A multiobjective stochastic simulation optimization algorithm

Sebastian Rojas Gonzalez[a,*], Hamed Jalali[b], Inneke Van Nieuwenhuyse[a,c]

[a] *Department of Decision Sciences and Information Management, KU Leuven, Belgium.*
[b] *Department of Information Systems, Supply Chain and Decision Making, Neoma Business School, 1 Rue du Maréchal Juin, Mont-Saint-Aignan, 76825, France*
[c] *Quantitative Methods, Research Group Logistics, UHasselt, Belgium*

Abstract

The use of kriging metamodels in simulation optimization has become increasingly popular during recent years. The majority of the algorithms so far uses the ordinary (deterministic) kriging approach for constructing the metamodel, assuming that solutions have been sampled with infinite precision. This is a major issue when the simulation problem is stochastic: ignoring the noise in the outcomes may not only lead to an inaccurate metamodel, but also to potential errors in identifying the optimal points among those sampled. Moreover, most algorithms so far have focused on single-objective problems. In this article, we test the performance of a multiobjective simulation optimization algorithm that contains two crucial elements: the search phase implements stochastic kriging to account for the inherent noise in the outputs when constructing the metamodel, and the accuracy phase uses a well-known multiobjective ranking and selection procedure in view of maximizing the probability of selecting the true Pareto-optimal points by allocating extra replications on competitive designs. We evaluate the impact of these elements on the search and identification effectiveness, for a set of test functions with different Pareto front geometries, and varying levels of heterogeneous noise. Our results show that the use of stochastic kriging is essential in improving the search efficiency; yet, the allocation procedure appears to lose effectiveness in settings with high noise. This emphasizes the need for further research on multiobjective ranking and selection methods.

Keywords: Simulation, Stochastic processes, Multiobjective simulation optimization, Kriging

1. Introduction

Simulation optimization refers to optimization using (stochastic) simulation (Chen & Lee, 2010); the simulation model acts as a *black box* that numerically evaluates the objective for any given decision vector, and the goal is to find the values of the decision vector(s) that optimize the objective of interest. Yet, in most settings, the analyst has limited computational budget: the challenge thus is to develop a strategy that searches

*Corresponding author
Email address: sebastian.rojasgonzalez@kuleuven.be (Sebastian Rojas Gonzalez)

the solution space in a way that is both effective (i.e., it has high probability of finding the optimal decision vectors), and efficient (it does so with limited computational budget). This is a significant challenge, especially in settings where the search space is vast, and/or the simulation itself is computationally expensive. Consequently, different approaches have been developed to provide inexpensive *metamodels* (also referred to as *surrogate models*) that seek to provide predictions of the objective, using the information obtained from a limited number of sampled points. Evidently, in order for such metamodels to be useful, they have to be accurate and sufficiently cheap to evaluate.

The use of *kriging* (Sacks et al., 1989; Van Beers & Kleijnen, 2003) metamodels has been increasingly popular in modeling and optimizing deterministic computer experiments. They are commonly used in engineering (see e.g., Forrester et al. (2008); Wang & Shan (2007)), and machine learning (see e.g., Rasmussen & Williams (2005); Koch et al. (2015); Zuluaga et al. (2016)); in the latter case, they are often referred to as Gaussian Process Regression (GPR) or Gaussian random field metamodels. Recently, *ordinary* kriging is also increasingly popular in the Operations Research and Management Science fields (see e.g., Kleijnen (2015); Fu (2015)), as it can successfully approximate outputs over the entire search space (yielding a *global metamodel*), while also providing a quantification of the prediction uncertainty through the mean square error (MSE), also known as *kriging variance*. In general, kriging-based optimization methods perform a *sequential* procedure, as shown in Algorithm 1:

Algorithm 1 Generic procedure of kriging-based optimization algorithms

1: Sample initial design (using e.g., latin hypercube sampling)
2: Evaluate response(s) using the expensive simulator
3: **while** Stopping criterion not met **do**
4: Fit kriging metamodel to the observed simulation response(s)
5: Search for new infill point(s) using an infill criterion that exploits the kriging information
6: Evaluate the infill point(s) using the expensive simulator
7: **end while**
8: Identify the optimal point(s)

During the search phase, an *infill criterion*, also referred to as *improvement function* (Wagner et al., 2010) or *acquisition function* (Rasmussen & Williams, 2005), directs the search towards interesting solutions by exploiting the kriging information (i.e., the prediction and its uncertainty). The metamodel is sequentially updated by simulating the selected *infill point(s)* in the search space, until the computational budget is depleted or a desired performance level is reached. The well-known Efficient Global Optimization (EGO) algorithm (Jones et al., 1998), developed for deterministic and single-objective optimization problems, introduced the *Expected Improvement* (EI) criterion; since then, EI has been used in numerous kriging-based single- and multiobjective algorithms for deterministic problems (see e.g., Knowles (2006); Zhang et al. (2010)).

However, ordinary kriging (hereafter referred to as deterministic kriging) models and

their related algorithms and infill criteria (such as EGO, resp. EI) are ill-suited for stochastic simulation settings, as they ignore the noise in the observed objective outcomes, assuming they were sampled with infinite precision (Quan et al., 2013; Jalali et al., 2017). Given that analysts typically have a limited simulation budget, stochastic settings always contain a trade-off between the number of distinct points that can be sampled during the search, and the accuracy obtained in the sampling process. Ignoring the noise in the outcomes may obviously lead to an inaccurate metamodel (and, thus, mislead the search for interesting new points to sample: see step 5 in Algorithm 1). Moreover, it also leads to potential errors in identifying the solutions with the best expected performance among the points sampled (step 6 in Algorithm 1). Current *multiobjective* optimization algorithms simply identify these based on the observed mean outcomes of the sampled points, again ignoring the noise. This may lead to two types of error due to sampling variability (Lee et al., 2010): 1) a *Type I* error occurs when designs that actually belong to the Pareto set, are observed as dominated; 2) a *Type II* error occurs when designs that are actually dominated, fall into the observed Pareto set. Evidently, we are interested in identifying the points with the best *true* performance. For single-objective settings, multiple *ranking and selection* (R&S) procedures have been developed (Boesel et al., 2003; Chen & Lee, 2010) to aid in the correct selection of the optimal designs, by smartly allocating extra replications to critically competitive designs, without wasting budget on designs that are clearly suboptimal. *Multiobjective ranking and selection* (MORS) methods, however, have received relatively little attention in the literature so far (Hunter et al., 2019).

In this paper, we propose an approach for the optimization of stochastic discrete-event simulation problems with *multiple objectives*. Evidently, the goal of the algorithm is to find the true *Pareto set*, i.e., the set of *non-dominated* solutions: these are the decision vectors for which none of the objectives can be improved without deteriorating the outcome for another objective. The evaluation of these solutions in the objective space corresponds to the *Pareto front*. The current literature on algorithms tackling such multiobjective stochastic problems is really scarce (Rojas-Gonzalez & Van Nieuwenhuyse, 2019). To guide the search, these algorithms rely on deterministic kriging (i.e., they fit an ordinary kriging metamodel to the observed sample means of each objective), or kriging with a (fixed) nugget effect, which models the effect of white noise in the observations, under the assumption that the variance of the noise is homogeneous (Gramacy & Lee, 2012).

The algorithm evaluated in this paper is referred to as SK-MOCBA, and has the following characteristics:

1. In the search phase, it uses *stochastic kriging metamodeling* (Ankenman et al., 2010), to take account of the noise when selecting new points to sample. The stochastic kriging information is reflected in the Modified Expected Improvement (MEI) infill criterion (Quan et al., 2013). Consequently, as opposed to the multiobjective algorithms of Koch et al. (2015), Zuluaga et al. (2016) and Horn et al. (2017), our algorithm is able to handle *heterogenous* noise, i.e., noise that varies in strength

over the search space. This is the most prevalent type of noise in practical settings (Kleijnen & Van Beers, 2005; Kim & Nelson, 2006).

2. It integrates a MORS method in the identification phase, in view of maximizing the probability of selecting the *true* Pareto-optimal points. We opt for the Multiobjective Optimal Computing Budget Allocation (MOCBA) procedure (Lee et al., 2010; Chen & Lee, 2010), a well-known Bayesian allocation framework.

To the best of our knowledge, this is the first stochastic multiobjective algorithm to implement these aspects. A basic version of the algorithm was published in Rojas-Gonzalez et al. (2018); yet, in this paper, we discuss the full improved version of the proposed method in detail, and test it using standard benchmark functions for multiobjective optimizers, exhibiting different Pareto front geometries (Huband et al., 2006) and different levels of noise. We evaluate the performance (both in objective space and decision space) using quality indicators that are commonly used in the literature (Zitzler et al., 2008), and explicitly evaluate the power of the stochastic kriging element (combined with the MEI criterion) in improving the search efficiency of the algorithm, by comparing with its deterministic counterpart. We also explicitly evaluate the contribution of the MORS procedure in correctly identifying the true Pareto-optimal points, by comparing results with and without MOCBA.

The remainder of this article is structured as follows: In Section 2 we provide the basic theory of stochastic kriging metamodeling. Section 3 describes the proposed algorithm in detail. Section 4 explains the numerical experiments designed to test the performance of the proposed algorithm, while Section 5 discusses the results. We conclude the article in Section 6, and identify some promising directions for further research.

2. Stochastic kriging

Stochastic kriging (SK) is a recently developed metamodeling technique for representing the response surface implied by a stochastic simulation (Ankenman et al., 2010). For a given objective and an arbitrary design point $\mathbf{x}^i$, the model represents the observed objective value $\tilde{f}_r(\mathbf{x}^i)$ in the r^{th} replication as:

$$\tilde{f}_r(\mathbf{x}^i) = \mathbf{f}(\mathbf{x}^i)^T\boldsymbol{\beta} + M(\mathbf{x}^i) + \epsilon_r(\mathbf{x}^i) \tag{1}$$

In this expression, $\mathbf{f}(\mathbf{x}^i)$ is a vector of known functions of $\mathbf{x}^i$, with $\boldsymbol{\beta}$ a vector of unknown parameters of compatible dimension. $M(\mathbf{x}^i)$ is a realization of a mean 0 covariance stationary Gaussian random field at the design point $\mathbf{x}^i$. It is assumed that this field exhibits spatial correlation: i.e., $M(\mathbf{x}^i)$ and $M(\mathbf{x}^h)$ will tend to be similar when $\mathbf{x}^i$ is close to $\mathbf{x}^h$ in the design space. This assumption is analogous to the assumption made in the traditional deterministic kriging metamodel (Van Beers & Kleijnen, 2003; Forrester et al., 2008; Kleijnen, 2015); essentially, this type of uncertainty is imposed on the problem to aid in developing the metamodel. Hence, it is referred to as *extrinsic uncertainty*. Different

spatial correlation functions (also referred to as *kernels*) exist; the most popular ones in the kriging literature are the Gaussian and Matérn kernels (Rasmussen & Williams, 2005).

The *intrinsic uncertainty* $\epsilon_r(\mathbf{x}^i)$ is, naturally, independent and identically distributed across replications, having mean 0 and variance $\tau^2(\mathbf{x}^i)$ at any arbitrary point $\mathbf{x}^i$. Note that the model allows for heterogenous noise, implying $\tau^2(\mathbf{x}^i)$ need not be constant throughout the design space. The model also allows for $Corr[\epsilon_r(\mathbf{x}^i), \epsilon_r(\mathbf{x}^h)] > 0$, as tends to be the case with the use of common random numbers (CRN); yet, this is not desirable, as discussed in Chen et al. (2012). In what follows, we only consider the case where the first term in Expression 1 is a constant, β_0, representing the overall mean of the response surface, as this has been shown to be the most useful model in practice (Santner et al., 2013; Kleijnen, 2015).

The stochastic kriging prediction $\hat{f}(\mathbf{x}^i)$ at *any* design point $\mathbf{x}^i$ (whether it has been sampled or not) is given by

$$\hat{f}(\mathbf{x}^i) = \beta_0 + \Sigma_M(\mathbf{x}^i, \cdot)^T[\Sigma_M + \Sigma_\epsilon]^{-1}(\overline{\mathbf{f}} - \beta_0 \mathbf{1}_p) \tag{2}$$

This expression is analogous to the kriging predictor in the well-known ordinary (deterministic) kriging model, except for the impact of the intrinsic noise, present through Σ_ϵ. Σ_ϵ is the $p \times p$ covariance matrix with (i,h) element $cov\left[\sum_{j=1}^{r_i} \epsilon_j(\mathbf{x}^i)/r^i, \sum_{j=1}^{r_h} \epsilon_j(\mathbf{x}^h)/r^h\right]$ accross all design points $\mathbf{x}^h$ and $\mathbf{x}^i$; when CRN are not used, this reduces to the diagonal matrix $\text{diag}[\tau^2(\mathbf{x}^1)/r^1, ..., \tau^2(\mathbf{x}^p)/r^p]$. The notation $\overline{\mathbf{f}}$ is the vector containing all the observed mean outcomes at the already sampled design points: $\overline{\mathbf{f}} = [\bar{f}(\mathbf{x}^1), ..., \bar{f}(\mathbf{x}^p)]^T$, with $\bar{f}(\mathbf{x}^i) = \sum_{k=1}^{r^i} \tilde{f}_k(\mathbf{x}^i)/r^i$, and $\mathbf{1}_p$ is a $p \times 1$ vector of ones. Analogous to the ordinary kriging model, Σ_M denotes the $p \times p$ matrix containing the covariances between each couple of already sampled points, as implied by the extrinsic spatial correlation model: $\Sigma_M(\mathbf{x}^i, \mathbf{x}^j) = Cov[M(\mathbf{x}^i), M(\mathbf{x}^j)]$. The notation $\Sigma_M(\mathbf{x}^i, \cdot)$ is the $p \times 1$ vector containing the covariances between the point under study, and the p already sampled points: $\Sigma_M(\mathbf{x}^i, \cdot) = [Cov[M(\mathbf{x}^i), M(\mathbf{x}^1)], Cov[M(\mathbf{x}^i), M(\mathbf{x}^2)], ..., Cov[M(\mathbf{x}^i), M(\mathbf{x}^p)]^T$. As opposed to the ordinary kriging predictor, the stochastic kriging predictor no longer interpolates, due to the presence of the intrinsic noise.

The MSE of the stochastic kriging predictor (i.e., the stochastic kriging variance), denoted $s^2(\mathbf{x}^i)$, is given by (Ankenman et al., 2010; Chen & Kim, 2014):

$$s^2(\mathbf{x}^i) = \Sigma_M(\mathbf{x}^i, \mathbf{x}^i) - \Sigma_M(\mathbf{x}^i, \cdot)^T[\Sigma_M + \Sigma_\epsilon]^{-1}\Sigma_M(\mathbf{x}^i, \cdot) + \frac{\gamma^T\gamma}{\mathbf{1}_p^T[\Sigma_M + \Sigma_\epsilon]^{-1}\mathbf{1}_p} \tag{3}$$
$$\text{with } \gamma = 1 - \mathbf{1}_p^T[\Sigma_M + \Sigma_\epsilon]^{-1}\Sigma_M(\mathbf{x}^i, \cdot)$$

Again, the difference with the ordinary kriging expressions lies in the impact of the intrinsic noise, through Σ_ϵ; in the absence of intrinsic noise, expressions (2) and (3) thus reduce to the ordinary kriging predictor, and its variance. The presence of intrinsic noise *inflates* the MSE, as discussed in Ankenman et al. (2010). The above expressions suppose that β_0, Σ_ϵ and Σ_M are known; clearly, in a realistic application, they must be estimated. This

is commonly done using maximum likelihood estimation (MLE); yielding $\hat{\beta}_0$, $\hat{\Sigma}_\epsilon$ and $\hat{\Sigma}_M$. We refer to Ankenman et al. (2010) for the detailed derivation of these MLE estimators. These estimators are then commonly used in expressions (2) and (3) to yield the estimated kriging predictor $\hat{f}$, and its variance $\hat{s}^2$.

3. Proposed algorithm

This section discusses the proposed algorithm, which consists of 3 phases. Section 3.1 discusses the distribution of the computational budget across these phases, while Section 3.2 gives the step-by-step outline of the procedure.

3.1. Distribution of the replication budget

The algorithm consists of 3 phases: the initial design phase, exploratory phase and accuracy phase. It requires an upfront choice regarding the distribution of the total replication budget, denoted T, across these three phases (see Table 1). Let n_0 denote the number of points to sample in the initial design phase, with B the (fixed) number of replications per initial design point, and let N denote the number of infill points one wishes to sample in the exploratory phase. The total replication budget is then set to $T = B(n_0 + N)$ replications; the part of the budget spent in the initial design phase obviously equals $T_{init} = Bn_0$ replications. Yet, in the exploratory phase, a reduced number of replications $b < B$ is used per infill point, thus $T_{exp} = bN$ replications. This allows to save budget for the accuracy phase (i.e., $T_{acc} = N(B - b)$) replications; this remaining budget will be allocated only to competitive designs, according to the MOCBA procedure.

Table 1: Overview of the distribution of the replication budget across the 3 phases of the algorithm

Total budget: $T = B(n_0 + N)$		
Initial design phase	Exploratory phase	Accuracy phase
$T_{init} = Bn_0$ n_0: number of initial design points B: number of replications per initial design point	$T_{exp} = bN$ N: number of infill points b: number of replications per infill point	$T_{acc} = N(B - b)$

While efforts have been made (see, e.g., Quan et al. (2013) and Hernández L. et al. (2016)) to allocate the budget dynamically during the run of the optimization algorithm, we explicitly chose static resampling during the initial design and exploratory phases of the algorithm, using B and b replications respectively. The reason for this is twofold. Firstly, the issue of how to optimally distribute the budget while sampling remains an important challenge in current research (see, e.g., Jalali et al. (2017) for comments on the heuristic presented in Quan et al. (2013); see also Binois et al. (2018) for an approach to dynamically allocate budget for resampling existing points versus exploring new points). Secondly, as the algorithm uses stochastic kriging to construct the metamodel, we expect that the model outcomes *inherently* take into account the resulting amount of noise in the observations, so the choice of B and b should not be so crucial; we also test this explicitly in our experiments, varying B and b across high versus low replication budgets (as discussed

in Section 4). Only in the accuracy phase, extra replications are added dynamically to the already sampled points (as discussed in section 3.2).

3.2. SK-MOCBA: Algorithm outline

The algorithm focuses on solving the following multiobjective optimization problem:

$$\min_{\mathbf{x}\in D}[f^1(\mathbf{x}), f^2(\mathbf{x}), ..., f^m(\mathbf{x})] \tag{4}$$

for m objectives in the objective space Θ, and decision vectors $\mathbf{x} = [x_1, ..., x_d]^T$ in the decision space D. The objectives are analytically intractable; they can only be observed through (noisy) simulation estimates. Algorithm 2 shows the sequential steps performed by the proposed algorithm, which we refer to as SK-MOCBA.

As is common in kriging-based optimization, **step 1** uses a maximin latin hypercube sample to obtain a space filling set of initial design points. As suggested in the literature (Jones et al., 1998; Forrester et al., 2008), we set $n_0 = 11d - 1$, with d the dimension of the design space D. In **step 2** we run a fixed number of replications B per design point, yielding an estimate of the response for each objective, and an estimate of each response variance, for the points in the initial design. In **step 3** the *exploratory* loop is initialized by first normalizing the objectives with respect to their observed ranges so that each objective function lies between $[0, 1]$. Then the observed means of the objectives are combined into one scalarized objective $Z_{\boldsymbol{\lambda}}(\mathbf{x})$, which is, by consequence, also noisy. As is common in the literature, we use the *augmented Tchebycheff* scalarization function (see e.g., Miettinen (1999); Knowles (2006); Zhang et al. (2010)):

$$Z_{\boldsymbol{\lambda}}(\mathbf{x}) = \max_{j=1,..,m}(\lambda^j f^j(\mathbf{x})) + \rho \sum_{j=1}^{m} \lambda^j f^j(\mathbf{x}) \tag{5}$$

with $0 \leq \lambda^j \leq 1, \sum_{j=1}^{m} \lambda^j = 1, \forall j \in \{1, .., m\}$, and ρ is a small positive value (e.g., $\rho = 0.05$). This scalarization approach is chosen (as opposed to, e.g., the weighted sum approach) because the nonlinear term of the function ensures that points on nonconvex regions of the front can be detected, while the linear term ensures that *weak* Pareto optimal solutions are rewarded less than *strict* Pareto optimal solutions (Miettinen & Mäkelä, 2002; Knowles, 2006). Each set of λ values is defined as a weight vector $\boldsymbol{\lambda} = \lambda^1, ..., \lambda^m$. By assigning different weight vectors at each iteration, we aim at exploring points that are sufficiently diverse (i.e., spread across different areas of the Pareto front).

In **step 4**, we use stochastic kriging to approximate the scalarized objectives over the design space. The kriging information is then used in **step 5** to compute the *Modified Expected Improvement* (MEI) infill criterion (Quan et al., 2013) over the design space, in order to select a new infill point to simulate:

$$\text{MEI}(\mathbf{x}) = E\{\max[\hat{Z}(\mathbf{x}_{\min}) - Z_N^*(\mathbf{x}), 0]\} \tag{6}$$

Algorithm 2 SK-MOCBA

Input:

n_0 ←Number of initial design points that we wish to sample.
B ←Number of replications per initial design point.
N ←Number of points that we wish to sample during the exploratory phase.
b ←Number of replications per infill point sampled during the exploratory phase.
B_{max} ←Maximum number of replications per sampled point.

Output:

PF_{obs} → The observed Pareto front.
PS_{obs} → The observed Pareto set.

Initial design

$S = \emptyset$: Initialize the set of sampled points.

Step 1: Construct the initial design (LHS), using a maximin latin hypercube sample of n_0 points.

Step 2: Simulate B replications for each initial design point and update the set of sampled points: $S \leftarrow S \cup$ LHS.

Exploratory phase

for $i = 1 : N$ **do**

Step 3: Normalize the objective values of all simulated points so far in S. Randomly select a weight vector $\boldsymbol{\lambda}$ and scalarize the normalized objectives of each point $\mathbf{x}$ into $Z_{\boldsymbol{\lambda}}(\mathbf{x})$.

Step 4: Fit a stochastic kriging metamodel to the scalarized objective values: $\hat{Z}_{\boldsymbol{\lambda}}(\mathbf{x})$.

Step 5: Search and select the infill point $\mathbf{x}^i$ with highest MEI using the kriging information.

Step 6: Simulate b replications on the selected point and update the set of sampled points: $S \leftarrow S \cup \{\mathbf{x}^i\}$.

end for

Accuracy phase: MOCBA procedure

$\beta_i \leftarrow$ Total number of replications run so far for any point $\mathbf{x}^i \in S$.

$S_{max} = \emptyset$: Initialize set of points that have reached B_{max} replications.

while $T_{acc} > 0$ **do**

Step 7: Evaluate equations 11 - 17 in Appendix A to determine the subsets of sampled points regarded as dominated (S_A) and non-dominated (S_B).

Step 8: Compute and normalize the allocation quantities α_i for each $\mathbf{x}^i \in S$, according to equations 9 and 10 (see Appendix A).

Step 9:

for each $\mathbf{x}^i \in S$ **do**

$\gamma_i = \alpha_i T_{acc}$

if $\gamma_i + \beta_i < B_{max}$ **then**

Run γ_i replications on point $\mathbf{x}^i$.

Update the number of replications run on point $\mathbf{x}^i$: $\beta_i = \beta_i + \gamma_i$.

else

$\gamma_i = B_{max} - \beta_i$

Run γ_i replications on point $\mathbf{x}^i$.

Update S and S_{max}: $S \backslash \{\mathbf{x}^i\}$, $S_{max} \cup \{\mathbf{x}^i\}$

end if

Step 10: Update the available accuracy budget: $T_{acc} = T_{acc} - \gamma_i$.

end for

end while

Identification of the observed PF and the observed PS

Step 11:

Update the set of candidate points: $S \leftarrow S \cup S_{max}$.

Run a non-dominated sort on S to identify PF_{obs} and PS_{obs}.

where $\hat{Z}(\mathbf{x}_{\min})$ is the stochastic kriging prediction at $\mathbf{x}_{\min} = \arg\min_{\mathbf{x}\in S} \bar{Z}(\mathbf{x})$ (i.e. the alternative with the lowest sample mean among the already sampled points); and $Z_N^*(\mathbf{x})$ is the normal random variable:

$$Z_N^*(\mathbf{x}) \sim \mathcal{N}[\hat{Z}(\mathbf{x}), \hat{s}_d(\mathbf{x})] \tag{7}$$

where the mean $\hat{Z}(\mathbf{x})$ is the stochastic kriging prediction at solution $\mathbf{x}$ (Eq. 2), and $\hat{s}_d^2(\mathbf{x})$ the estimated *deterministic* kriging variance (see Jones et al. (1998)). The MEI criterion can be calculated as:

$$\text{MEI}(\mathbf{x}) = [\hat{Z}(\mathbf{x}_{\min}) - \hat{Z}(\mathbf{x})]\Phi\left(\frac{\hat{Z}(\mathbf{x}_{\min}) - \hat{Z}(\mathbf{x})}{\hat{s}_d(\mathbf{x})}\right) + \hat{s}_d(\mathbf{x})\phi\left(\frac{\hat{Z}(\mathbf{x}_{\min}) - \hat{Z}(\mathbf{x})}{\hat{s}_d(\mathbf{x})}\right) \tag{8}$$

where Φ denotes the normal cumulative distribution and ϕ denotes the normal probability density function. When the design space is continuous, maximizing MEI requires an iterative search procedure; a metaheuristic approach (e.g., a genetic algorithm) could be used to avoid getting stuck in a local optimum (see, e.g., Knowles (2006); Scott et al. (2011)). By definition, the optimality of such a heuristic approach can't be guaranteed; moreover, its performance also typically depends on a number of user-defined parameters (such as population size, mutation and crossover rates, etc.). To avoid these issues in our experiments, we discretized the search space into a large but finite set of points (see Section 4.2): the algorithm then determines MEI for all unvisited alternatives, and chooses the alternative with the highest MEI as the next infill point. In addition, discretizing the search space facilitates the performance evaluation of the algorithm in the design space, as discussed further in Section 4.3.

In **step 6**, the point with highest MEI is added to the set of sampled points by performing b simulation replications at the new point. The MEI criterion is analogous to the EI criterion of Jones et al. (1998), except for the use of the *stochastic* kriging predictor (Eq. 2) instead of the deterministic kriging predictor. This is a straightforward choice, given the heterogeneous nature of the noise. The MEI criterion still balances local exploitation and global exploration of the design space, as does the original EI criterion, through the use of $\hat{s}_d(\mathbf{x})$: this helps the algorithm to escape local optima, directing it towards promising regions of the response surface while at the same time reducing the spatial uncertainty of the metamodel (Quan et al., 2013). Steps (3) to (6) are repeated until the maximum number of iterations (N) has been performed. Note that the same weight vector can be repeated during the algorithm; the same scalarized objective function may thus be minimized several times during the run of the algorithm, based upon the kriging information available at that iteration.

Once the exploratory budget has been depleted, the algorithm moves to the Accuracy phase (steps 7 to 10), where the MOCBA procedure (Multi-Objective Computing Budget Allocation) is used to allocate extra replications to competitive designs in the set of already sampled points. In this way, we aim to maximize the probability of correctly selecting the

true non-dominated points. We focus on the simplified MOCBA procedure presented in Chen & Lee (2010); as shown by the authors, there is no significant difference in the results when compared with the original MOCBA procedure discussed in Lee et al. (2010). As we implement the procedure without further changes, we do not discuss it in detail; the related expressions can be found in Appendix A, and the interested reader is referred to Lee et al. (2010) and Chen & Lee (2010) for further discussion of the MOCBA framework.

In **step 7**, MOCBA labels each sampled point as *dominated* or *non-dominated*, based on the observed objective values and their respective variances (see expressions 14 and 15). In **step 8**, the allocation share of the remaining accuracy budget for each individual point is computed using equations 9 and 10, then normalized. These new replications are performed in **step 9**, and the number of replications run so far on point $\mathbf{x}^i$ are updated. In order to avoid MOCBA replicating excessively on a single point, the user may want to limit the maximum number of replications allowed for any given point by choosing a specific upper bound B_{max} (which should in any case be larger than B, to allow the accuracy budget to be used). Points that have reached a total of B_{max} replications are not further considered when allocating a new set of replications in the accuracy phase.

The accuracy budget is then updated in **step 10**. When the accuracy budget is depleted, the algorithm proceeds to *identify* the *observed* Pareto front in **step 11**, by considering the mean objective values observed for all sampled points, and running a non-dominated sorting on the observed means (Deb et al., 2002).

4. Design of numerical experiments

This section discusses the setup of the experiments. Section 4.1 gives details on the test functions. Section 4.2 outlines the different scenarios. Section 4.3 explains the performance metrics used, and Section 4.4 discusses the implementation details .

4.1. Test functions

To assess the performance of the algorithms, we run it on three well-known, scalable multiobjective functions from the ZDT and DTLZ test suites, which are commonly used in the literature to test multiobjective optimizers. The selected test problems differ in the geometries of the resulting Pareto fronts. We summarize the functions in Table 2. As evident from this table, ZDT1 is a bi-objective function scalable only in the number of dimensions d, while the DTLZ functions are scalable in the number of dimensions d and number of objectives m. Figure 1 displays the shape of the Pareto fronts for these test problems, for two and three objectives. We refer to Huband et al. (2006) for further details on the properties and characteristics of these test problems.

Table 2: Analytical benchmark functions

Name	Function	Properties
ZDT1	$\min f^1 = x_1$ $\min f^2 = g \times h(f_1, g)$ $g = 1 + \frac{9}{d}\sum_{i=2}^{d} x_i$ $h = 1 - \sqrt{\frac{f_1}{g}}$	Domain = $[0,1]$ Geometry: convex
DTLZ6	$\min f^1 = (1+g)\prod_{i=1}^{m-1}\cos(\frac{\pi}{2}\theta_i)$ $\min f^{j=2:m-1} = (1+g)\left(\prod_{i=1}^{m-j}\cos(\frac{\pi}{2}\theta_i)\right)\sin(\frac{\pi}{2}\theta_{m-j+1})$ $\min f^m = (1+g)\sin(\frac{\pi}{2}\theta_1)$ $\theta_i = \frac{\pi(1+2gx_i)}{4(1+g)}$, for $i = \{2,...,m-1\}$ $g = \sum_{i=1}^{d} x_i^{0.1}$	Domain = $[0,1]$ Geometry: $m \leq 3$: concave $m \geq 4$: unknown
DTLZ7	$f^{j=1:m-1} = x_j$ $f^m = (1+g)\left(m - \sum_{i=1}^{m-1}\left[\frac{f_i}{1+g}(1+\sin(3\pi f_i))\right]\right)$ $g = 1 + \frac{9}{d}\sum_{i=1}^{d} x_i$	Domain = $[0,1]$ Geometry: disconnected

As is common in the literature (see e.g., Picheny (2015), we discretize the search space using a Sobol sequence (i.e., a quasi-Monte Carlo sampling method with desirable space-filling properties). Other appropriate sequences can be used such as the Halton sequence, though we didn't see a significant difference in results during preliminary experiments. We set the number of discrete points in the search space equal to $1000 \times d$ (see Lemieux (2009) for further details on quasi-Monte Carlo sampling); this set of points is further referred to as the design space D. Given that the test problems are analytical functions, the points in D that form the Pareto set can be exactly determined, along with their true objective values.

To test the performance of SK-MOCBA, all objectives are perturbed by heterogeneous Gaussian noise. More specifically, we obtain noisy observations $\tilde{f}_k^j(\mathbf{x}^i) = f^j(\mathbf{x}^i) + \epsilon_k(\mathbf{x}^i)$, with $\epsilon_k(\mathbf{x}^i) \sim \mathcal{N}(0, \tau^j(\mathbf{x}^i))$ for $j = \{1,...,m\}$ objectives at the k^{th} replication. As in the experiments in Huang et al. (2006) and Picheny et al. (2013), the magnitude of the noise imposed on any objective j is determined based on the range of that objective over the design space D: $R_f^j = \max_{\mathbf{x}\in D} f^j(\mathbf{x}) - \min_{\mathbf{x}\in D} f^j(\mathbf{x}), \forall j \in \{1,...,m\}$. Analogous to the single-objective experiments in Jalali et al. (2017), we consider a *low noise* case, where the standard deviation of the noise ($\tau^j(\mathbf{x})$) for each objective, after B replications, varies between $0.01 \times R_f^j$ and $0.5 \times R_f^j$, and a *high noise* case, where it varies between $0.5 \times R_f^j$ and $1.5 \times R_f^j$. In between these limits, $\tau^j(\mathbf{x})$ decreases linearly with the objective value: $\tau^j(\mathbf{x}) = a^j f^j(\mathbf{x}) + a^j \times b^j$, $\forall j \in \{1,...,m\}$. We thus assume minimum noise at the global minimum of each individual objective: $\min_{\mathbf{x}\in D} \tau^j(\mathbf{x})$ at $\min_{\mathbf{x}\in D} f^j(\mathbf{x})$ and $\max_{\mathbf{x}\in D} \tau^j(\mathbf{x})$ at $\max_{\mathbf{x}\in D} f^j(\mathbf{x})$. We run the experiments without the use of common random numbers, since its use increases the variance of the kriging predictor and variance of the constant trend estimator, as shown in Chen et al. (2012).

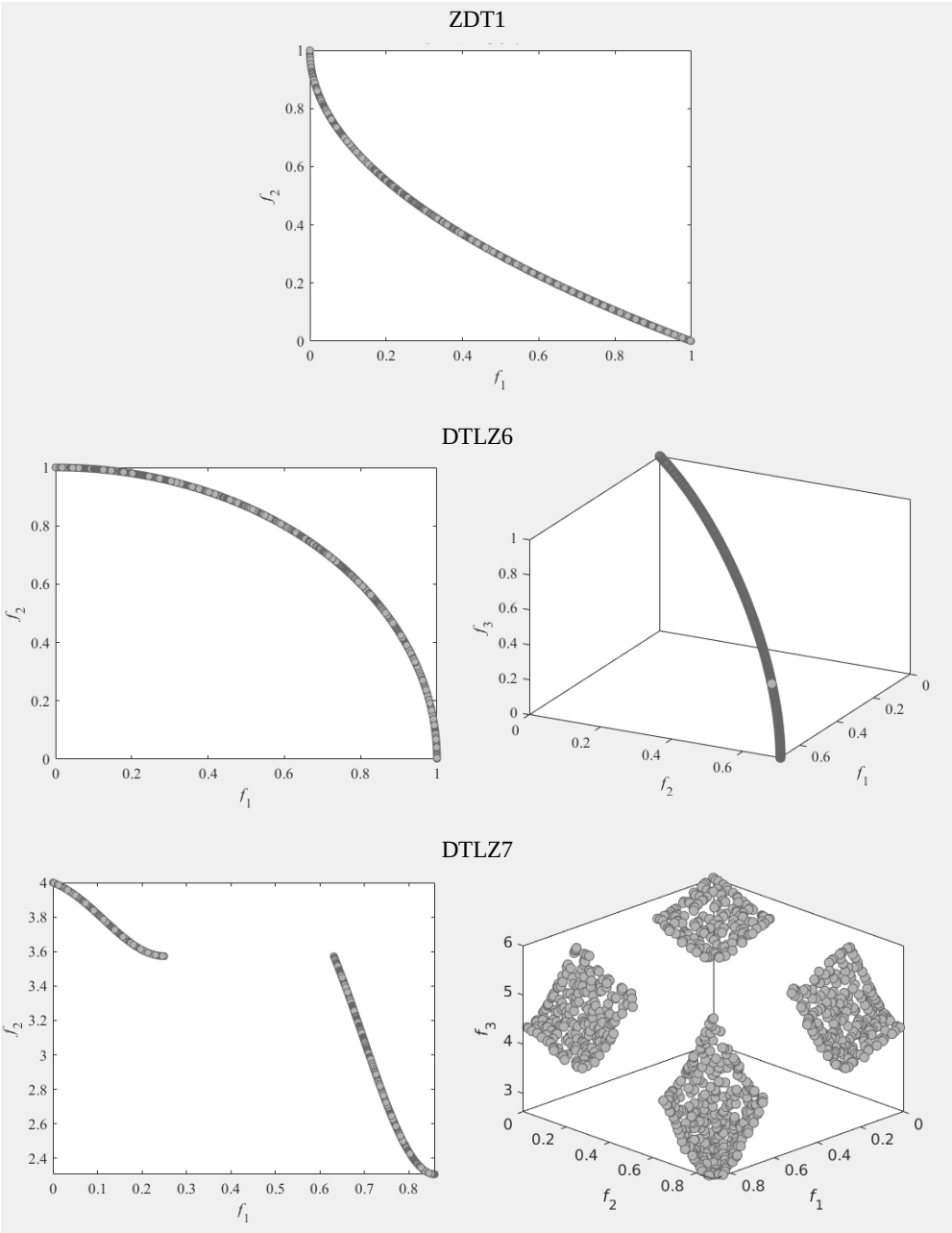

Figure 1: Bi-objective (left and center) and tri-objective (right) Pareto fronts of the test functions. The DTLZ testbed is scalable in both the number of objectives and decision variables; the ZDT1 (bi)objective function is only scalable in the number of decision variables.

4.2. Parameters and scenarios

As prevalent in the literature, we fit all stochastic kriging models with a constant trend (Eq. 1), by means of maximum likelihood estimation. We use the Gaussian kernel (see Chapter 4 in Rasmussen & Williams (2005) for more details on this and other covariance functions). Furthermore, we consider high and low replication budgets (i.e., the values for b, B and B_{max}, and N). For ease of reference, we summarize the parameters used in the experiments in Table 3.

Table 3: Summary of parameters for the experiments

Kriging metamodels	
Kernel: Gaussian	$k_G(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left[-\sum_{i=1}^{m} \left(\frac{\lvert \mathbf{x}_i - \mathbf{x}'_i \rvert}{\sqrt{2} l_i}\right)^2\right]$
Noise levels	
Low noise	$0.01 \times R_f^j \leq \tau(\mathbf{x}) \leq 0.5 \times R_f^j$
High noise	$0.5 \times R_f^j \leq \tau(\mathbf{x}) \leq 1.5 \times R_f^j$
Design space size	
LHS	$11d - 1$
Quasi-random sequence	$d \times 1000$
Exploratory and accuracy budgets	
Number of evaluations N	Low: 150 High: 300
rep_L: Low replication budget	$b = 25$ $B = 50$ $B_{max} = 100$
rep_H: High replication budget	$b = 50$ $B = 100$ $B_{max} = 200$

Table 4 gives an overview of the scenarios evaluated in the experiments. As evident from the table, we vary the number of dimensions, number of objectives, and the replication budgets across the scenarios: the first and second scenarios for each test function only differ in the replication budget used (low versus high), while the third scenario increases the number of dimensions and/or objectives (keeping the high budget). The test functions also clearly differ in the range of the objective outcomes: the range of the m^{th} objective for the DTLZ7 function is clearly much larger than those of the ZDT1 and DTLZ6 objectives. Each scenario is evaluated with high and low noise levels. The notation ref refers to the reference point used to compute the hypervolume indicator (see Section 4.3).

4.3. Performance metrics

In deterministic multiobjective settings, the quality of the Pareto front is usually evaluated by a quantitative indicator, such as the *hypervolume* (HV) and/or the *inverted generational distance* (IGD) (Zitzler et al., 2008). The hypervolume is the portion (volume) of the objective space covered by a particular Pareto front with respect to a given reference point (see Table 4 for the reference points used in the experiments), whereas the IGD measures the Euclidean distance between a member of the approximated Pareto front and the closest member of a reference front (e.g., the true Pareto front). Thus, the former is to be maximized, and the latter minimized.

Yet, as we study a stochastic setting, the Pareto front resulting from our algorithm is characterized by *noisy* estimates of the objective values. While MOCBA intends to

Table 4: Overview of the 9 scenarios for the experiments, each one run with low and high noise levels.

ZDT1	$R_f^1 \simeq 1$ $R_f^2 \simeq 1.93$	$R_f^1 \simeq 1$ $R_f^2 \simeq 1.93$	$R_f^1 \simeq 1$ $R_f^2 \simeq 2.94$
	$d = 5$ $m = 2$ $N = 150$ Budget: rep_L $\|PS\| \simeq 75$	$d = 5$ $m = 2$ $N = 150$ Budget: rep_H $\|PS\| \simeq 75$	$d = 10$ $m = 2$ $N = 150$ Budget: rep_H $\|PS\| \simeq 100$
	$ref = [2, 2]$	$ref = [2, 2]$	$ref = [2, 2]$
DTLZ6	$1.79 \lesssim R_f^{j=1:m} \lesssim 1.99$	$1.79 \lesssim R_f^{j=1:m} \lesssim 1.99$	$2.62 \lesssim R_f^{j=1:m} \lesssim 2.99$
	$d = 3$ $m = 3$ $N = 300$ Budget: rep_L $\|PS\| \simeq 90$	$d = 3$ $m = 3$ $N = 300$ Budget: rep_H $\|PS\| \simeq 90$	$d = 5$ $m = 4$ $N = 150$ Budget: rep_H $\|PS\| \simeq 90$
	$ref = [2, 2, 2]$	$ref = [2, 2, 2]$	$ref = [2, 2, 2, 2]$
DTLZ7	$R_f^1 \simeq 1$ $R_f^2 \simeq 18.53$	$R_f^1 \simeq 1$ $R_f^2 \simeq 18.53$	$R_f^{j=1:m-1} \simeq 1$ $R_f^m \simeq 25.27$
	$d = 5$ $m = 2$ $N = 300$ Budget: rep_L $\|PS\| \simeq 75$	$d = 5$ $m = 2$ $N = 300$ Budget: rep_H $\|PS\| \simeq 75$	$d = 10$ $m = 3$ $N = 300$ Budget: rep_H $\|PS\| \simeq 230$
	$ref = [2, 5]$	$ref = [2, 5]$	$ref = [2, 2, 8]$

maximize the probability of correct selection of points (i.e., the true best points), it does not eliminate the noise on the outcomes. The resulting estimates for HV and IGD are thus also noisy. For that reason, we also evaluate the performance of the algorithm in the design space, by evaluating the quality of the *Pareto set* obtained at the end of the algorithm (PS_{obs}). Ideally, this set should coincide perfectly with the *true* Pareto set (PS) present in the design space D. Differences between PS and PS_{obs} may be due to multiple causes:

1. Some members of PS_{obs} may be erroneously labeled as non-dominated due to the noisy objective outcomes; this was referred to as a Type 2 error in the Introduction, and we report the total number of these Type 2 errors as $ET2$.

2. The algorithm may not even have *sampled* all members of PS during the search. This could be due to an inadequate choice of the number of iterations (N); yet, for all experiments in Section 5, we ensure that the size of PS is smaller than N, so it is theoretically possible for the algorithm to find the entire true set of non-dominated points. The subset of points in PS that were effectively sampled by the algorithm is referred to as PS_s. We report on the *percentage* of PS points that was effectively sampled as $PS_s(\%)$; this thus gives an indication of the search effectiveness of the algorithm.

3. Among the PS_s points sampled, some may not have been correctly *identified* as Pareto optimal at the end of the algorithm, due to noise. The subset of PS_s that was correctly identified as Pareto optimal is referred to as PS_{ID}, and we report on

the *percentage* of PS_s points that were correctly identified as $PS_{ID}(\%)$. This thus gives an indication of the identification accuracy of the algorithm. Points in PS_s that were erroneously labeled as dominated are referred to as Type 1 error points (as discussed in the Introduction), and are reported in the counter $ET1$.

For ease of reference, Table 5 summarizes the notations, while Table 6 gives an overview of the performance measures used.

Table 5: Overview of the different sets of points used to evaluate the algorithm.

Notation	Description
PS	True Pareto set in the design space D.
PS_{obs}	Observed Pareto set.
PS_s	Subset of points in PS that have been sampled.
PS_{ID}	Subset of points in PS_s that have been correctly identified as Pareto-optimal.

Table 6: Overview of performance measures used to evaluate the algorithms.

Notation	Description
HV	Hypervolume determined by the PF obtained with respect to a reference point.
IGD	Inverted generational distance of the PF obtained with respect to the true front.
PS_s	Percentage of points in the true PS that was effectively sampled: $PS_s(\%) = \frac{\lvert PS_s \rvert}{\lvert PS \rvert}$.
PS_{ID}	Percentage of points in PS_s that was correctly identified as Pareto optimal: $PS_{ID}(\%) = \frac{\lvert PS_{ID} \rvert}{\lvert PS_s \rvert}$
$ET1$	Number of points in PS_s that are incorrectly identified at dominated (Type I error).
$ET2$	Number of points in PS_{obs} that are incorrectly identified as non-dominated (Type II error).

As discussed in the Introduction, we explicitly wish to evaluate the impact of the following elements in the SK-MOCBA procedure:

1. the use of the stochastic kriging metamodel and MEI (as opposed to deterministic kriging and EI) during the search phase;

2. the added value of using MOCBA (as opposed to omitting a MORS procedure) in the accuracy phase.

To that end, we compare the results of SK-MOCBA to the results that are obtained by its deterministic counterpart (referred to as DK/EI, as it uses deterministic kriging and EI), and a counterpart that uses SK and MEI, but leaves out the MOCBA procedure (referred to as SK/MEI). For a fair comparison, each of these counterparts gets the same total replication budget as SK-MOCBA; yet, as they don't have an accuracy phase, they sample each infill point in the exploratory phase with B replications (so $T_{exp} = BN$ and $T_{acc} = 0$). Note that the DK/EI algorithm is very similar to the ParEGO algorithm (Knowles, 2006), except that we do not use a genetic algorithm for maximizing EI during the search, but exhaustively search the (discretized) search space for the next infill point.

4.4. Implementation details

All experiments are coded in Matlab; MOCBA's source code is written in C, but used in Matlab. The code used for *deterministic kriging* is based on the open source STK toolbox available here (http://kriging.sourceforge.net/htmldoc/); for *stochastic kriging* it is based on the code available on the official website (users.iems.northwestern.edu/∼nelsonb/SK/). The computations of the non-dominated set of points are done using the well-known *non-dominated sorting* algorithm of Deb et al. (2002), and the computations of the hypervolume are based on the algorithm in Tian et al. (2017). For each scenario (see Table 4), the algorithms were run 5 times, each time using a new LHS design and a new search space D (these runs are referred to as *macroreplications*; every algorithm was tested on the same set of macroreplications). The results for the performance metrics shown in all tables are the average values over these 5 macroreplications. All experiments are performed on a Dell laptop running 64-bit Linux Debian, with an Intel i7 VPro CPU with 8 cores, 2.40 GHz processing speed, and 16 GB of RAM; the average running time of the algorithms (in seconds) is reported as T_c.

5. Results

Tables 7, 8 and 9 summarize the performance metrics for the ZDT1, DTLZ6 and DTLZ7 functions, with low and high noise levels. We show the average metrics obtained over 5 macroreplications; for completeness, the numbers between brackets show the minimum and maximum values of the corresponding metric across the 5 macroreplications. The most important results in the tables are shown in bold.

For the ZDT1 function, the use of SK/MEI over DK/EI yields a big improvement in search efficiency: the $PS_s(\%)$ improves drastically, reaching 100% in both high and low noise cases, even with a low replication budget (see first and second scenarios). The choice of the budget thus appears to be less crucial when stochastic kriging is used; as expected, the fact that it accounts for the noise prevents the search from being misguided. Yet, the identification accuracy in SK/MEI remains low: while a high percentage of Pareto optimal points is sampled during the search, many of these points are incorrectly labeled as dominated (resulting in a high ET1), and only a relatively low percentage is correctly identified as Pareto optimal ($PS_{ID}(\%)$). The additional use of MOCBA (as in SK-MOCBA) drastically improves this accuracy, particularly in the low noise cases. In the high noise cases, SK-MOCBA still succeeds in sampling a very high number of Pareto points (see $PS_s(\%)$), in spite of the fact that it can spend less replications in the exploratory phase than its counterparts. Yet, the accuracy budget now seems insufficient to bring the identification accuracy to a really high level. This issue is illustrated in Figure 2, which shows the Pareto fronts observed after running the algorithms, along with the ET1 and ET2 points, for 1 given macroreplication of the first scenario (low budget).

Table 7: Summary of results for the ZDT1 function (average value over 5 macroreplications, [minimum value; maximum value]).

ZDT1 function with low noise									
	$d=5; m=2;\ N=150$; Budget: rep_L; $\lvert PS\rvert=75$			$d=5; m=2;\ N=150$; Budget: rep_H; $\lvert PS\rvert=75$			$d=10; m=2; N=150$; Budget: rep_H; $\lvert PS\rvert=100$		
	DK/EI	SK/MEI	SK-MOCBA	DK/EI	SK/MEI	SK-MOCBA	DK/EI	SK/MEI	SK-MOCBA
HV	3.656 [3.65;3.66]	3.662 [3.66;3.67]	3.660 [3.66;3.67]	3.655 [3.65;3.66]	3.660 [3.65;3.66]	3.660 [3.66;3.67]	3.653 [3.65;3.66]	3.662 [3.66;3.67]	3.660 [3.65;3.66]
IGD	0.014 [0.014;0.015]	0.011 [0.010;0.012]	0.007 [0.006;0.007]	0.011 [0.008;0.014]	0.007 [0.007;0.008]	0.004 [0.003;0.004]	0.012 [0.011;0.013]	0.007 [0.007;0.008]	0.005 [0.004;0.009]
ET1	6.2 [4;10]	18.2 [15;23]	6.6 [4;9]	1.6 [1;2]	8.2 [6;11]	0.2 [0;1]	0.6 [0;2]	15 [10;18]	3.6 [1;6]
ET2	0.4 [0;1]	1.6 [0;4]	0.4 [0;1]	0.2 [0;1]	1.0 [0;2]	0.4 [0;1]	0.2 [0;1]	0.4 [0;1]	0.6 [0;1]
$PS_s(\%)$	**61.07** [56.00;66.67]	**100** [100;100]	**100** [100;100]	**64.27** [49.33;77.33]	**99.73** [98.67;100]	**100** [100;100]	**39.40** [37.08;45.00]	**93.00** [90.01;96.05]	**93.80** [86.01;99.04]
$PS_{ID}(\%)$	**52.80** [49.33;56.01]	**75.73** [69.33;80.02]	**91.20** [88.00;94.66]	**62.13** [48.04;74.66]	**88.80** [85.33;90.66]	**99.73** [98.66;100]	**38.80** [37.01;43.07]	**78.01** [75.02;81.03]	**90.20** [82.03;98.01]
$T_c(s)$	98.34 [98.03;98.98]	148.51 [142.46;162.05]	148.01 [145.82;150.80]	98.51 [97.59;99.74]	154.41 [146.23;167.51]	154.80 [143.60;160.97]	230.95 [230.58;231.38]	479.27 [455.41;521.65]	454.53 [433.63;498.69]

ZDT1 function with high noise									
	$d=5; m=2;\ N=150$; Budget: rep_L; $\lvert PS\rvert=75$			$d=5; m=2;\ N=150$; Budget: rep_H; $\lvert PS\rvert=75$			$d=10; m=2; N=150$; Budget: rep_H; $\lvert PS\rvert=100$		
	DK/EI	SK/MEI	SK-MOCBA	DK/EI	SK/MEI	SK-MOCBA	DK/EI	SK/MEI	SK-MOCBA
HV	3.677 [3.63;3.71]	3.756 [3.73;3.78]	3.689 [3.66;3.72]	3.688 [3.68;3.70]	3.701 [3.67;3.73]	3.679 [3.67;3.70]	3.699 [3.68;3.74]	3.732 [3.72;3.75]	3.689 [3.68;3.70]
IGD	0.055 [0.05;0.06]	0.063 [0.06;0.07]	0.031 [0.03;0.03]	0.031 [0.03;0.04]	0.029 [0.02;0.03]	0.017 [0.02;0.02]	0.040 [0.03;0.05]	0.042 [0.03;0.05]	0.023 [0.02;0.02]
ET1	13.4 [5;19]	58.0 [55;63]	44.2 [40;47]	11.6 [8;15]	43.4 [38;48]	31.6 [29;33]	16.2 [12;21]	66.6 [63;71]	58.8 [53;66]
ET2	5.6 [3;8]	4.0 [3;6]	4.6 [2;8]	2.0 [0;6]	2.4 [0;5]	2.5 [0;4]	1.4 [0;3]	2.8 [1;5]	2.8 [2;5]
$PS_s(\%)$	35.73 [21.33;44.01]	**100** [100;100]	**97.87** [89.33;100]	46.67 [38.67;52.06]	**100** [100;100]	**100** [100;100]	35.40 [30.12;40.02]	**95.00** [91.03;100]	**99.20** [96.34;100]
$PS_{ID}(\%)$	17.87 [14.66;20.15]	**22.67** [16.14;26.67]	**38.93** [36.08;41.33]	31.20 [28.00;33.33]	**42.13** [36.03;49.33]	**57.87** [56.21;61.33]	19.20 [18.04;22.11]	**28.40** [26.14;31.31]	**40.40** [34.04;47.02]
$T_c(s)$	100.58 [99.65;102.04]	127.16 [123.41;130.72]	119.95 [114.63;126.55]	99.59 [98.37;100.74]	144.61 [136.27;151.99]	140.26 [133.31;147.95]	230.37 [229.51;231.15]	379.36 [369.56;394.41]	363.18 [354.67;372.12]

Table 8: Summary of results for the DTLZ6 function (average value over 5 macroreplications, [minimum value; maximum value]).

DTLZ6 function with low noise									
	$d=3; m=3; N=300$; Budget: rep_L; $\|PS\|=90$			$d=3; m=3; N=300$; Budget: rep_H; $\|PS\|=90$			$d=5; m=4; N=150$; Budget: rep_H; $\|PS\|=90$		
	DK/EI	SK/MEI	SK-MOCBA	DK/EI	SK/MEI	SK-MOCBA	DK/EI	SK/MEI	SK-MOCBA
HV	6.103 [6.08;6.12]	6.138 [6.11;6.15]	6.102 [6.10;6.11]	6.031 [5.98;6.08]	6.101 [6.09;6.10]	6.085 [6.08;6.09]	11.502 [11.46;11.56]	11.549 [11.50;11.62]	11.507 [11.49;11.52]
IGD	0.038 [0.03;0.06]	0.031 [0.02;0.04]	0.016 [0.01;0.02]	0.032 [0.02;0.05]	0.017 [0.015;0.020]	0.010 [0.01;0.01]	0.105 [0.09;0.11]	0.069 [0.04;0.12]	0.085 [0.05;0.12]
ET1	24.8 [19;31]	50.8 [44;57]	42.8 [37;47]	20.0 [18;22]	43.2 [38;47]	37.4 [32;44]	7.0 [5;9]	21.8 [20;24]	18.6 [15;24]
ET2	1.0 [0;3]	0.2 [0;1]	0.4 [0;1]	0.4 [0;1]	1.0 [0;2]	0.6 [0;2]	0.8 [0;2]	1.0 [0;4]	1.2 [0;3]
$PS_s(\%)$	**56.05** [50.00;61.63]	**100** [100;100]	**100** [100;100]	**55.58** [47.67;60.47]	**100** [100;100]	**100** [100;100]	**46.35** [43.53;48.24]	**83.29** [77.65;88.23]	**74.82** [71.76;81.18]
$PS_{ID}(\%)$	**27.21** [23.26;31.40]	**40.93** [33.72;48.83]	**50.23** [45.35;56.98]	**32.33** [23.26;37.21]	**49.77** [45.35;55.81]	**56.52** [48.84;62.79]	**38.12** [35.29;41.18]	**57.67** [52.94;61.18]	**52.94** [44.71;61.18]
$T_c(s)$	138.85 [137.55;140.13]	263.8 [248.22;286.49]	310.6 [291.53;328.99]	139.01 [138.75;139.12]	212.05 [205.98;219.96]	270.97 [260.76;286.45]	101.94 [100.64;102.75]	237.49 [221.36;260.62]	169.86 [154.83;182.93]

DTLZ6 function with high noise									
	$d=3; m=3; N=300$; Budget: rep_L; $\|PS\|=90$			$d=3; m=3; N=300$; Budget: rep_H; $\|PS\|=90$			$d=5; m=4; N=150$; Budget: rep_H; $\|PS\|=90$		
	DK/EI	SK/MEI	SK-MOCBA	DK/EI	SK/MEI	SK-MOCBA	DK/EI	SK/MEI	SK-MOCBA
HV	6.637 [6.45;6.83]	6.792 [6.66;6.93]	6.411 [6.38;6.44]	6.302 [6.22;6.38]	6.383 [6.30;6.44]	6.234 [6.17;6.30]	12.090 [11.72;12.98]	12.389 [12.19;12.90]	11.928 [11.82;11.99]
IGD	0.161 [0.11;0.24]	0.108 [0.08;0.14]	0.075 [0.06;0.09]	0.078 [0.06;0.11]	0.066 [0.05;0.08]	0.042 [0.03;0.05]	0.163 [0.15;0.18]	0.123 [0.10;0.14]	0.087 [0.07;0.10]
ET1	30.4 [28;34]	62.2 [59;66]	54.6 [50;63]	31.6 [24;39]	57.6 [51;61]	53.6 [49;58]	7.2 [6;10]	25.8 [21;32]	22.6 [21;26]
ET2	14.0 [9;19]	12.4 [9;16]	10.6 [6;16]	7.8 [1;14]	11.0 [5;22]	7.0 [3;10]	14.8 [7;21]	9.2 [3;17]	5.2 [2;7]
$PS_s(\%)$	50.23 [40.70;56.98]	**100** [100;100]	**91.39** [84.88;98.84]	59.54 [52.33;69.77]	**100** [100;100]	**100** [100;100]	36.94 [31.76;44.70]	**83.53** [75.29;87.05]	**78.82** [75.29;83.53]
$PS_{ID}(\%)$	14.88 [8.14;20.93]	**27.67** [23.26;31.40]	**27.91** [24.42;30.24]	22.79 [12.79;30.23]	**33.02** [29.07;40.70]	**37.67** [32.56;43.02]	28.47 [24.71;32.94]	**53.18** [47.06;58.82]	**52.23** [48.23;56.47]
$T_c(s)$	135.44 [133.70;138.23]	239.25 [230.32;251.30]	231.14 [217.81;243.09]	137.97 [134.99;139.52]	292.64 [276.71;307.33]	255.36 [244.49;262.53]	102.06 [101.50;102.96]	126.67 [124.21;129.10]	121.23 [116.89;124.19]

Table 9: Summary of results for the DTLZ7 function (average value over 5 macroreplications, [minimum value; maximum value]).

DTLZ7 function with low noise									
	$d=5;\ m=2; N=300;$ Budget: $rep_L; \lvert PS\rvert = 75$			$d=5;\ m=2; N=300;$ Budget: $rep_H; \lvert PS\rvert = 75$			$d=10; m=3; N=300;$ Budget: $rep_H; \lvert PS\rvert = 230$		
	DK/EI	SK/MEI	SK-MOCBA	DK/EI	SK/MEI	SK-MOCBA	DK/EI	SK/MEI	SK-MOCBA
HV	4.436 [4.42;4.45]	4.457 [4.45;4.46]	4.427 [4.42;4.43]	4.425 [4.41;4.43]	4.421 [4.42;4.43]	4.418 [4.41;4.42]	17.343 [17.32;17.37]	17.351 [17.3;17.4]	17.349 [17.3;17.4]
IGD	0.033 [0.03;0.04]	0.036 [0.03;0.04]	0.020 [0.02;0.02]	0.019 [0.01;0.02]	0.018 [0.01;0.02]	0.009 [0.007;0.011]	0.033 [0.03;0.04]	0.026 [0.02;0.03]	0.014 [0.011;0.017]
ET1	35.80 [33;42]	40.8 [38;44]	29.6 [23;34]	25.4 [19;29]	27.2 [25;29]	14.4 [11;17]	6.0 [2;9]	8.2 [3;14]	4.4 [3;7]
ET2	0.8 [0;2]	1.2 [0;3]	0.6 [0;1]	0.8 [0;3]	0.2 [0;1]	0.8 [0;3]	1.0 [0;2]	0.8 [0;2]	1.0 [0;2]
$PS_s(\%)$	**85.07** [80.21;90.67]	**92.23** [92.33;93.33]	**93.33** [93.33;93.33]	**87.47** [81.33;93.33]	**93.33** [93.33;93.33]	**93.33** [93.33;93.33]	**87.03** [85.71;89.01]	**100** [100;100]	**100** [100;100]
$PS_{ID}(\%)$	**37.33** [34.67;42.67]	**38.93** [34.66;42.66]	**53.87** [48.00;62.66]	**53.60** [45.33;60]	**57.07** [54.66;60]	**74.13** [70.66;78.66]	**80.44** [75.82;83.51]	**90.99** [84.60;96.71]	**95.17** [92.32;96.71]
$T_c(s)$	255.93 [254.54;257.09]	514.59 [492.25;532.07]	481.38 [468.14;492.62]	253.89 [252.15;256.15]	519.50 [498.64;548.57]	487.74 [474.46;499.16]	667.65 [659.05;674.56]	1454.5 [1380.8;1568.4]	1228.3 [1177.8;1281.6]

DTLZ7 function with high noise									
	$d=5;\ m=2; N=300;$ Budget: $rep_L; \lvert PS\rvert = 75$			$d=5;\ m=2; N=300;$ Budget: $rep_H; \lvert PS\rvert = 75$			$d=10; m=3; N=300;$ Budget: $rep_H; \lvert PS\rvert = 230$		
	DK/EI	SK/MEI	SK-MOCBA	DK/EI	SK/MEI	SK-MOCBA	DK/EI	SK/MEI	SK-MOCBA
HV	5.192 [4.84;5.57]	5.761 [5.46;6.08]	4.948 [4.76;5.04]	4.974 [4.70;5.22]	4.990 [4.89;5.12]	4.696 [4.57;4.88]	18.550 [18.3;18.79]	18.619 [18.22;19.19]	17.816 [17.6;18.1]
IGD	0.339 [0.08;0.51]	0.340 [0.24;0.41]	0.133 [0.11;0.16]	0.142 [0.11;0.21]	0.186 [0.16;0.25]	0.082 [0.07;0.09]	0.175 [0.14;0.21]	0.144 [0.12;0.17]	0.091 [0.08;0.10]
ET1	18.8 [15;22]	61.4 [60;63]	55.6 [51;58]	41.4 [45;46]	58.6 [57;61]	54.2 [53;56]	43.4 [39;52]	44.4 [37;51]	23.0 [17;27]
ET2	1.8 [0;3]	4.4 [1;8]	4.2 [1;9]	2.8 [2;3]	3.2 [1;5]	2.8 [2;4]	8.6 [3;13]	7.0 [2;10]	10.0 [6;20]
$PS_s(\%)$	33.6 [29.33;36.02]	**93.33** [93.33; 93.33]	**93.33** [93.33;93.33]	70.67 [60.41;78.67]	**93.33** [93.33;93.33]	**93.33** [93.33;93.33]	88.79 [84.62;91.20]	**100** [100;100]	**100** [100;100]
$PS_{ID}(\%)$	8.53 [5.33;13.33]	**11.47** [9.34;13.34]	**19.20** [16.00;25.33]	15.46 [13.34;17.34]	**15.20** [12.00;17.33]	**21.07** [18.66;22.66]	41.10 [32.97;48.35]	**51.21** [44.32;59.34]	**74.73** [70.31;81.35]
$T_c(s)$	259.58 [256.35;261.15]	431.84 [426.09;439.03]	423.83 [393.30;447.62]	253.91 [252.84;255.49]	428.38 [415.60;439.35]	395.67 [387.28;414.80]	653.99 [646.80;664.64]	1050.20 [1010.4;1094.7]	1155.4 [1074.9;1252.9]

Clearly, the use of stochastic kriging (as in SK/MEI and SK-MOCBA) guarantees a big jump in the number of true Pareto points visited during the exploratory search; yet, while SK/MEI shows a high number of ET1 points at the end of the algorithm (with very inaccurate observed objective values in the high noise case), SK-MOCBA succeeds in improving the objective estimates and decreasing ET1, particularly in the low noise case; in the high noise case, however, the identification accuracy is much lower.

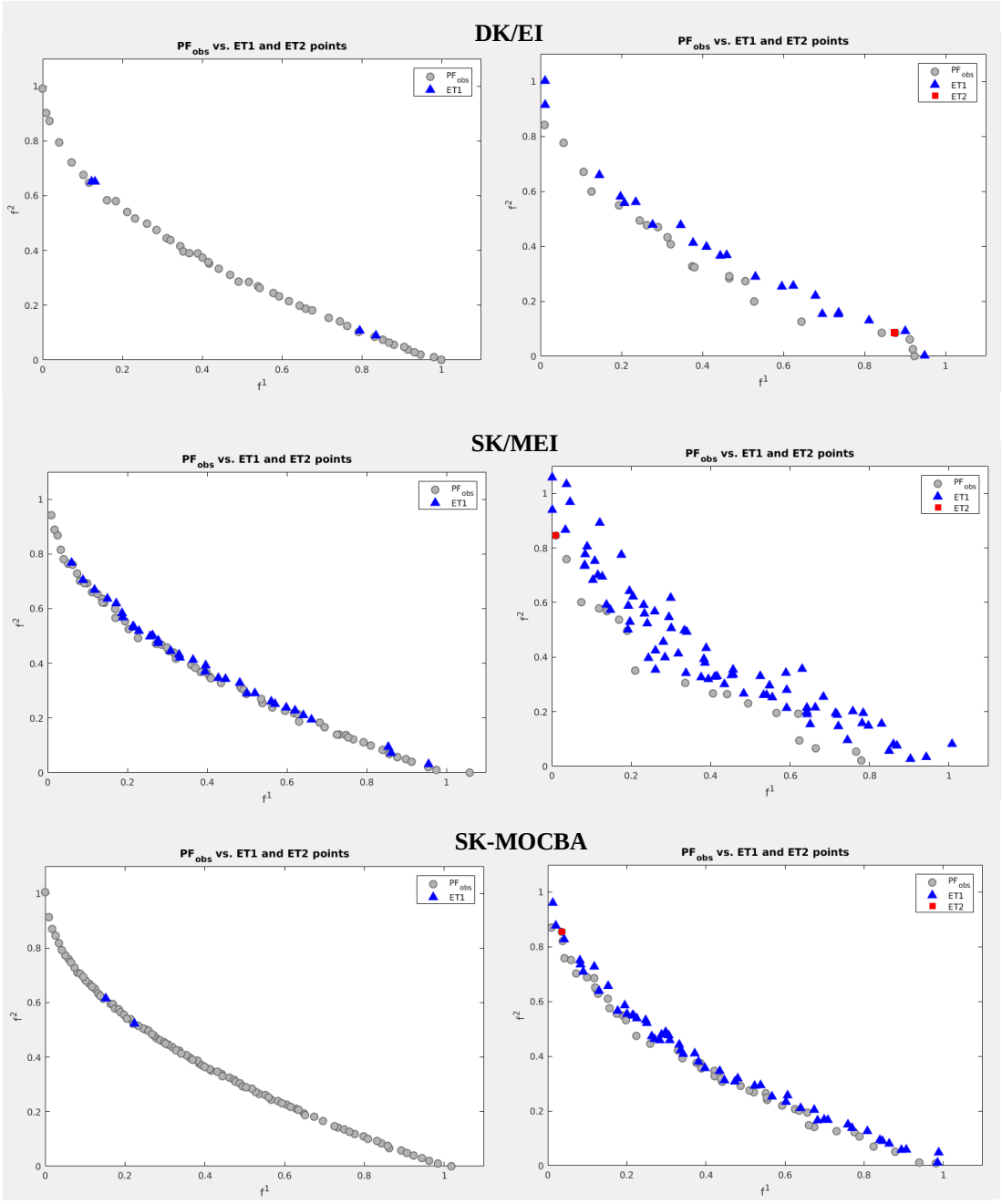

Figure 2: ZDT1 function, first scenario. The left column shows the low noise case, and the right column the high noise case. The figure displays the observed Pareto front along with the error Type I and Type II points resulting from each algorithm, for a given macroreplication.

As MOCBA sorts all points as either dominated or non-dominated based on the observed means and variances, and aims at maximizing the probability of correct selection, the procedure may allocate a lot of accuracy budget to points that appear to *marginally* dominate other points (or vice versa), while they are actually both (non-)dominating. Clearly, this can waste a lot of budget in the high noise case, without effectively improving the correct identification of the Pareto optimal points.

A similar type of conclusion is evident from the DTLZ6 results. The use of stochastic kriging (as in SK/MEI and SK-MOCBA) again guarantees a jump in search efficiency both in the low *and* high noise cases (recall that SK-MOCBA needs to split its replication budget across the exploratory and accuracy phases, contrary to its counterparts, yet it still effectively performs the search even with high noise on the objective estimates). Naturally, $PS_s(\%)$ suffers in the third scenario, even when SK/MEI or SK-MOCBA are used, as the number of evaluations N is cut in half. The advantage of using MOCBA is less clear for this function: as evident from the results, the intervals of the $PS_{ID}(\%)$ results tend to overlap rather substantially for SK/MEI and SK-MOCBA, resulting in relatively small differences in the average $PS_{ID}(\%)$.

The results for the DTLZ7 function again confirm the previous findings: the use of stochastic kriging is key in improving the search efficiency of the algorithms, while MOCBA appears to be relatively ineffective in achieving a high identification accuracy with the given budgets, particularly in the high noise cases; in the low noise case, the increase in budget in the second scenario is sufficient to yield a significant improvement in $PS_{ID}\%$ compared with the first scenario. Figure 3 illustrates the observed Pareto fronts, along with the Type I and Type II errors, for a given macroreplication of the first scenario. Again, we see that the use of MOCBA succeeds in yielding better estimates for the truly non-dominating points (note the shift of ET1 points towards the front, in both the high and low noise cases); yet, it is not able to solve the ET1 errors by the end of the algorithm. The noise level has again a detrimental effect during the accuracy stage; see for instance some of the cases with high noise, where SK/MEI and SK-MOCBA sample more than 93% of the true Pareto set, but correctly identify only 11% and 19% respectively.

Note also that the HV indicator is very misleading in a stochastic setting, often showing little to no difference in the quality of the observed fronts for the different algorithms, or yielding contradictory results (showing better outcomes for DK/EI and/or SK/MEI, while actually the number of ET1 and/or ET2 points is worse than for SK-MOCBA). Indeed, due to the absence of an accuracy phase, the observed Pareto front obtained by DK/EI and SK/MEI may remain very inaccurate, yielding "better" HV results than the more accurate estimates obtained by SK-MOCBA. On the contrary, the IGD indicator is consistent with the improvements obtained by SK-MOCBA, as it measures the distance between the (more accurate) observed front estimates and (in our experiments) the true Pareto front.

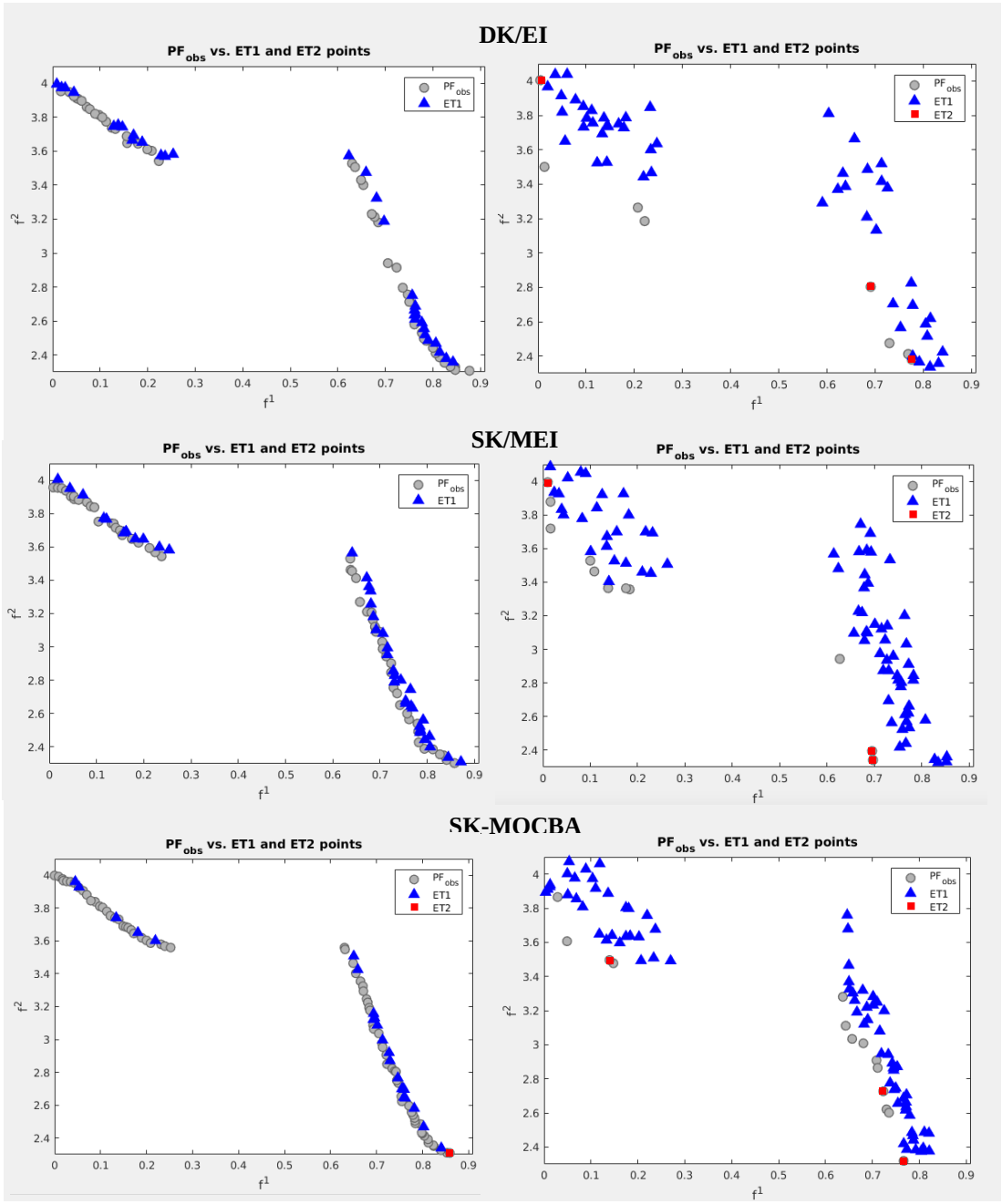

Figure 3: DTLZ7 function, first scenario. The left column shows the low noise case, and the right column the high noise case. The figure displays the observed Pareto front along with the error Type I and Type II points resulting from each algorithm, for a given macroreplication.

6. Conclusions and future research

In this work, we studied the benefits of using stochastic kriging (along with the MEI infill criterion) and the MOCBA procedure for optimizing multiobjective problems in settings with heterogeneous noise on the objectives. The resulting algorithm, SK-MOCBA, was evaluated on three analytical test functions. The results clearly show that the use of stochastic kriging yields big improvements in search efficiency, irrespective of the noise

level. Apparently, the fact that the intrinsic noise is inherently taken into account in the metamodel and the infill criterion prevents the search from being misguided. The use of MOCBA is effective in improving the accuracy of the observed Pareto points (moving them closer to the true Pareto front), yet the budget requirement for reducing/avoiding ET1 errors is high, as the procedure may allocate a lot of budget to points that appear to marginally dominate others (denoted as S_B in Appendix A), while actually they are all truly non-dominated.

The remarkable performance of stochastic kriging improves the prospects for optimization problems with limited budget and heterogeneous noise, as it allows to shift more budget to the accuracy phase. Evidently, the current research on multiobjective R&S algorithms may also yield algorithms that outperform MOCBA for the same budget. While MOCBA aims at maximizing the probability of correct selection, it may be more appropriate to adopt an indifference zone approach: indeed, it is unlikely that a decision maker will truly differentiate between marginally different solutions in real life.

Finally, our results have highlighted that the traditional hypervolume performance metric, can be very misleading in stochastic settings, as the objective outcomes are noisy. As we tested the algorithms on known analytical functions perturbed by noise, we could reliably evaluate the performance of the algorithms in the design space. Also, we could calculate the IGD metric using the true Pareto front as the reference front. Obviously, such approaches are not possible in real-life problems, as the true Pareto points are unknown. The current literature lacks of solutions or guidelines to reliably assess the Pareto front quality in such cases: all too often, one simply resorts to computing deterministic performance measures (such as HV and IGD) on the estimated objective outcomes, thereby ignoring the inherent noise. Some of the few works include Syberfeldt et al. (2010) and Fieldsend & Everson (2015), but we consider this issue as a main challenge for further research.

Appendix A MOCBA allocation rules

The allocation rules for the simplified MOCBA algorithm Chen & Lee (2010) are summarized below (the notation used in Section 2 is slightly modified for simplification purposes):

$\bar{f}_{ij}$: The averaged observed performance of design i for objective j after a certain number of replications.

p_i : The design that dominates design i with the highest probability.

$j^i_{p_i}$: The objective j of p_i that dominates the corresponding objective of design i with the lowest probability.

τ_{ij} : The observed intrinsic variance of design i for objective j after a certain number of replications.

α_i : The budget allocation for design i.

S: The entire set of sampled points.

S_A: The subset of designs in S labeled as being dominated.

S_B: The subset of designs in S labeled as being non-dominated.

For any given design $g, h \in S_A$ and $d \in S^B$:

$$\alpha_h = \left(\frac{\tau^2_{hj^h_{p_h}} / \delta_{hp_hj^h_{p_h}}}{\tau^2_{gj^g_{p_g}} / \delta_{gp_gj^g_{p_g}}} \right)^2 \tag{9}$$

$$\alpha_d = \sqrt{\sum_{h \in D_d} \frac{\tau^2_{dj^h_d}}{\tau^2_{hj^h_d}} \alpha^2_h} \tag{10}$$

where

$$\delta_{ipj} = \bar{f}_{ij} - \bar{f}_{pj} \tag{11}$$

$$j^i_p = \underset{j \in \{1,\ldots,m\}}{\arg\min}\ P(\bar{X}_{pj} \leq \bar{X}_{ij}) = \underset{j \in \{1,\ldots,m\}}{\arg\max} \frac{\delta_{ipj}|\delta_{ipj}|}{\tau^2_{ij} + \tau^2_{pj}} \tag{12}$$

$$p_i = \underset{\substack{p \in S \\ p \neq i}}{\arg\max} \prod_{j=1}^{m} P(\bar{f}_{pj} \leq \bar{f}_{ij}) = \underset{\substack{p \in S \\ p \neq i}}{\arg\min} \frac{\delta_{ipj^i_p}|\delta_i pj^i_p|}{\tau^2_{ij^i_p} + \tau^2_{pj^i_p}} \tag{13}$$

$$S_A = \left\{ h | h \in S, \frac{\delta^2_{hp_hj^h_{p_h}}}{\tau^2_{hj^h_{p_h}} + \tau^2_{p_hj^h_{p_h}}} < \min_{i \in D_h} \frac{\delta^2_{ihj^i_h}}{\tau^2_{ij^i_h} + \tau^2_{hj^i_h}} \right\} \tag{14}$$

$$S_B = S \setminus S_A \tag{15}$$

$$D_h = \{i | i \in S, p_i = h\} \tag{16}$$

$$D_d = \{h | h \in S_A, p_h = d\} \tag{17}$$

Acknowledgment

This research was supported by the Research Foundation-Flanders, grant number G076815.

References

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

Binois, M., Huang, J., Gramacy, R. B., & Ludkovski, M. (2018). Replication or exploration? sequential design for stochastic simulation experiments. *Technometrics*, *0*, 1–43.

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

Chen, C.-h., & Lee, L. H. (2010). *Stochastic simulation optimization: an optimal computing budget allocation* volume 1. World Scientific.

Chen, X., Ankenman, B. E., & Nelson, B. L. (2012). The effects of common random numbers on stochastic kriging metamodels. *ACM Trans. Model. Comput. Simul.*, *22*, 7:1–7:20.

Chen, X., & Kim, K.-K. (2014). Stochastic kriging with biased sample estimates. *ACM Transactions on Modeling and Computer Simulation*, *24*, 8:1–8:23.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*, 182–197.

Fieldsend, J. E., & Everson, R. M. (2015). The rolling tide evolutionary algorithm: A multiobjective optimizer for noisy optimization problems. *IEEE Transactions on Evolutionary Computation*, *19*, 103–117.

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

Fu, M. C. (2015). *Handbook of Simulation Optimization*. New York, NY: Springer.

Gramacy, R. B., & Lee, H. K. (2012). Cases for the nugget in modeling computer experiments. *Statistics and Computing*, *22*, 713–722.

Hernández L., D., Hernandez-Lobato, J., Shah, A., & Adams, R. (2016). Predictive entropy search for multi-objective bayesian optimization. In *International Conference on Machine Learning* (pp. 1492–1501).

Horn, D., Dagge, M., Sun, X., & Bischl, B. (2017). First investigations on noisy model-based multi-objective optimization. In H. T. et al. (Ed.), *International Conference on Evolutionary Multi-Criterion Optimization* (pp. 298–313). Springer.

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

Huband, S., Hingston, P., Barone, L., & While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, *10*, 477–506.

Hunter, S. R., Applegate, E. A., Arora, V., Chong, B., Cooper, K., Rincón-Guevara, O., & Vivas-Valencia, C. (2019). An introduction to multiobjective simulation optimization. *ACM Trans. Model. Comput. Simul.*, *29*, 7:1–7:36.

Jalali, H., Van Nieuwenhuyse, I., & Picheny, V. (2017). Comparison of kriging-based algorithms for simulation optimization with heterogeneous noise. *European Journal of Operational Research*, *261*, 279 – 301.

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

Kim, S.-H., & Nelson, B. L. (2006). Chapter 17: Selecting the Best System. In S. G. Henderson, & B. L. Nelson (Eds.), *Simulation* (pp. 501 – 534). Elsevier volume 13 of *Handbooks in Operations Research and Management Science*.

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

Kleijnen, J. P. C. (2015). *Design and Analysis of Simulation Experiments*. (2nd ed.). NY: Springer.

Knowles, J. (2006). ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Trans. Evol. Comput.*, *10*, 50–66.

Koch, P., Wagner, T., Emmerich, M. T., Bäck, T., & Konen, W. (2015). Efficient multi-criteria optimization on noisy machine learning problems. *Applied Soft Computing*, *29*, 357–370.

Lee, L. H., Chew, E. P., Teng, S., & Goldsman, D. (2010). Finding the non-dominated pareto set for multi-objective simulation models. *IIE Transactions*, *42*, 656–674.

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

Miettinen, K. (1999). *Nonlinear multiobjective optimization* volume 12. Springer Science & Business Media.

Miettinen, K., & Mäkelä, M. M. (2002). On scalarizing functions in multiobjective optimization. *OR spectrum*, *24*, 193–213.

Picheny, V. (2015). Multiobjective optimization using gaussian process emulators via stepwise uncertainty reduction. *Statistics and Computing*, *25*, 1265–1280.

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

Quan, N., Yin, J., Ng, S. H., & 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., & Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive computation and machine learning)*. (1st ed.). Cambridge, Massachusetts, USA: The MIT Press.

Rojas-Gonzalez, S., Jalali, H., & Van Nieuwenhuyse, I. (2018). A stochastic-kriging-based multiobjective simulation optimization algorithm. In M. e. a. Rabe (Ed.), *Proceedings of the 2018 Winter Simulation Conference* (pp. 2155–2166). Piscataway, New Jersey, USA: IEEE.

Rojas-Gonzalez, S., & Van Nieuwenhuyse, I. (2019). A survey on kriging-based infill algorithms for multiobjective simulation optimization. *Computers and Operations Research*, *In press*.

Sacks, J., Welch, W. J., Mitchell, T. J., & Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical Science*, (pp. 409–423).

Santner, T. J., Williams, B. J., & Notz, W. I. (2013). *The design and analysis of computer experiments*. Springer Science & Business Media.

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

Syberfeldt, A., Ng, A., John, R. I., & Moore, P. (2010). Evolutionary optimisation of noisy multi-objective problems using confidence-based dynamic resampling. *European Journal of Operational Research*, *204*, 533–544.

Tian, Y., Cheng, R., Zhang, X., & Jin, Y. (2017). PlatEMO: A matlab platform for evolutionary multi-objective optimization. *IEEE Computational Intelligence Magazine*, *12*, 73–87.

Van Beers, W. C., & Kleijnen, J. P. C. (2003). Kriging for interpolation in random simulation. *Journal of the Operational Research Society*, *54*, 255–262.

Wagner, T., Emmerich, M., Deutz, A., & Ponweiser, W. (2010). On expected-improvement criteria for model-based multi-objective optimization. *Parallel Problem Solving from Nature, PPSN XI*, (pp. 718–727).

Wang, G. G., & Shan, S. (2007). Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical design*, *129*, 370–380.

Zhang, Q., Liu, W., Tsang, E., & Virginas, B. (2010). Expensive multiobjective optimization by MOEA/D with Gaussian process model. *IEEE Transactions on Evolutionary Computation*, *14*, 456–474.

Zitzler, E., Knowles, J., & Thiele, L. (2008). Quality assessment of pareto set approximations. In J. Branke, K. Deb, K. Miettinen, & R. Słowiński (Eds.), *Multiobjective*

Optimization: Interactive and Evolutionary Approaches (pp. 373–404). Berlin, Heidelberg: Springer Berlin Heidelberg.

Zuluaga, M., Krause, A., & Püschel, M. (2016). e-pal: An active learning approach to the multi-objective optimization problem. *Journal of Machine Learning Research*, *17*, 1–32.