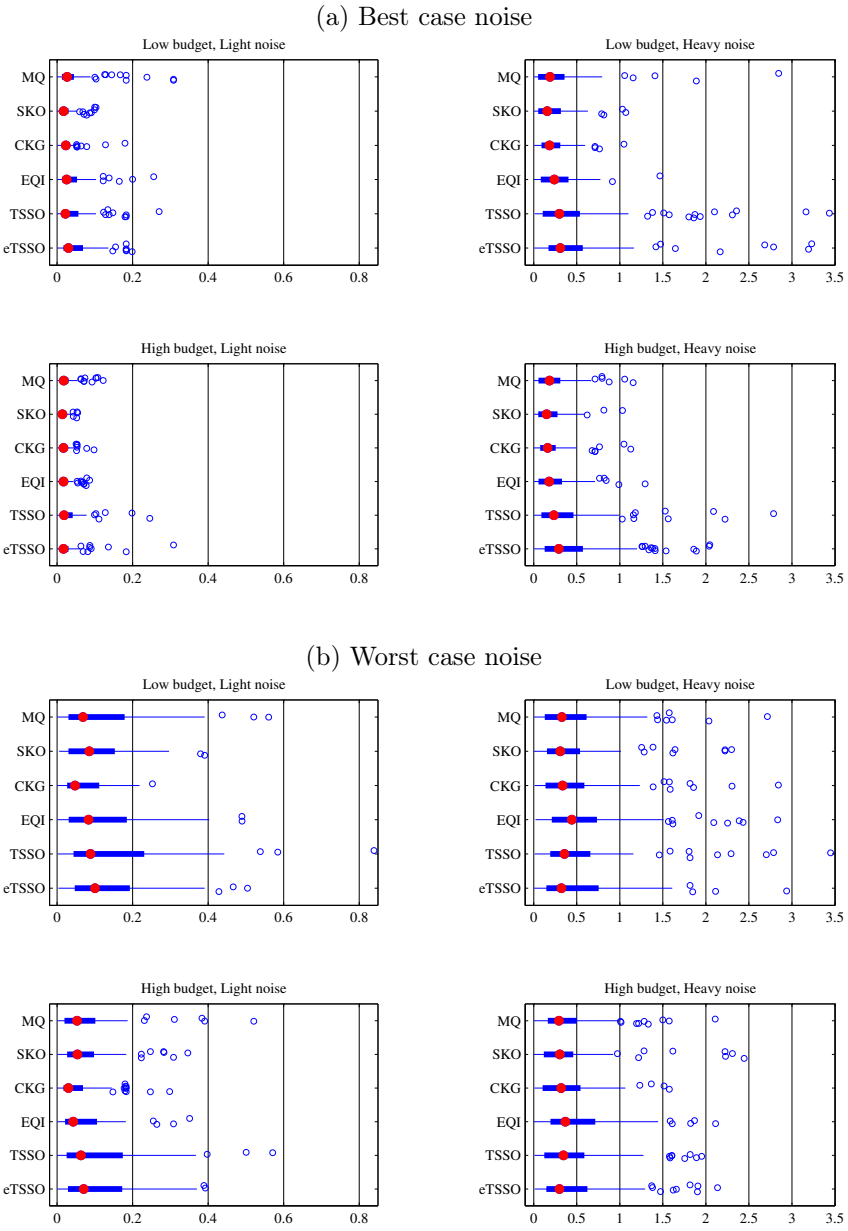


Figure C.1: GAP for the Branin function ($R_f = 6$). At each scenario, the boxplots show the distribution of GAP for each method. The median is indicated in red; the edges of the boxes are the 25th and 75th percentiles. The results are analogous to the camel-back function: CKG and SKO perform somewhat better than the benchmark method (MQ) while other algorithms don't. TSSO and eTSSO usually have the largest GAP.

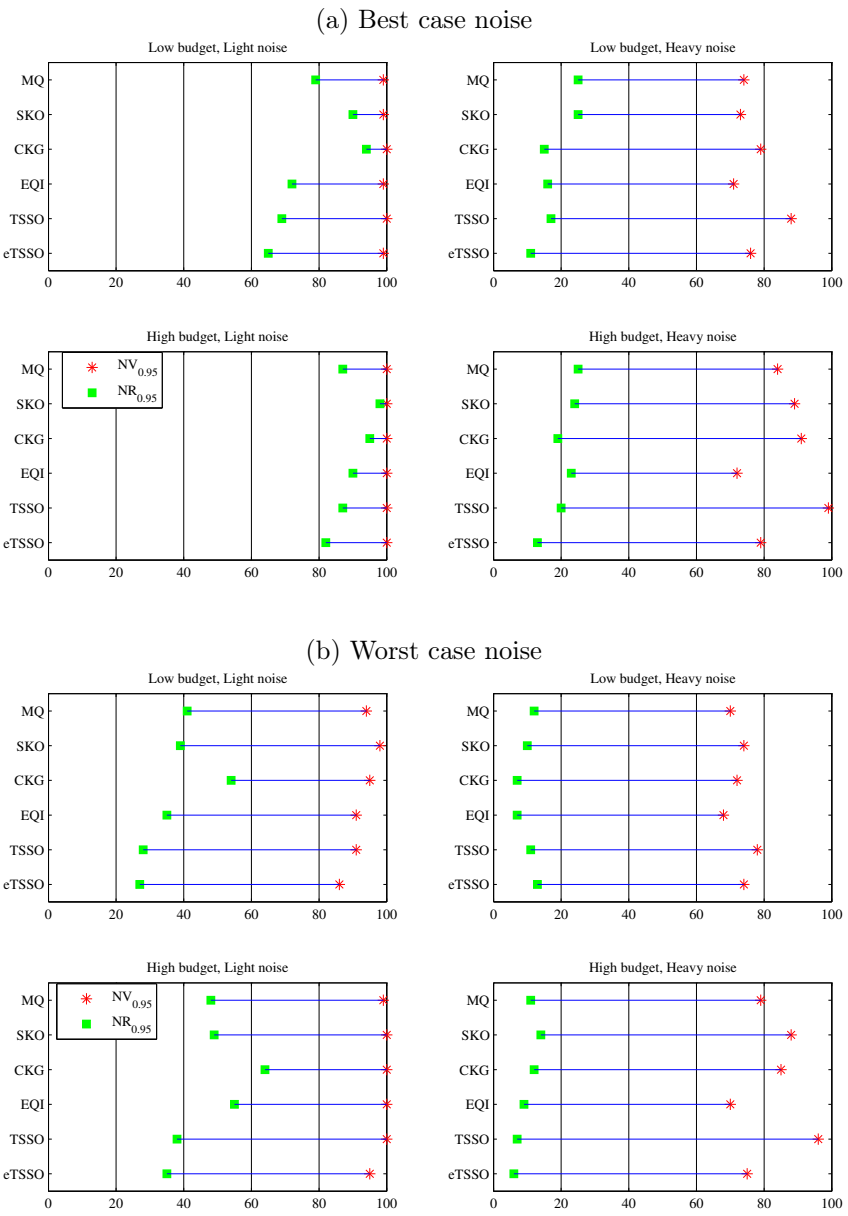
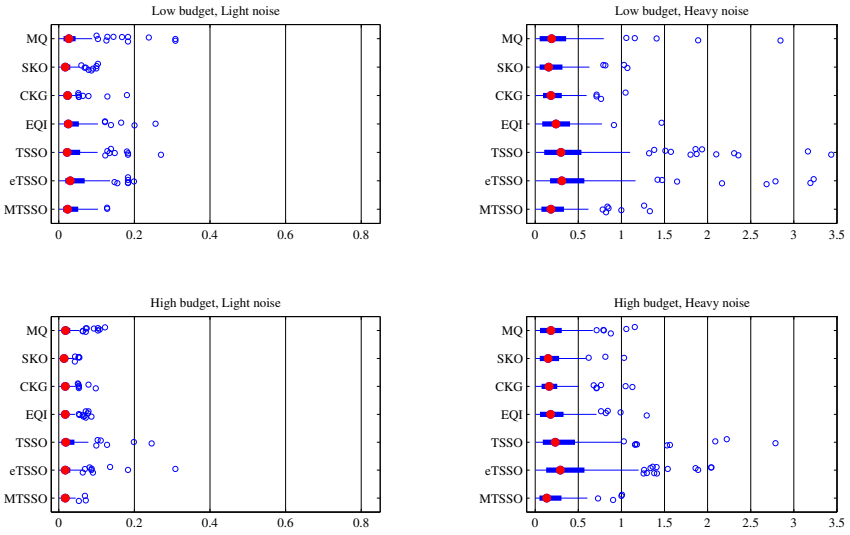


Figure C.2: $NV_{0.95}$ (indicated in red) and $NR_{0.95}$ (indicated in green) for the Branin function. As with camel-back, TSSO has the highest $NV_{0.95}$. MQ is again very competitive; EQI and eTSSO usually provide the lowest $NV_{0.95}$. In all scenarios, the performance of all algorithms suffers from the identification problem.

(a) Best case noise



(b) Worst case noise

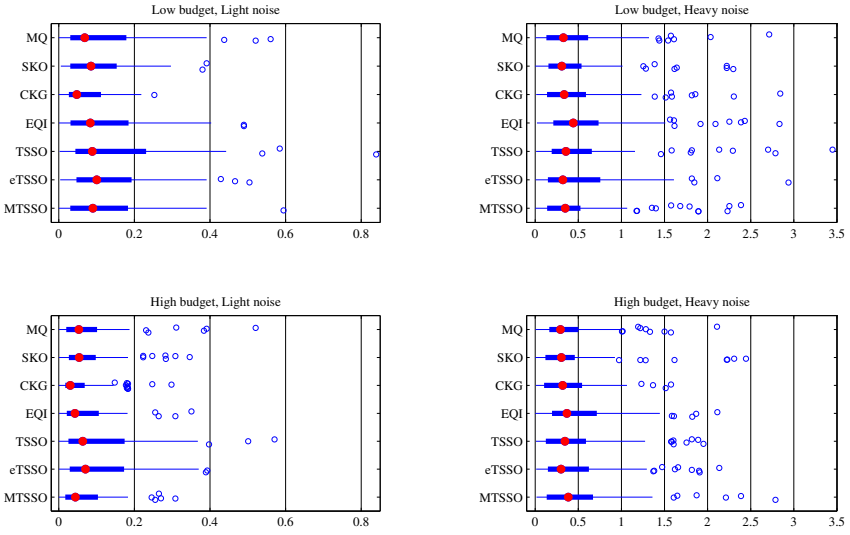
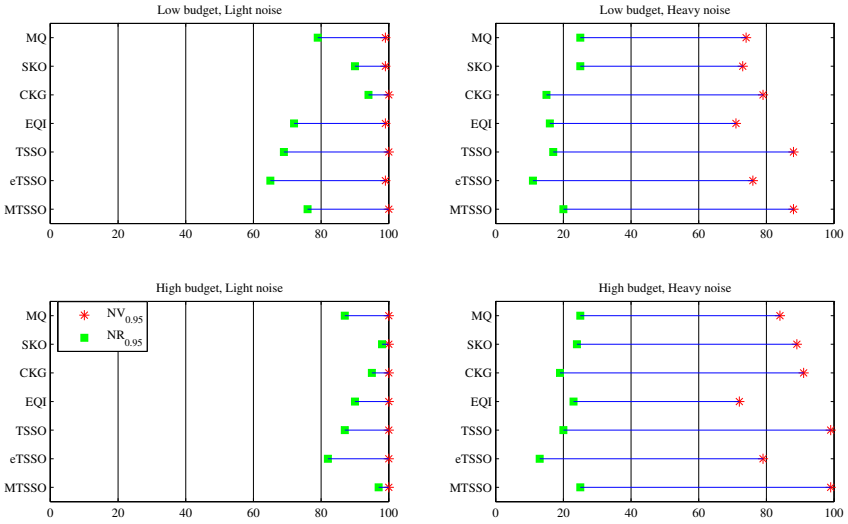


Figure C.3: GAP for the Branin function with MTSSO included ($R_f = 6$). At each scenario, the boxplots show the distribution of GAP for each method. The median is indicated in red; the edges of the boxes are the 25th and 75th percentiles. MTSSO usually provides smaller GAP than TSSO.

(a) Best case noise



(b) Worst case noise

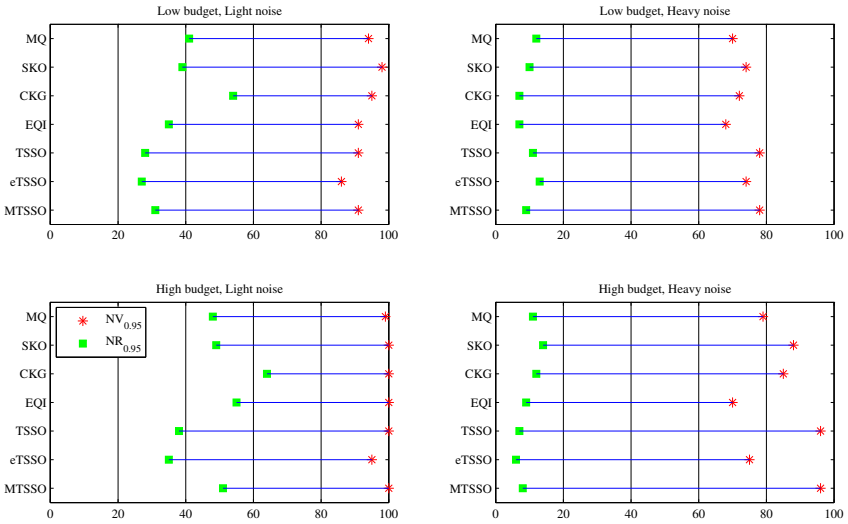


Figure C.4: $NV_{0.95}$ (indicated in red) and $NR_{0.95}$ (indicated in green) for the Branin function with MTSSO included. Note that MTSSO outperforms TSSO in terms of $NR_{0.95}$

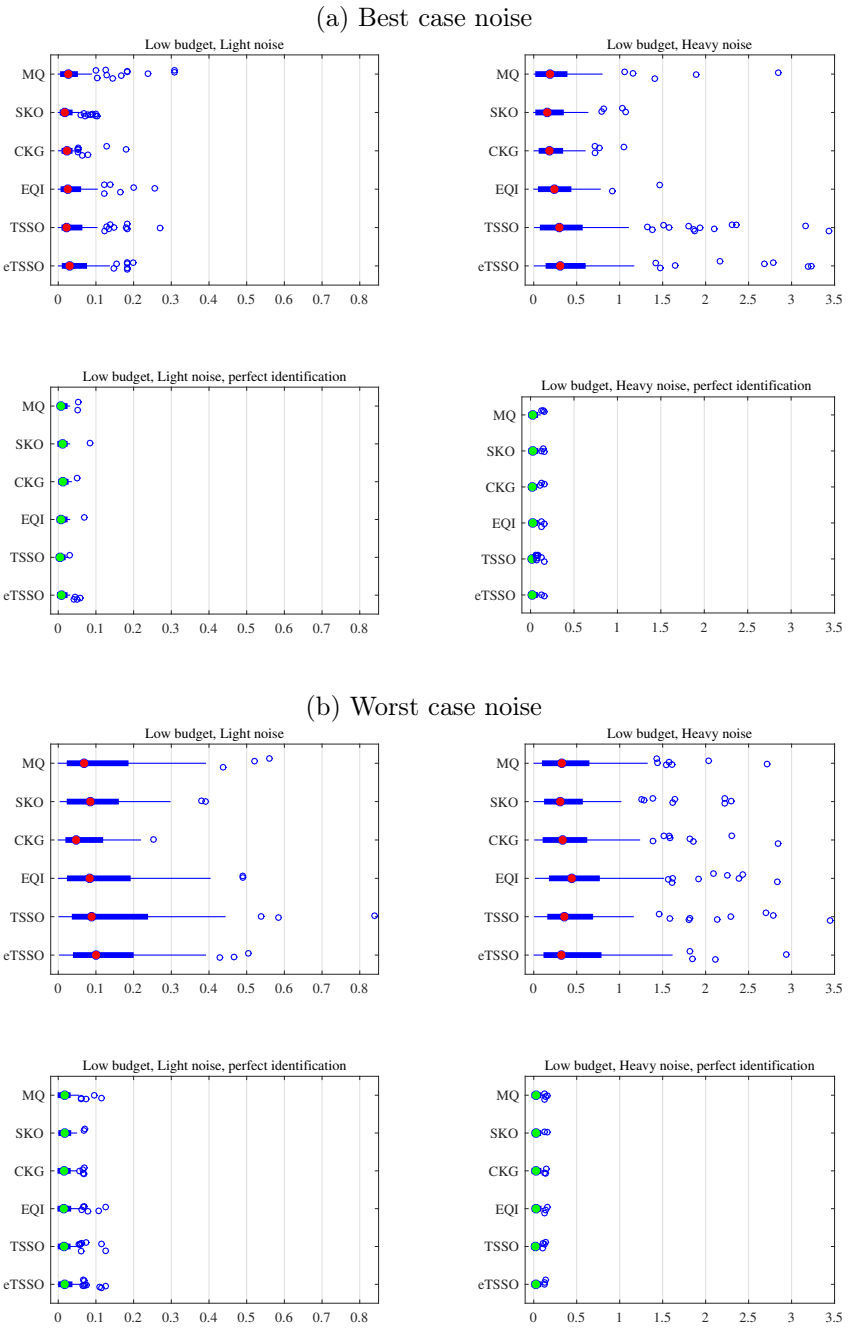
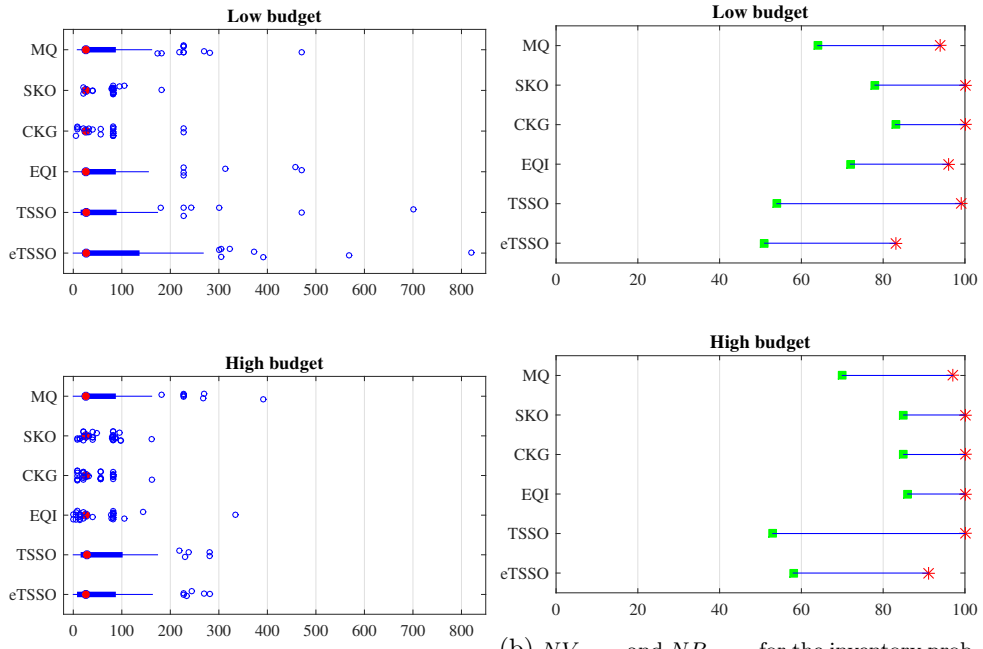


Figure C.5: Comparing previous GAP results of Branin ($R_f = 6$) with GAP results using perfect identification, for all algorithms. At each scenario, the boxplots show the distribution of GAP for each method. The median of the previous GAP is indicated in red; the median of the new GAP (with perfect identification) is indicated in green. The edges of the boxes are the 25th and 75th percentiles.

Appendix D Results for the (s, S) inventory problem



(a) GAP for the inventory problem

(b) $NV_{0.999}$ and $NR_{0.999}$ for the inventory problem

Figure D.1: GAP and $NV_{0.999}$ and $NR_{0.999}$ for the (s, S) inventory problem ($R_f = 8584$). In the GAP figure, the boxplots show the distribution of GAP for each method. The median is indicated in red; the edges of the boxes are the 25th and 75th percentiles. The results are analogous to the camel-back function: CKG and SKO perform somewhat better than the benchmark method (MQ) while other algorithms usually don't (EQI gives better results only with high budget). TSSO and eTSSO usually have the largest GAP and eTSSO performs the worst in terms of $NV_{0.999}$. Both with high and low budget, the performance of all algorithms suffers from the identification problem (especially TSSO).

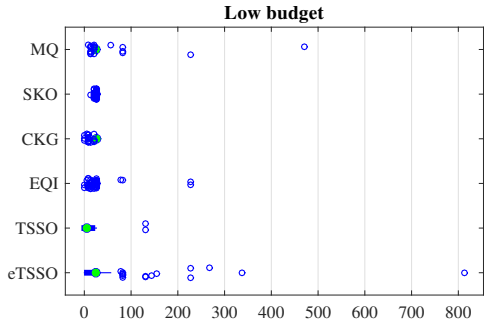


Figure D.2: GAP results using perfect identification, for all algorithms. At each scenario, the boxplots show the distribution of GAP for each method. The median is indicated in green. The edges of the boxes are the 25th and 75th percentiles.

Appendix E MTSSO results for camel-back function

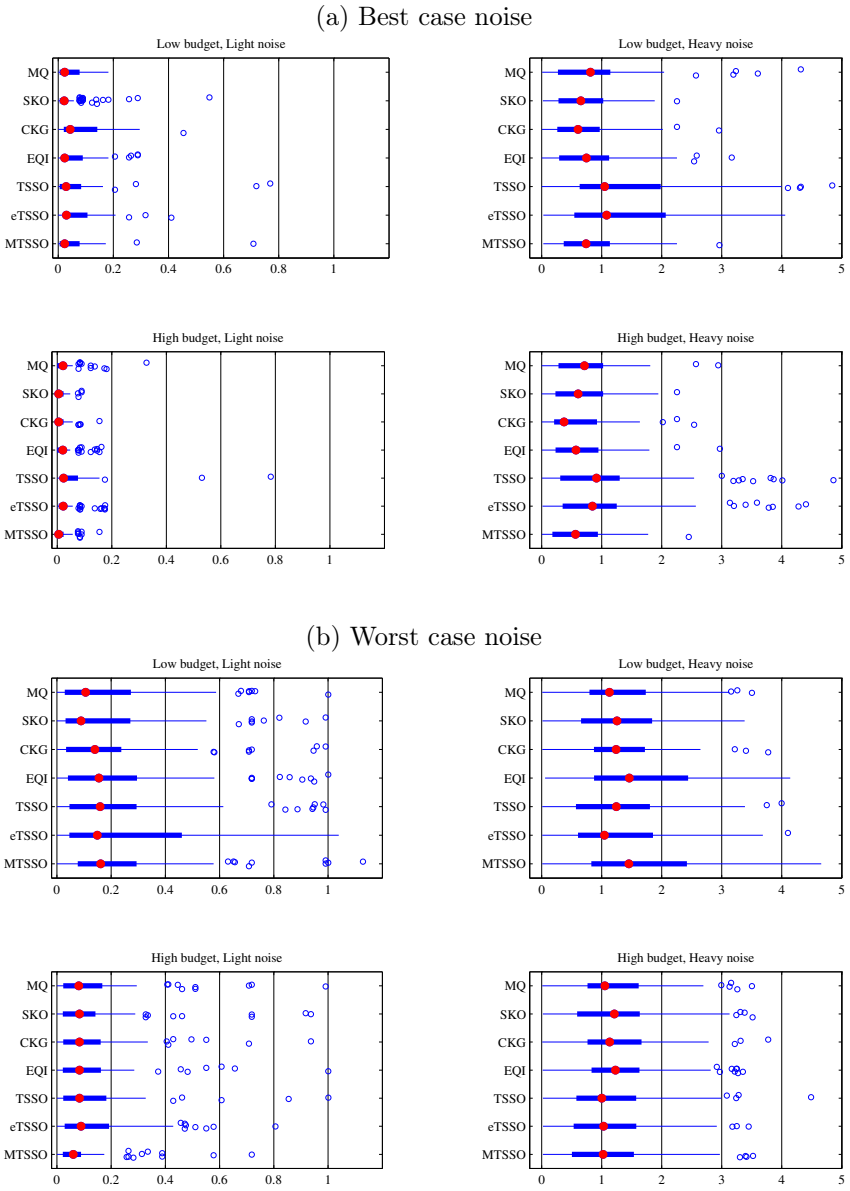


Figure E.1: GAP for the camel-back function with MTSSO included ($R_f = 7.3$). At each scenario, the boxplots show the distribution of GAP for each method. The median is indicated in red; the edges of the boxes are the 25th and 75th percentiles.

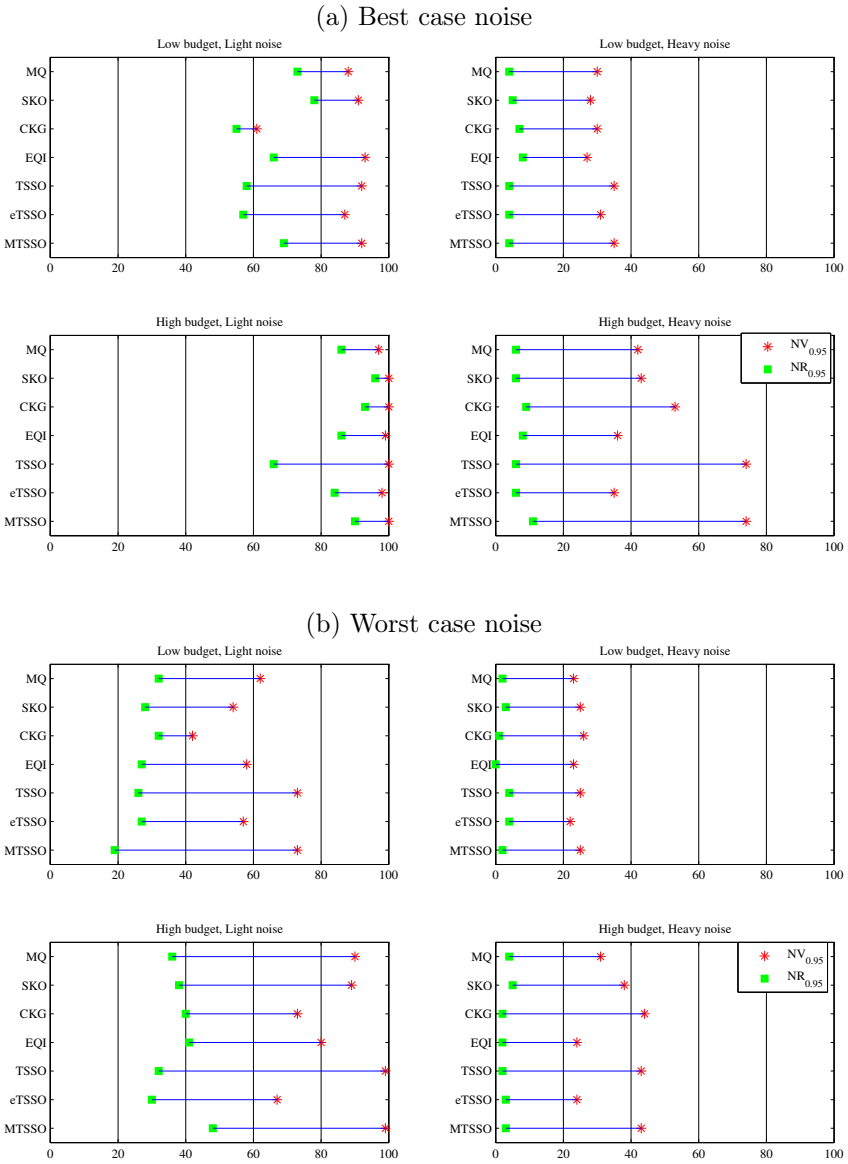


Figure E.2: $NV_{0.95}$ (indicated in red) and $NR_{0.95}$ (indicated in green) for the camel-back function with MTSSO included. Note that $NV_{0.95}$ of TSSO is the same as MTSSO.

Appendix F Number of distinct points sampled

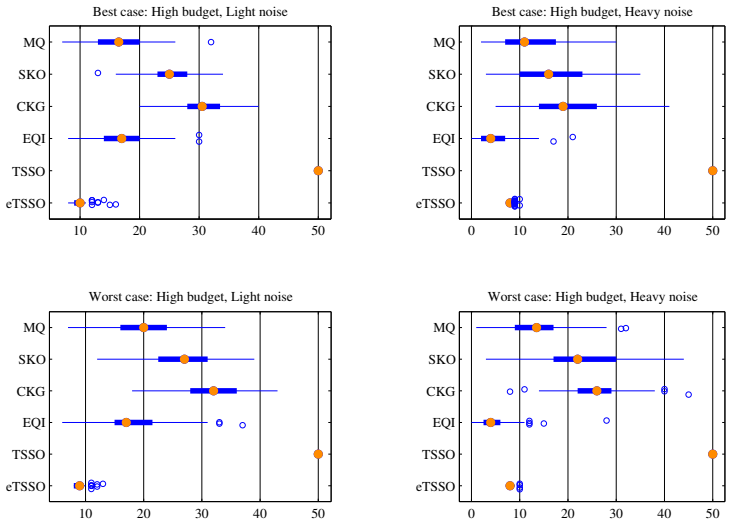


Figure F.1: Number of distinct points sampled for the Branin function (High budget). At each scenario, the boxplots show the distribution of the number of distinct points sampled by each method. The median is indicated in orange; the edges of the boxes are the 25th and 75th percentiles.

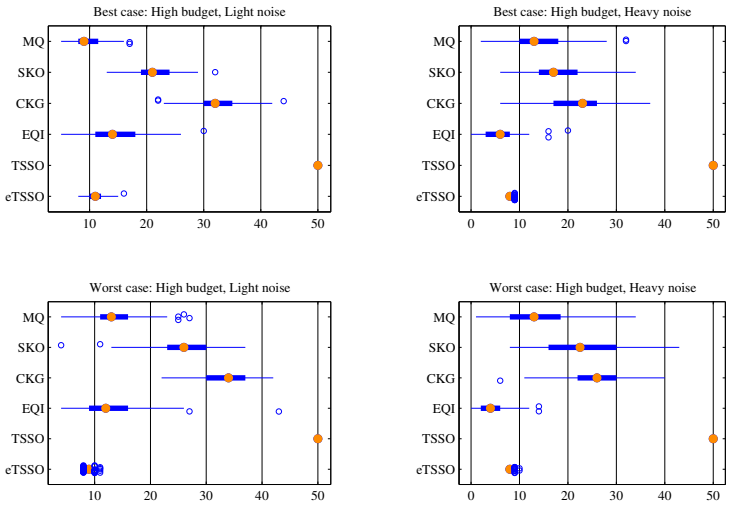


Figure F.2: Number of distinct points sampled for the camel-back function (High budget). At each scenario, the boxplots show the distribution of the number of distinct points sampled by each method. The median is indicated in orange; the edges of the boxes are the 25th and 75th percentiles.

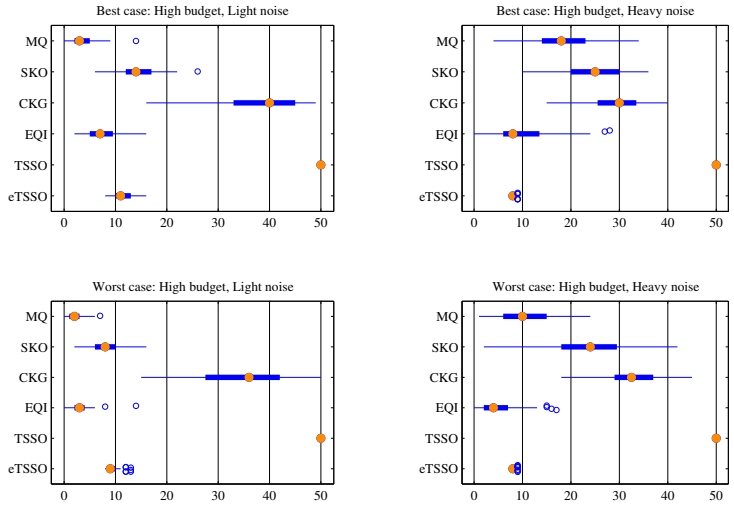


Figure F.3: Number of distinct points sampled for the Hartmann-6 function (High budget). At each scenario, the boxplots show the distribution of the number of distinct points sampled by each method. The median is indicated in orange; the edges of the boxes are the 25th and 75th percentiles.

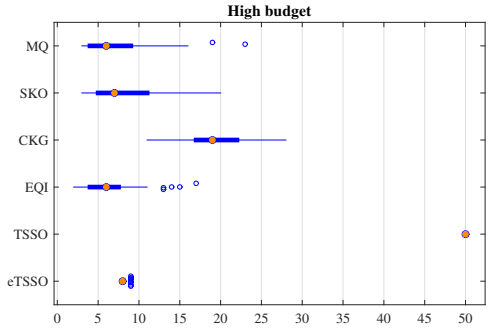


Figure F.4: Number of distinct points sampled for the (s, S) inventory problem (High budget). At each scenario, the boxplots show the distribution of the number of distinct points sampled by each method. The median is indicated in orange; the edges of the boxes are the 25th and 75th percentiles.

References

Ankenman, B., Nelson, B. L., and Staum, J. (2010). Stochastic kriging for simulation metamodeling. *Operations Research*, 58:371–382.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256.

Boesel, J., Nelson, B. L., and Kim, S.-H. (2003). Using ranking and selection to "clean up" after simulation optimization. *Operations Research*, 51(5):814–825.

Brochu, E., Cora, V. M., and De Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.

Chen, C.-H., Lin, J., Yücesan, E., and Chick, S. E. (2000). Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems*, 10(3):251–270.

Chen, X., Ankenman, B. E., and Nelson, B. L. (2012). The effects of common random numbers on stochastic kriging metamodels. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 22(2):7.

Chen, X. and Kim, K.-K. (2014). Stochastic kriging with biased sample estimates. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 24(2):8.

Cressie, N. (1993). *Statistics for spatial data*. John Wiley & Sons, New York.

Dixon, L. C. W. and Szegö, G. P. (1978). *Towards global optimisation 2*. North-Holland Amsterdam, The Netherlands.

Forrester, A., Sobester, A., and Keane, A. (2008). *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, Chichester.

Frazier, P., Powell, W., and Dayanik, S. (2009). The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, 21(4):599–613.

Fu, M. C. and Healy, K. J. (1997). Techniques for optimization via simulation: an experimental study on an (s, s) inventory system. *IIE transactions*, 29(3):191–199.

Hong, L. J. and Nelson, B. L. (2007). Selecting the best system when systems are revealed sequentially. *IIE Transactions*, 39(7):723–734.

Hong, L. J., Nelson, B. L., and Xu, J. (2015). Discrete optimization via simulation. In *Handbook of Simulation Optimization*, pages 9–44. Springer.

Huang, D., Allen, T. T., Notz, W. I., and Zeng, N. (2006). Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, 34:441–466.

Jalali, H. and Van Nieuwenhuyse, I. (2015). Simulation optimization in inventory replenishment: a classification. *IIE Transactions*, 47:1–19.

Jones, D. R. (2001). A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492.

Kim, S.-H. and Nelson, B. L. (2006). Selecting the best system. *Handbooks in operations research and management science*, 13:501–534.

Kleijnen, J. P. (2009). Kriging metamodeling in simulation: A review. *European Journal of Operational Research*, 192:707–716.

Kleijnen, J. P. (2014). Simulation-optimization via kriging and bootstrapping: a survey. *Journal of Simulation*, 8(4):241–250.

Kleijnen, J. P. and Van Beers, W. C. (2005). Robustness of kriging when interpolating in random simulation with heterogeneous variances: some experiments. *European Journal of Operational Research*, 165(3):826–834.

Law, A. M. (2015). *Simulation Modeling and Analysis*. McGraw-Hill,New York.

Lemieux, C. (2009). *Monte carlo and quasi-monte carlo sampling*. Springer Science & Business Media.

Liu, C., Pedrielli, G., and Ng, S. H. (2014). etsso: Adaptive search method for stochastic global optimization under finite budget. *Working paper*.

McCormack, R. and Coates, G. (2015). A simulation model to enable the optimization of ambulance fleet allocation and base station location for increased patient survival. *European Journal of Operational Research*.

Naoum-Sawaya, J., Ghaddar, B., Arandia, E., and Eck, B. (2015). Simulation-optimization approaches for water pump scheduling and pipe replacement problems. *European Journal of Operational Research*.

Picheny, V. and Ginsbourger, D. (2014). Noisy kriging-based optimization methods: A unified implementation within the DiceOptim package. *Computational Statistics & Data Analysis*, 71:1035–1053.

Picheny, V., Ginsbourger, D., Richet, Y., and Caplin, G. (2013a). Quantile-based optimization of noisy computer experiments with tunable precision. *Technometrics*, 55:2–13.

Picheny, V., Wagner, T., and Ginsbourger, D. (2013b). A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization*, 48:607–626.

Preuss, M., Wagner, T., and Ginsbourger, D. (2012). High-dimensional model-based optimization based on noisy evaluations of computer games. In *LION*, pages 145–159. Springer.

Quan, N., Yin, J., Ng, S. H., and Lee, L. H. (2013). Simulation optimization via kriging: a sequential search using expected improvement with computing budget constraints. *IIE Transactions*, 45:763–780.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT press, Cambridge, MA.

Saif, A. and Elhedhli, S. (2015). Cold supply chain design with environmental considerations: A simulation-optimization approach. *European Journal of Operational Research*.

Scott, W., Frazier, P., and Powell, W. (2011). The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression. *SIAM Journal on Optimization*, 21(3):996–1026.

Xu, J. (2012). Efficient discrete optimization via simulation using stochastic kriging. In *Proceedings of the 2012 Winter Simulation Conference (WSC)*, pages 1–12. IEEE.

Yin, J., Ng, S., and Ng, K. (2011). Kriging metamodel with modified nugget-effect: The heteroscedastic variance case. *Computers & Industrial Engineering*, 61:760–777.