Efficient Global Optimization for High-Dimensional Constrained Problems by Using the Kriging Models Combined with the Partial Least Squares Method

Mohamed Amine Bouhlel · Nathalie Bartoli · Rommel G. Regis · Abdelkader Otsmane · Joseph Morlier

Abstract In many engineering optimization problems, the number of function evaluations is often very limited because of the computational cost to run one high-fidelity numerical simulation. Using a classic optimization algorithm, such as a derivative-based algorithm or an evolutionary algorithm, directly on a computational model is not suitable in this case. A common approach to addressing this challenge is to use black-box surrogate modeling techniques. The most popular surrogate-based optimization algorithm is the Efficient Global Optimization (EGO) algorithm, which is an iterative sampling algorithm that adds one (or many) point(s) per iteration. This algorithm is often based on an infill sampling criterion, called expected improvement, which represents a trade-off between promising and uncertain areas. Many studies have shown the efficiency of EGO, particularly when the number of input variables is relatively low. However, its performance

M. A. Bouhlel
Department of Aerospace Engineering, University of Michigan, 1320 Beal avenue, Ann Arbor, MI, 48109, USA
E-mail: mbouhlel@umich.edu

N. Bartoli
ONERA, 2 avenue Édouard Belin, 31055 Toulouse, France
Tel.: +33-(0)5-62252644
E-mail: nathalie.bartoli@onera.fr

J. Morlier
Université de Toulouse, Institut Clément Ader, CNRS, ISAE-SUPAERO, 10 Avenue Edouard Belin, 31055 Toulouse Cedex 4, France
Tel.: +33-(0)5-61338131
E-mail: joseph.morlier@isae-SUPAERO.fr

R. G. Regis
Saint Joseph's University, 5600 City Avenue, Philadelphia, PA 19131, USA
Tel.: (610)660-1514
E-mail: rregis@sju.edu

on high-dimensional problems is still poor since the Kriging models used are time-consuming to build. To deal with this issue, this paper introduces a surrogate-based optimization method that is suited to high-dimensional problems. The method first uses the "locating the regional extreme" criterion, which incorporates minimizing the surrogate model while also maximizing the expected improvement criterion. Then, it replaces the Kriging models by the KPLS(+K) models (Kriging combined with the partial least squares method), which are more suitable for high-dimensional problems. Finally, the proposed approach is validated by a comparison with alternative methods existing in the literature on some analytical functions and on 12-dimensional and 50-dimensional instances of the benchmark automotive problem "MOPTA08".

Keywords Kriging, KPLS, Partial Least Squares, Optimization, Expected Improvement

1 Introduction

The field of optimization has become significantly important in engineering and industrial design. In the literature, many approaches have been used for finding the global optimum. One of the most popular methods is to run a gradient-based optimization algorithm with a multi-start procedure (Hickernell and Yuan 1997; Sendin and Banga 2009). For this type of method, the derivative of the objective function is needed. When the derivative of the objective function is not available, it can be approximated by finite-difference methods. However, finite-difference methods are unreliable when the function to be optimized is non-smooth. For this reason, derivative-free optimization methods (Conn et al 2009) and heuristic methods such as simulated annealing (Laarhoven and Aarts 1987), evolutionary algorithms (Simon 2013) and particle swarm optimization (Kennedy and Eberhart 1995) became popular in the last few decades.

In many engineering problems, numerical simulations require several hours, sometimes more, to run one simulation representative of the physics. Thus, direct optimization methods for such cases are too expensive and infeasible in practice. In addition, many iterations are often required when the number of input variables is large. Likewise, to find the global optimum using a relatively small number of evaluations of the true function is almost impossible for high-dimensional problems. Hopefully, it is possible to obtain a reasonably good feasible solution through a good management of the number of function evaluations.

A suitable optimization approach is to use a surrogate model, also called metamodel, instead of the true function. The evaluation of new points through a surrogate model, called predictions, is very cheap since it consists in computing a simple analytical function. Moreover, the surrogate model is a good tool for indicating promising areas where new high-fidelity simulations should be run. Several approaches for constrained black-box optimization have been developed recently. For instance, Li et al (2016) developed a Kriging-based algorithm for constrained black-box optimization that involves two phases.

The first phase finds a feasible point while the second phase obtains a better feasible point. This approach is similar to what is done in the COBRA algorithm (Regis 2014, briefly described in Sec. 4.2.2 in this paper). Liu et al (2016) proposed a two-phase method that uses a constraint-handling technique based on the DIRECT algorithm and that uses an adaptive metamodeling strategy for the objective and constraints. Gramacy et al (2016) developed an hybrid approach that uses Kriging with the expected improvement (EI) criterion combined with the augmented Lagrangian framework for constrained optimization. Moreover, Regis and Wild (2017) developed a model-based trust-region algorithm for constrained optimization that uses RBF models. In addition, Kriging-based approaches have been proposed for constrained multi-objective optimization (e.g., Singh et al (2014); Feliot et al (2016)). An important application of Kriging in an optimization context is the Efficient Global Optimization (EGO) algorithm (Jones et al 1998). It uses both the prediction and error estimations provided by Kriging to guide an infill sampling criterion towards promising areas. To the best of the authors' knowledge, EGO is used for problems with a relatively small number of input variables because of the Kriging limitations in high dimension (Haftka et al 2016). Sasena (2002) has developed a constrained version of EGO, called SuperEGO (denoted SEGO in this paper), where the optimization of the infill sampling criterion takes into account the satisfaction of constraint functions.

In this paper, two new methods for solving high-dimensional constrained optimization problems are developed, both based on SEGO approach: (i) SEGOKPLS uses SEGO with the KPLS model (Bouhlel et al 2016b); and (ii) SEGOKPLS+K uses SEGO with the KPLS+K model (Bouhlel et al 2016a). The SEGOKPLS(+K) algorithms build KPLS(+K) for each output function at each iteration of optimization. Either KPLS or KPLS+K are used since they are faster to build than the classical Kriging models while maintaining a good accuracy for high-dimensional problems. Once the KPLS(+K) models are built, a specific infill sampling criterion, that adds promising points into the training points in order to improve the accuracy of the metamodels in relevant areas, is maximized. For this purpose, the "locating the regional extreme" (WB2) criterion proposed by Watson and Barnes (1995) is used. Sasena et al (2002) implemented and validated this criterion on several analytical functions. This approach proposes a balance between exploitation and exploration of the metamodel—herein, exploitation means that the metamodel is minimized and exploration means that points are added where the uncertainty of the metamodel is high. Moreover, WB2 is more local than the popular EI criterion. This latter characteristic is important for expensive high-dimensional problems, since the number of true function evaluations is limited and the uncertainty of the metamodel is high.

There have been other methods that use principal components for high-dimensional surrogate-based optimization. For example, Kyriacou et al (2014) used principal components analysis (PCA) in a metamodel-assisted evolutionary algorithm to guide the application of evolution operators and the training of the metamodel. Moreover, Chen et al (2015) used Karhunen-Loève expansion, which is similar to PCA, to reduce design-space dimensionality in metamodel-based shape optimization. The proposed approach, which is based on the EGO algorithm, incorporates a Kriging model that uses principal components in the context of partial least squares to make it suitable for high-dimensional constrained optimization. Several analytical problems and an automotive problem (MOPTA08) are tested with the proposed approaches. In addition, a comparison between SEGOKPLS(+K) and several optimization methods existing in the literature are done. This comparison has shown that the SEGOKPLS+(K) methods outperform the alternative algorithms on most of the test problems.

The paper is organized as follows. Section 2 details the proposed optimization algorithms, SEGOKPLS(+K). Sections 3 and 4 apply and validate the SEGOKPLS(+K) algorithms on the analytical and MOPTA08 problems, respectively. Finally, the Section 5 concludes the paper.

2 SEGOKPLS(+K) for high-dimensional constrained optimization problems

Assume that a deterministic cost function has been evaluated at n points $\mathbf{x}^{(i)}$ $(i = 1, \dots, n)$ with $\mathbf{x}^{(i)} = \left[x_1^{(i)}, \dots, x_d^{(i)}\right] \in B$, with $B \subset \mathbb{R}^d$. For simplicity, consider B to be a hypercube expressed by the product of intervals of each direction space, i.e., $B = \prod_{j=1}^{d} [a_j, b_j]$, where $a_j, b_j \in \mathbb{R}$ with $a_j \leq b_j$ for $j = 1, \dots, d$. Also, denote by $\mathbf{X}$ the matrix $\left[\left(\mathbf{x}^{(1)}\right)^t, \dots, \left(\mathbf{x}^{(n)}\right)^t\right]^t$. Evaluating these n inputs gives the outputs $\mathbf{y} = \left[y^{(1)}, \dots, y^{(n)}\right]^t$ with $y^{(i)} = y\left(\mathbf{x}^{(i)}\right)$, for $i = 1, \dots, n$.

2.1 The KPLS(+K) models

The KPLS and KPLS+K models are derived by combining the Kriging model with the Partial Least Squares (PLS) technique. Therefore, the theory of Kriging and PLS are briefly depicted before describing the KPLS(+K) models.

2.1.1 The Kriging model

The Kriging model has been developed first in geostatistics (Krige 1951; Cressie 1988; Goovaerts 1997) before being extended to computer experiments (Sacks et al 1989; Schonlau 1998; Jones et al 1998; Sasena et al 2002; Forrester et al 2006; Picheny et al 2010; Liem et al 2015). In order to develop the Kriging model, assume that the deterministic response $y(\mathbf{x})$ is a realization of a stochastic process $Y(\mathbf{x})$ (Koehler and Owen 1996; Schonlau 1998; Sasena 2002)

$$Y(\mathbf{x}) = \beta_0 + Z(\mathbf{x}), \qquad (1)$$

where β_0 is an unknown constant and $Z(\mathbf{x})$ is a stochastic term considered as a realization of a stationary Gaussian process with $\mathbb{E}[Z(\mathbf{x})] = 0$ and a covariance function, also called a kernel function, given by

$$\mathrm{Cov}(Z(\mathbf{x}), Z(\mathbf{x}')) = k(\mathbf{x}, \mathbf{x}') = \sigma^2 r(\mathbf{x}, \mathbf{x}') = \sigma^2 r_{\mathbf{x}\mathbf{x}'},\ \forall \mathbf{x}, \mathbf{x}' \in B, \tag{2}$$

where σ^2 is the process variance and $r_{\mathbf{x}\mathbf{x}'}$ is the correlation function between $\mathbf{x}$ and $\mathbf{x}'$. In the following, the exponential covariance function type is only considered, in particular the squared Gaussian kernel, given by

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \prod_{i=1}^{d} \exp\left(-\theta_i \left(x_i - {x'}_i\right)^2\right), \forall\, \theta_i \in \mathbb{R}^+. \tag{3}$$

Equation (1) corresponds to a particular case of the Kriging model: the ordinary Kriging model (Forrester et al 2008). The function k, given by Equation (3), depends on some hyperparameters θ. To construct the Kriging model, these hyperparameters are assumed to be known. Also, denote the $n \times 1$ vector as $\mathbf{r}_{\mathbf{x}\mathbf{X}} = [r_{\mathbf{x}\mathbf{x}^{(1)}}, \ldots, r_{\mathbf{x}\mathbf{x}^{(n)}}]^t$ and the $n \times n$ covariance matrix as $\mathbf{R} = [\mathbf{r}_{\mathbf{x}^{(1)}\mathbf{X}}, \ldots, \mathbf{r}_{\mathbf{x}^{(n)}\mathbf{X}}]$. Moreover, $\hat{y}(\mathbf{x})$ denotes the prediction of the true function $y(\mathbf{x})$. Under the hypothesis considered above, the best linear unbiased predictor for $y(\mathbf{x})$, given the observations $\mathbf{y}$, is

$$\hat{y}(\mathbf{x}) = \hat{\beta_0} + \mathbf{r}_{\mathbf{x}\mathbf{X}}^t \mathbf{R}^{-1} \left(\mathbf{y} - \hat{\beta_0}\mathbf{1}\right), \tag{4}$$

where $\mathbf{1}$ denotes an n-vector of ones and

$$\hat{\beta_0} = \left(\mathbf{1}^t \mathbf{R}^{-1} \mathbf{1}\right)^{-1} \mathbf{1}^t \mathbf{R}^{-1} \mathbf{y}. \tag{5}$$

In addition, the estimate of σ^2 is

$$\hat{\sigma}^2 = \frac{1}{n} \left(\mathbf{y} - \mathbf{1}\hat{\beta_0}\right)^t \mathbf{R}^{-1} \left(\mathbf{y} - \mathbf{1}\hat{\beta_0}\right). \tag{6}$$

Moreover, the ordinary Kriging model provides an estimate of the variance of the prediction given by

$$s^2(\mathbf{x}) = \hat{\sigma}^2 \left(1 - \mathbf{r}_{\mathbf{x}\mathbf{X}}^t \mathbf{R}^{-1} \mathbf{r}_{\mathbf{x}\mathbf{X}}\right). \tag{7}$$

Finally, the vector of hyperparameters $\theta = \{\theta_i\}$, for $i = 1, \ldots, d$, is estimated by the maximum likelihood estimation method.

2.1.2 The partial least squares technique

The PLS method is a statistical method that searches out the best multidimensional direction $\mathbf{X}$ that explains the characteristics of the output $\mathbf{y}$. Moreover, it finds a linear relationship between input variables and output variable by projecting input variables onto principal components (Wold 1966). The PLS technique reveals how inputs depend on the output variable. In the following, h denotes the number of principal components retained that is much lower than d ($h \ll d$). The PLS components are computed sequentially. In fact, the principal component $\mathbf{t}_l$ is computed by seeking the best direction $\mathbf{w}^{(l)}$ that maximizes the squared covariance between $\mathbf{t}_l = \mathbf{X}^{(l-1)}\mathbf{w}^{(l)}$ and $\mathbf{y}^{(l-1)}$

$$\mathbf{w}^{(l)} = \begin{cases} \underset{\mathbf{w}^{(l)}}{\operatorname{argmax}} \ {\mathbf{w}^{(l)}}^t {\mathbf{X}^{(l-1)}}^t \mathbf{y}^{(l-1)} {\mathbf{y}^{(l-1)}}^t \mathbf{X}^{(l-1)} \mathbf{w}^{(l)} \\ \text{s.t.} {\mathbf{w}^{(l)}}^t \mathbf{w}^{(l)} = 1. \end{cases} \tag{8}$$

where $\mathbf{X} = \mathbf{X}^{(0)}$, $\mathbf{y} = \mathbf{y}^{(0)}$, and, for $l = 1, \dots, h$, $\mathbf{X}^{(l)}$ and $\mathbf{y}^{(l)}$ is the residual matrix from the local regression of $\mathbf{X}^{(l-1)}$ onto the principal component $\mathbf{t}_l$ and from the local regression of $\mathbf{y}^{(l-1)}$ onto the principal component $\mathbf{t}_l$, respectively, such that

$$\begin{aligned} \mathbf{X}^{(l-1)} &= \mathbf{t}_l \mathbf{p}^{(l)} + \mathbf{X}^{(l)}, \\ \mathbf{y}^{(l-1)} &= c_l \mathbf{t}_l + \mathbf{y}^{(l)}, \end{aligned} \tag{9}$$

where $\mathbf{p}^{(l)}$ (a $1 \times d$ vector) and c_l (a coefficient) contain the regression coefficients. For more details of how the PLS method works, please see (Helland 1988; Frank and Friedman 1993; Alberto and González 2012).

The principal components represent the new coordinate system obtained upon rotating the original system with axes, $x_1, \dots, x_d$ (Alberto and González 2012). For $l = 1, \dots, h$, we write $\mathbf{t}_l$ as

$$\mathbf{t}_l = \mathbf{X}^{(l-1)}\mathbf{w}^{(l)} = \mathbf{X}\mathbf{w}_*^{(l)}. \tag{10}$$

This important relationship is mainly used for developing the KPLS model that is described in the following section.

2.1.3 Construction of the KPLS and KPLS+K models

The hyperparameters $\theta = \{\theta_i\}$, for $i = 1, \dots, d$, given by Equation (3) can be interpreted as measuring how strongly the variables $x_1, \dots, x_d$, respectively, affect the output y. For building KPLS, consider the coefficients given by the vectors $\mathbf{w}_*^{(l)}$, for $l = 1, \dots, h$, as a measure of the influence of input variables $x_1, \dots, x_d$ on the output y. By some elementary operations applied on the kernel functions, define the KPLS kernel by

$$k_{1:h}(\mathbf{x}, \mathbf{x}') = \prod_{l=1}^{h} k_l(F_l(\mathbf{x}), F_l(\mathbf{x}')), \tag{11}$$

where $k_l : B \times B \to \mathbb{R}$ is an isotropic stationary kernel and

$$\begin{aligned} F_l : B &\longrightarrow B \\ \mathbf{x} &\longmapsto \left[w_{*1}^{(l)} x_1, \dots, w_{*d}^{(l)} x_d \right]. \end{aligned} \tag{12}$$

More details of such construction are given in Bouhlel et al (2016b). Considering the example of the squared Gaussian kernel given by Equation (3) yields

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \prod_{l=1}^{h} \prod_{i=1}^{d} \exp\left[-\theta_l \left(w_{*i}^{(l)} x_i - w_{*i}^{(l)} {x'}_i \right)^2 \right], \forall\ \theta_l \in \mathbb{R}^+. \tag{13}$$

This equation is the new kernel associated to the KPLS model. Since a small number of principal components is retained, the estimation of the hyperparameters $\theta_1, \ldots, \theta_l$ is very fast compared to the hyperparameters $\theta_1, \ldots, \theta_d$ given by Equation (3), where $h << d$. Moreover, the solution of the maximum likelihood of the KPLS covariance function can be improved. To do this, add a new step, right after the estimation of the θ_l-parameters, that is based on the following transition

$$\begin{aligned} k_{1:h}(\mathbf{x}, \mathbf{x}') &= \sigma^2 \prod_{l=1}^{h} \prod_{i=1}^{d} \exp\left(-\theta_l {w_{*i}^{(l)}}^2 (x_i - {x'}_i)^2\right) \\ &= \sigma^2 \exp\left(\sum_{i=1}^{d} \sum_{l=1}^{h} -\theta_l {w_{*i}^{(l)}}^2 (x_i - {x'}_i)^2\right) \\ &= \sigma^2 \exp\left(\sum_{i=1}^{d} -\eta_i (x_i - {x'}_i)^2\right) \\ &= \sigma^2 \prod_{i=1}^{d} \exp\left(-\eta_i (x_i - {x'}_i)^2\right), \end{aligned} \tag{14}$$

with $\eta_i = \sum\limits_{l=1}^{h} \theta_l {w_{*i}^{(l)}}^2$, for $i = 1, \ldots, d$. Equation (14) makes it possible to express the hyperparameters' solution, provided by the KPLS kernel, from the reduced space (with h dimensions) into the original space (with d dimensions). Thus, $\eta_i = \sum\limits_{l=1}^{h} \theta_l {w_{*i}^{(l)}}^2$, for $i = 1, \ldots, d$, is used as a starting point for a gradient-based maximization of the likelihood function for a standard Kriging model. This optimization is done in the complete space where the vector $\eta = \{\eta_i\} \in {\mathbb{R}^+}^d$. Such construction is similar to the method developed by Ollar et al (2016), where a gradient-free optimization algorithm is used with an isotropic Kriging model followed by a gradient-based optimization starting from the solution given by the first optimization. More details of this model can be found in Bouhlel et al (2016a). The two models KPLS and KPLS+K developed for high-dimensional problems are denoted by KPLS(+K) below.

2.2 The SEGOKPLS(+K) algorithms

During the last decade, the EGO algorithm (Jones et al 1998) has become one of the most popular methods for surrogate-based optimization. It is an adaptive sampling algorithm that adds one (or more) point per cycle (Chevalier and Ginsbourger 2013). This algorithm uses the well known expected improvement (EI) criterion based on the prediction value and the uncertainty provided by the Kriging model. In this Section, this algorithm is adapted to constrained black-box optimization problems in high dimension.

2.2.1 Definition of the constrained optimization problem

The optimization problem with constraint functions is formalized as follows

$$\min_{\mathbf{x}\in B} \begin{cases} f(\mathbf{x}) \\ \text{s.t.} \\ g_k(\mathbf{x}) \leq 0,\ \ k = 1, \dots, m. \end{cases} \tag{15}$$

From this formulation, note that equality constraints are not considered in this paper. The functions f, $g_1, \dots, g_m$ are deterministic black-box functions that are computationally expensive. Moreover, assume that the derivatives of the objective function f and the constraint functions g_k $(k = 1, \dots, m)$ are unavailable. These assumptions are typical in many engineering applications.

2.2.2 EGO for constrained optimization problems: SEGO

To construct a surrogate model in an optimization context is a complex task. Many classifications exist in the literature, mainly based on the type of the metamodel and the method used to select the search points. For more details, Jones (2001) gives a taxonomy of surrogate-based optimization methods. In this paper, we focus on the so-called 'two-stage' method. The first step consists in fitting the metamodel and estimating the associated hyperparameters. The second step consists in using this metamodel instead of the true function and searching promising points following a chosen infill sampling points. The details of this approach for bound constrained optimization problems are given as follows:

1. Evaluate the initial design of experiments: a set of initial points $(\mathbf{X}, \mathbf{y}) = \left(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}, y^{(1)}, \dots, y^{(n)}\right)$ is constructed. In this paper, the latin hypercube design (Jin et al 2005) is used. Since the proposed approach is designed to optimize time-consuming engineering design problems, only a small number of initial points n is used. In addition, set $n = d + 1$ for comparison conveniences with Regis (2014).
2. Construct (or update) the metamodel involved in the optimization problem: the hyperparameters of the metamodel are fitted with the initial (or the enriched) design of experiments.
3. Optimize the infill sampling criterion.
4. Evaluate the new point $\left(\mathbf{x}^{(n+1)}, y^{(n+1)}\right)$.
5. Check if the number of iterations is reached: if the number of points permitted is reached (stopping criterion), the algorithm stops, otherwise, the new point is added in the design of experiments and the algorithm returns to the step 2.

The EI sampling criterion is one of the most popular infill criteria and is used to balance local and global search (Jones et al 1998; Forrester et al 2008). This algorithm performs well when the number of dimensions is relatively low. However, its cannot effectively deal with high-dimensional problems, this is mainly due to the uncertainty of the model, which is often high when the number of input variables is large. Indeed, the EI criterion puts more emphasis on metamodel exploration rather than on its exploitation, because it is almost impossible to fill all regions of the domain in this case. For this reason, the criterion used by Sasena (WB2) “to locate the regional extreme” is chosen in this paper (for more details, see Sasena (2002)). The WB2 criterion is essentially the predicted function value added to the EI function and is, therefore, a slightly more local searching criterion. The expression of the WB2 criterion is given by

$$\mathrm{WB2}(\mathbf{x}) = \begin{cases} -\hat{y}(\mathbf{x}) + \overbrace{(f_{\min} - \hat{y}(\mathbf{x}))\Phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right) + s(\mathbf{x})\phi\left(\frac{f_{\min} - \hat{y}(\mathbf{x})}{s(\mathbf{x})}\right)}^{\mathrm{EI}(\mathbf{x})}, & \text{if } s < 0 \\ -\hat{y}(\mathbf{x}), & \text{if } s = 0 \end{cases} \tag{16}$$

The WB2 is quite similar to the EI and needs to be maximized. The only difference is in the additional first term $(-\hat{y}(\mathbf{x}))$ on the right-hand side of Equation (16). The main advantage of this criterion is that it is smoother than the EI function since it does not return to zero at the sampled points. After comparing the WB2 and the EI criteria on several analytical functions, Sasena (2002) recommends using the WB2 criterion instead of the EI criterion. In addition, the WB2 criterion exploits the surrogate model more than the EI criterion. This is more suitable for high-dimensional problems since the budget allocated is usually low and not proportional to the number of dimensions. Therefore, an algorithm needs to rapidly converge towards a promising solution while maintaining its exploration behavior.

To extend the algorithm for constrained optimization problems, a KPLS(+K) model for each constraint function is constructed, and the constrained infill sampling criterion is defined by

$$\min_{\mathbf{x}\in B} \begin{cases} \mathrm{WB2}(\mathbf{x}) \\ \text{s.t.} \\ \hat{g}_k(\mathbf{x}) \leq 0,\ k = 1,\dots,m, \end{cases} \tag{17}$$

where $\hat{g}_k(\mathbf{x})$, for $k = 1,\dots,m$ are the predictions of $g_k(\mathbf{x})$ given by Equation (15).

This approach will be called SEGOKPLS if the KPLS models are used, otherwise SEGOKPLS+K when the KPLS+K models are used. The final procedure is summarized on Figure 1:

1. Evaluate of the initial design of experiments:
   $(\mathbf{X}, \mathbf{y}, \mathbf{g}_1, \dots, \mathbf{g}_m) = \left(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}, y^{(1)}, \dots, y^{(n)}, g_1^{(1)}, \dots, g_1^{(n)}, \dots, g_m^{(1)}, \dots, g_m^{(n)}\right)$.
2. Construct or update of the KPLS(+K) models: the parameters of the metamodel are fitted with the initial or the enriched design of experiments.
3. Maximize Equation (16).
4. Evaluate the new point: the point solution is evaluated
$$\left(\mathbf{x}^{(n+1)}, y^{(n+1)}, g_1^{(n+1)}, \dots, g_m^{(n+1)}\right).$$
5. Check if the number of iterations is reached: if the number of points permitted is reached (stopping criterion), the algorithm stops, otherwise, the new point is added in the design of experiments and the algorithm returns to the step 2.

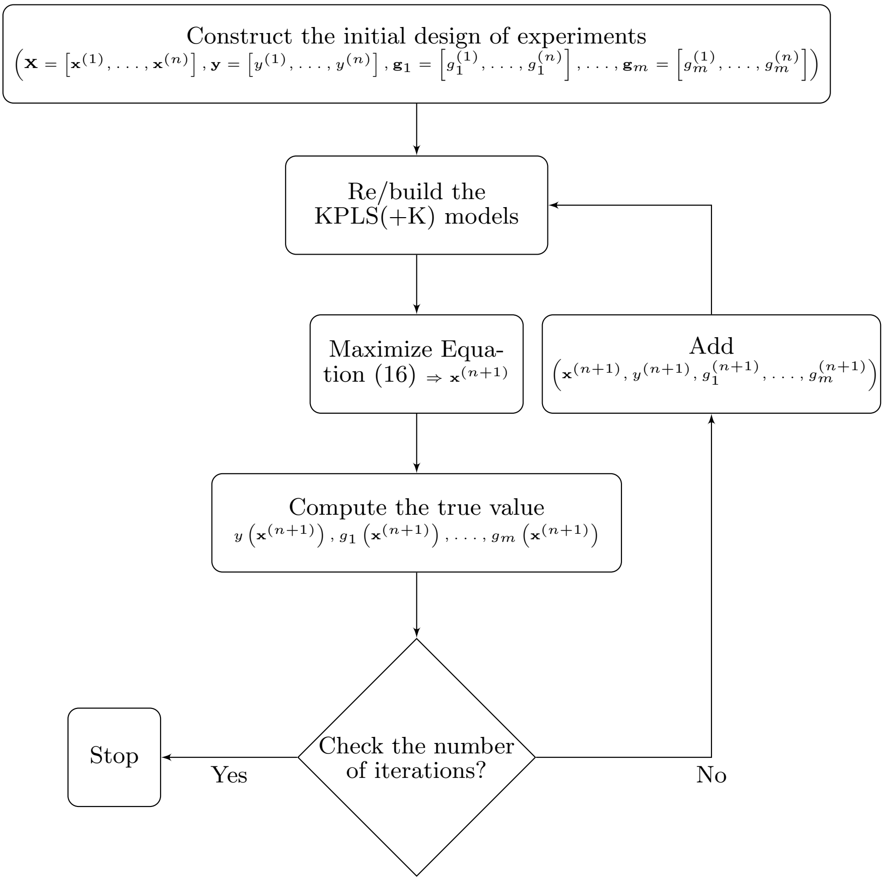

Construct the initial design of experiments
$\left(\mathbf{X} = \left[\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\right], \mathbf{y} = \left[y^{(1)}, \ldots, y^{(n)}\right], \mathbf{g}_1 = \left[g_1^{(1)}, \ldots, g_1^{(n)}\right], \ldots, \mathbf{g}_m = \left[g_m^{(1)}, \ldots, g_m^{(n)}\right]\right)$
Re/build the KPLS(+K) models
Maximize Equation (16) $\Rightarrow \mathbf{x}^{(n+1)}$
Add $\left(\mathbf{x}^{(n+1)}, y^{(n+1)}, g_1^{(n+1)}, \ldots, g_m^{(n+1)}\right)$
Compute the true value $y\left(\mathbf{x}^{(n+1)}\right), g_1\left(\mathbf{x}^{(n+1)}\right), \ldots, g_m\left(\mathbf{x}^{(n+1)}\right)$
Check the number of iterations?
Yes
No
Stop

Fig. 1: Steps used for the SEGOKPLS(+K) methods.

3 Analytical functions

To demonstrate the performance of the SEGOKPLS algorithm, 12 well known analytical functions, defined in Table A.1 in Appendix A (included in an online supplement), are used. The SEGOKPLS algorithm is compared to several optimization methods that have been applied to these analytical functions in Regis (2014).

It should be noted that the original g_{03} and g_{05} functions have equality constraints in their definition. This paper considers the same modified formulations used by Regis (2014) (noted by G3MOD and G5MOD in Regis (2014)) where each constraint function is replaced by an inequality ($\leq$) constraint.

3.1 Alternative optimization methods

We compare SEGOKPLS with COBRA (Constrained Optimization By RAdial basis function interpolation) (Regis 2014) and ConstrLMSRBF (Constrained Local Metric Stochastic Radial Basis Function) (Regis 2011). Both these methods are compared with several alternative optimization methods existing in the literature by Regis (2014):

– SDPEN for sequential penalty Derivative-free method for nonlinear constrained optimization (Liuzzi et al 2010),
– NOMAD for nonlinear optimization by mesh adaptive direct search that uses DACE surrogates (NOMADm-DACE) (Abramson and Audet 2006; Audet and Jr. 2006),
– GLOBALm algorithm (Sendin and Banga 2009) which is a multistart clustering algorithm and using two types of local solvers,
– OQNLP (Ugray et al 2007), which is implemented as the GlobalSearch solver in the Matlab Global Optimization toolbox (the Mathworks 2010).

In this paper, only the best result, given by Regis (2014), are used for the comparison with the SEGOKPLS algorithm as detailed in the following Section.

3.2 Experimental setup

The SEGOKPLS computation is performed using the Python toolbox "Scikit-learn v.014" (Pedregosa et al 2011) on an Intel(R) Core(TM) i7-4500U CPU @ 1.80 Hz 2.40 GHz. The results of alternative optimization methods are transcribed from Regis (2014) and they are obtained using an Intel(R) Core(TM) i7 CPU 860 2.8GHZ. To increase the reliability of the comparison between the results of the SEGOKPLS algorithm and the alternative optimization methods, the conditions in which alternative optimization methods are performed are closely approximated. To achieve these conditions, a new latin hypercube design of size $d+1$ is used (since the initial design of experiments used by Regis (2014) are not available but the same size is used) in which all points are infeasible. For each experiment, the SEGOKPLS algorithm is run 30 times and the computation budget is fixed at 100 objective and constraint function evaluations. These several runs reduce the influence of the initial design of experiments on the final results and on the performance of each optimization method involved into the comparison. The number of principal components used into the KPLS models for the SEGOKPLS algorithm is equal to 3. The constraint tolerance used in these tests is 10^{-5}.

Five criteria, to achieve the comparison, are used: the best solution, the worst solution, the median, the mean and the standard deviation error (std) of the 30 trials are computed. To facilitate the comparison of the results for each case test, only the best result of each statistic (e.g., the worst solution, ...) are transcribed from Table B3 in Regis (2014) . These results used at least one feasible point in the initial design of experiments because this is required by the alternative methods. However, the SEGOKPLS algorithm does not require an initial feasible point. In addition, searching for an initial feasible point is out of the scope of this paper. Therefore, as mentioned above, the initial design of experiments was intentionally chosen to be free from feasible points, which is a common situation in real engineering design.

As mentioned in Bouhlel et al (2016a), the KPLS+K model is more costly than the KPLS model; e.g. KPLS is over twice time faster than KPLS+K for a function with 60 input variables and 300 sampling points. For the analytical functions, 30 trials are used, so SEGOKPLS is compared only to alternative methods to reduce the cost of the experimental tests. A comparison is done between SEGOKPLS and SEGOKPLS+K on the MOPTA08-12D using 5 runs in section 4.1.

3.3 Results on the analytical functions

Table C.2 in the Appendix C (included in an online supplement) provides statistics for both results of the SEGOKPLS algorithm (first value written in brackets) and best results of the alternative algorithms (second value written in brackets) given by Regis (2014). The SEGOKPLS algorithm yields better results compared with the best algorithm by Regis (2014) for many test cases. In particular, SEGOKPLS is better than the alternative algorithms for all functions in terms of the worst, the median and the mean, except for the median of functions g_{03}, g_{10} and GTCD4. This result indicates that the SEGOKPLS algorithm is not very sensitive to the quality of the initial latin hypercube design. Moreover, the SEGOKPLS algorithm finds a better result than alternative algorithms, in terms of the best solution, for both g_{03} and g_{07} functions. In addition, the SEGOKPLS algorithm reaches the best-known solution over all statistics used in the comparison for 5 functions (SR7, Hesse, g_{04}, g_{05} and g_{07}).

For all the test functions, SEGOKPLS returns the best possible optimum in the worst-case scenario of those 30 trials considered. Since the initial distribution of the sampling points plays an important role into an optimization design process that can affect the final solution, the last result (best of worst cases) shows that SEGOKPLS' solution is less deteriorated than the alternative methods in the worst cases, that is an important characteristic in real-world engineering design optimization. Indeed, it is almost impossible to know in advance the best type and distribution of an initial design of experiments for a certain engineering problems.

These results show the capability of SEGOKPLS to compete with optimization algorithms from the literature. The application of this algorithm to a more realistic problem is considered in the following section with the MOPTA08 test case.

4 MOPTA08 test problems

MOPTA08 is an automotive problem (Jones 2008) available as a Fortran code at "http://www.miguelanjos.com/jones-benchmark". It is an optimization problem for minimizing the mass of a vehicle (1 objective function) under 68 inequality constraints which are well normalized. One run of a real simulation of the MOPTA08 problem takes about 1 to 3 days. However, one simulation with the Fortran code is immediate since it is a “good” approximation of the real version using the Kriging models. Nevertheless, the type of Kriging used is not available. The MOPTA08 problem contains 124 input variables normalized to $[0, 1]$. Thus, this problem is considered large-scale in the area of a surrogate-based expensive black-box optimization.

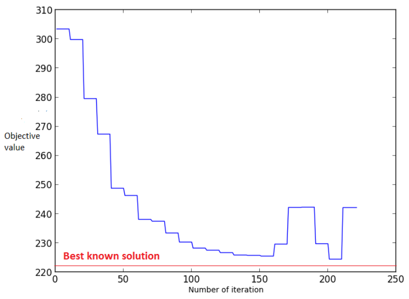
For this test case, the SEGOKPLS+K algorithm is first applied on the complete optimization problem, i.e., MOPTA08 problem with 124 input variables. Despite the good performance of SEGOKPLS+K during the first optimization iterations, serious numerical problems, that the most important of them are explained in details in Section 4.1, are encountered. Next, the SEGOKPLS+K algorithm is applied to two smaller problems: MOPTA08 with 12 and 50 input variables.

4.1 Why reduce the complexity of the MOPTA08 problem?

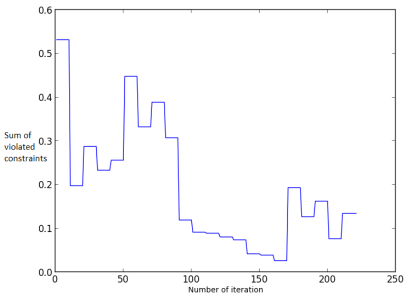
When applying the SEGOKPLS+K algorithm on the complete MOPTA08 problem, an initial latin hypercube design of size $d + 1$ (125 points) and 221 iterations are considered. The number of iterations is limited because there is a system crash after the 221^{th} iteration. In order to understand the reasons of a such system crash, consider the mean evolution of both the objective function and the sum of violated constraints occurring during 10 successive iterations, which are shown in Figure 2. The objective function value decreases during the first 100 iterations as shown in Figure 2a. However, it starts to oscillate at the 165^{th} iteration. Indeed, the SEGOKPLS+K algorithm focuses its search in a local region. Moreover, it adds several similar points (i.e., points that are nearly collinear) to the training points. This set of new points, as a consequence, leads to an ill-conditioned correlation matrix. Therefore, the algorithm crashes after a certain number of iterations, in particular when the number of dimensions d is very high (>50). In order to reduce the complexity of the problem, the number of input variables involved in the optimization problem is reduced. To achieve this, certain of the input variables are fixed to the best-known solution found by Regis (2014), which its objective function is 222.22.

In this paper, two reduced problems are treated: 12 and 50 input variables given in Sections 4.4.1 and 4.4.2, respectively. The indexes of the variables used during the optimization are given in Appendix D (included in an online supplement). The same ill-conditioning problem that occurs with 124 input variables are encountered beyond 50 input variables.

In addition, the SEGOKPLS algorithm is not included in the comparison. Indeed, the Table 1 shows the results of SEGOKPLS and SEGOKPLS+K on the MOPTA08-12D case for 5 trials with a constraint violation of 10^{-5}. The SEGOKPLS+K algorithm outperforms SEGOKPLS, and the best known solution of the problem is reached with less iterations using SEGOKPLS+K. Therefore, only SEGOKPLS+K is used in the following MOPTA08 study.



(a) Mean of the objective function for each 10 iterations.



(b) Mean of the sum of violated constraints for each 10 iterations.

Fig. 2: MOPTA08 test case with 124 input variables. Left panel: Mean of the best feasible objective occurring for 10 successive iterations. Right panel: Mean of the sum of violated constraints occurring for 10 successive iterations.

Table 1: MOPTA08-12D - Total number of calls (value for each of the 5 runs) of the black box function to reach the optimum with SEGOKPLS and SEGOKPLS+K approaches.

MOPTA08-12D	SEGOKPLS	SEGOKPLS+K
Trial 1	53	25
Trial 2	52	34
Trial 3	52	28
Trial 4	36	29
Trial 5	38	32

4.2 Alternative optimization methods

4.2.1 Description of the COBYLA algorithm

In this paper, the baseline method used is the so-called COBYLA method (Powell 1994). COBYLA is a derivative-free trust region method that uses an approximating linear interpolation models of the objective and constraint functions. The algorithm is written in Fortran and iteratively finds a candidate for the optimal solution of an approximating linear programming problem. Using the original objective and constraint functions, the candidate solution is evaluated at each iteration of optimization. Afterwards, it is used to improve the approximating linear programming problem while maintaining a regular shaped simplex over iterations. When the solution can no longer be improved, the step size is reduced until it becomes sufficiently small, and then the algorithm terminates.

4.2.2 Description of the COBRA algorithm

SEGOKPLS+K will be compared with an algorithm called COBRA (Regis 2014) that is designed for constrained black-box optimization when the objective and constraint functions are computationally expensive. Like SEGOKPLS+K, COBRA treats each inequality constraint individually instead of combining them into one penalty function. However, instead of using Kriging, it uses RBF surrogates to approximate the objective and constraint functions. Moreover, COBRA implements a two-phase approach where Phase I finds a feasible point while Phase II searches for a better feasible point.

In each iteration of Phase II of COBRA, RBF surrogates for the objective function and for each of the constraint functions are fit. Then, the iterate is chosen to be a minimizer of the RBF surrogate of the objective function that satisfies RBF surrogates of the constraints within some small margin and that also satisfies a distance requirement from previous iterates. As explained in Regis (2014), the margin is useful because it helps maintain feasibility of the iterates, especially on problems with many inequality constraints. More precisely, the next sample point x_{n+1} as a solution to the optimization subproblem:

$$\begin{aligned} \min\ & \hat{y}(\mathbf{x}) \\ \text{s.t.}\ & \mathbf{x} \in B \\ \hat{g}_i(\mathbf{x}) + \epsilon_n^{(i)} &\leq 0,\ i = 1, 2, \ldots, m \\ \|\mathbf{x} - \mathbf{x}^{(j)}\| &\geq \rho_n,\ j = 1, \ldots, n \end{aligned}$$

where $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ are the previous sample points, $\hat{y}, \hat{g}_1, \ldots, \hat{g}_m$ are the RBF surrogates for the objective function $f(x)$ and constraint functions $g_1(x), \ldots, g_m(x)$, respectively. Moreover, $\epsilon_n^{(i)} > 0$ is the margin for the inequality constraints, which varies depending on performance starting with a value of $0.005\ell(B)$, where $\ell(B)$ is the length of one side of the hypercube B. Also, ρ_n is the distance requirement from previous sample points and it is allowed to cycle to balance local and global search. That is, $\rho_n = \gamma_n \ell(B)$, where γ_n is an element of the cycle of distance requirements Ξ. Here, SEGOKPLS+K is compared with an implementation of COBRA that emphasizes local search since it performed better on the benchmark test problems used in (Regis 2014). In this implementation of COBRA that focuses on local search, $\Xi = \langle 0.01, 0.001, 0.0005 \rangle$.

4.3 Experimental setup

The SEGOKPLS+K computations are performed with the Python toolbox "Scikit-learn v.014" (Pedregosa et al 2011) using an Intel(R) Core(TM) i7-4500U CPU @ 1.80 Hz 2.40 GHz desktop machine. The alternative optimization methods are performed in Matlab 7.11.0 using an Intel(R) Core(TM) i7 CPU 860 2.8 Ghz desktop machine. This is one of the reasons that the computational cost was not investigated, in addition to the non-practical environment for getting such a consistent study. Bouhlel et al (2016a) provide a comparison of CPU time between KPLS and KPLS+K using different

Table 2: MOPTA08-12D - Total number of calls, the initial design of experiments (13 points) + number of iterations, for the 5 runs to reach the optimum with the COBYLA, COBRA and SEGOKPLS+K methods. The objective value of the best feasible point is given in brackets.

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
COBYLA	362	513	513	424	513
	(222.22)	(222.26)	(223.06)	(222.22)	(236.96)
COBRA	137	176	194	143	206
	(222.22)	(222.22)	(222.22)	(222.22)	(222.22)
SEGOKPLS+K	28	42	51	36	52
	(222.22)	(222.22)	(222.22)	(222.22)	(222.22)

number of sampling points and input variables, that gives an idea about the total cost of the optimization design. The optimization of the Equation (17) requires between 10 and 20 minutes per iteration.

For each set of experiments, each algorithm is run 5 times, and each run corresponds to a different latin hypercube design of size $d+1$. In addition, no any feasible point is included into the different initial samples. For the SEGOKPLS+K algorithm, a computational budget of 39 and 63 evaluations is fixed for MOPTA08 with 12 and 50 input variables, respectively. Each evaluation corresponds to 1 objective and constraint functions evaluation. On the other hand, a computational budget of 500 evaluations is fixed for both COBYLA and COBRA algorithms. Then, 3 principal components are fixed to build the KPLS+K models for all outputs. Finally, a constraint tolerance of 10^{-5} is used as for the analytical problems.

4.4 Results of the MOPTA08 problem

4.4.1 Comparison on the MOPTA08 benchmark with 12 dimensions

The objective in this section is to optimize the MOPTA08 problem in 12 dimensions and compare the three algorithms (COBYLA, COBRA and SEGOKPLS+K), the best-known solution of this weight minimization being 222.23.

Table 2 gives the total number of calls and the objective value of the best feasible point from the 5 runs performed. In this previous example, only 36 (59 - 13) iterations of optimization are sufficient for SEGOKPLS+K to find the best-known solution of the problem for all runs. Unlike the SEGOKPLS+K algorithm, the COBYLA and COBRA algorithms require more than 100 iterations of optimization to get close to the best-known solution. However, COBRA performs reasonably well on MOPTA08-12D. In fact, the COBRA algorithm requires 293 iterations in the worst case for finding the best-known solution, whereas, COBYLA finds the best-known solution for only two runs after 411 iterations in the worst case. Moreover, two runs converge to solutions with objective values 223.06 and 222.26. In addition, the COBYLA

Table 3: MOPTA08-50D - Total number of calls, the initial design of experiments (51 points) + number of iterations, for the 5 runs to reach the optimum with the COBYLA, COBRA and SEGOKPLS+K methods. The objective value of the best feasible point is given in brackets.

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
COBYLA	551	551	551	551	551
	–	–	–	–	–
COBRA	551	449	551	551	551
	(223.89)	(222.22)	(222.27)	(222.52)	(222.96)
SEGOKPLS+K	101	91	114	93	109
	(222.22)	(222.22)	(222.22)	(222.22)	(222.22)

algorithm fails to find the best-known solution for 1 run and remains on the objective value 236.96 with a sum of violated functions equals to 6.77.

Hence, for this relatively low-dimensional case, the SEGOKPLS+K algorithm shows a very good performance and seems to be a very competitive method among optimization methods existing in the literature.

4.4.2 Comparison on the MOPTA08 benchmark with 50 dimensions

To assess the performance of SEGOKPLS+K optimizer on a higher dimensional problem, the same test case with 50 design variables is considered. All optimization algorithms search to reach the best-known solution starting from different Latin hypercube designs of size 51. As for 12D test case, Table 3 gives the total number of calls and the objective value of the best feasible point from the 5 runs performed. In this case, 63 (114 - 51) iterations of optimization for SEGOKPLS+K are used whereas 500 iterations are used for both COBYLA and COBRA. In fact, 4 runs rapidly converge near the best-known solution before the 23th iteration. Only 1 run remains stable on the objective function value 261.33 during 49 iterations of optimization before converging to the best-known solution, which costs more iterations before converging. This leads to fixing 63 iterations of optimization for finding the best-known solution that is 222.22. Indeed, the total number of calls to the expensive black-box function (size of the initial design of experiments + number of iterations of optimization) is equal to 114, which is greater than $2d$. To optimize a function of 50 dimensions and 68 constraint functions using that total budget is an important improvement in the area of surrogate-based optimization, especially for high-dimensional and expensive black-box optimization problems.

The COBYLA algorithm does not perform well on the MOPTA08-50D problem. In fact, the solutions found by COBYLA after the 500th iteration from all runs do not satisfy all the constraints. The sums of constraint violations of the solutions from all runs are equal to 0.007, 0.35, 3.76, 6.76 and 8.72 and the corresponding objective values are equal to 238.15, 247.42, 271.68, 286.67 and 274.74, respectively. Thus, COBYLA is not really designed for high-dimensional expensive black-box functions.

For all runs, COBRA reaches feasible solutions detailed as follows: after the 500^{th} iteration, the objective values of 4 runs are equal to 222.52, 222.89, 222.96 and 223.27, and the remaining run finds the best-known solution after 398 iterations.

To summarize the comparisons of this test case, the optimization with the SEGOKPLS+K algorithm can reach the best-known solution for all runs. This means that SEGOKPLS+K is less sensitive to the distribution of sampling points than the alternative methods. Furthermore, SEGOKPLS+K rapidly finds the solution using very few calls to the expensive black-box functions. Therefore, SEGOKPLS+K is a very competitive optimization algorithm among existing optimization algorithms.

5 Conclusions

This paper introduced the SEGOKPLS and SEGOKPLS+K algorithms adapted to high-dimensional constrained surrogate based optimization methods using expensive black-box functions. Both combine the iterative optimization algorithm SEGO and the KPLS(+K) models. Moreover, SEGOKPLS(+K) aim to rapidly handle high-dimensional optimization problems through a good management of the budget available, which is represented by a number of calls of the expensive true functions. The key improvement of this method is the use of the KPLS(+K) models and the local infill sampling criterion WB2, which are suitable for high-dimensional problems (up to 50 dimensions). Furthermore, note that the SEGOKPLS(+K) algorithms treat each constraint individually instead of gathering them into one penalty function.

Analytical test cases and an automotive test problem are used for comparing the SEGOKPLS(+K) algorithms with many existing optimization methods. SEGOKPLS(+K) outperform the alternative methods including COBRA and COBYLA, in particular on the MOPTA-12D and MOPTA-50D problems. Moreover, SEGOKPLS+K is able to reach the best-known solution for the MOPTA-12D and MOPTA-50D problems in much less expensive calls than either COBYLA or COBRA.

An interesting direction for future work is to use classical preconditioning methods to handle problems with more than 50 dimensions, or to require SEGOKPLS(+K) to respect some conditions for adding new points into the design of experiments to avoid a bad conditioning of the covariance matrix.

References

Abramson MA, Audet C (2006) Convergence of mesh adaptive direct search to second-order stationary points. SIAM Journal on Optimization 17(2):606–619, DOI 10.1137/050638382, URL `http://dx.doi.org/10.1137/050638382`

Alberto PR, González FG (2012) Partial Least Squares Regression on Symmetric Positive-Definite Matrices. Revista Colombiana de Estadística 36(1):177–192

Audet C, Jr DJE (2006) Mesh adaptive direct search algorithms for constrained optimization. SIAM Journal on Optimization 17(1):188–217, DOI 10.1137/040603371, URL `http://dx.doi.org/10.1137/040603371`

Bouhlel MA, Bartoli N, Morlier J, Otsmane A (2016a) An Improved Approach for Estimating the Hyperparameters of the Kriging Model for High-Dimensional Problems through the Partial Least Squares Method. Mathematical Problems in Engineering vol. 2016, Article ID 6723410

Bouhlel MA, Bartoli N, Otsmane A, Morlier J (2016b) Improving Kriging Surrogates of High-Dimensional Design Models by Partial Least Squares Dimension Reduction. Structural and Multidisciplinary Optimization 53(5):935–952

Chen X, Diez M, Kandasamy M, Zhang Z, Campana EF, Stern F (2015) High-fidelity global optimization of shape design by dimensionality reduction, metamodels and deterministic particle swarm. Engineering Optimization 47(4):473–494, DOI 10.1080/0305215X.2014.895340, URL `http://dx.doi.org/10.1080/0305215X.2014.895340`, `http://dx.doi.org/10.1080/0305215X.2014.895340`

Chevalier C, Ginsbourger D (2013) Fast Computation of the Multi-Points Expected Improvement with Applications in Batch Selection. Springer Berlin Heidelberg, Berlin, Heidelberg

Conn AR, Scheinberg K, Vicente LN (2009) Introduction to Derivative-Free Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA

Cressie N (1988) Spatial Prediction and Ordinary Kriging. Mathematical Geology 20(4):405–421

Feliot P, Bect J, Vazquez E (2016) A bayesian approach to constrained single- and multi-objective optimization. Journal of Global Optimization pp 1–37, DOI 10.1007/s10898-016-0427-3, URL `http://dx.doi.org/10.1007/s10898-016-0427-3`

Forrester AIJ, Sóbester A, Keane AJ (2006) Optimization with missing data. Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 462(2067):935–945

Forrester AIJ, Sóbester A, Keane AJ (2008) Engineering Design via Surrogate Modeling: A Practical Guide. Wiley

Frank IE, Friedman JH (1993) A Statistical View of Some Chemometrics Regression Tools. Technometrics 35:109–148

Goovaerts P (1997) Geostatistics for Natural Resources Evaluation (Applied Geostatistics). Oxford University Press, New York

Gramacy RB, Gray GA, Le Digabel S, Lee HKH, Ranjan P, Wells G, Wild SM (2016) Modeling an Augmented Lagrangian for Blackbox Constrained Optimization. Technometrics 58(1):1–11, DOI 10.1080/00401706.2015.1014065, URL `http://dx.doi.org/10.1080/00401706.2015.1014065`

Haftka R, Villanueva D, Chaudhuri A (2016) Parallel surrogate-assisted global optimization with expensive functions–a survey. Structural and Multidisciplinary Optimization pp 1–11

Helland IS (1988) On Structure of Partial Least Squares Regression. Communication in Statistics - Simulation and Computation 17:581–607

Hickernell FJ, Yuan YX (1997) A simple multistart algorithm for global optimization

Jin R, Chen W, Sudjianto A (2005) An efficient algorithm for constructing optimal design of computer experiments. Journal of Statistical Planning and Inference 134(1):268–287

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

Jones DR (2008) Large-scale multi-disciplinary mass optimization in the auto-industry. presented at the *Modeling and Optimization: Theory and Application (MOPTA) 2008 Conference*. Ontario, Canada

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

Kennedy J, Eberhart R (1995) Particle swarm optimization. vol 4, pp 1942–1948

Koehler JR, Owen AB (1996) Computer experiments. Handbook of Statistics pp 261–308

Krige DG (1951) A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand. Journal of the Chemical, Metallurgical and Mining Society 52:119–139

Kyriacou SA, Asouti VG, Giannakoglou KC (2014) Efficient pca-driven eas and metamodel-assisted eas, with applications in turbomachinery. Engineering Optimization 46(7):895–911, DOI 10.1080/0305215X.2013.812726, URL `http://dx.doi.org/10.1080/0305215X.2013.812726`, `http://dx.doi.org/10.1080/0305215X.2013.812726`

Laarhoven PJM, Aarts EHL (eds) (1987) Simulated Annealing: Theory and Applications. Kluwer Academic Publishers, Norwell, MA, USA

Li Y, Wu Y, Zhao J, Chen L (2016) A Kriging-based constrained global optimization algorithm for expensive black-box functions with infeasible initial points. Journal of Global Optimization pp 1–24, DOI 10.1007/s10898-016-0455-z

Liem RP, Mader CA, Martins JRRA (2015) Surrogate models and mixtures of experts in aerodynamic performance prediction for aircraft mission analysis. Aerospace Science and Technology 43:126–151, 10.1016/j.ast.2015.02.019

Liu H, Xu S, Chen X, Wang X, Ma Q (2016) Constrained global optimization via a direct-type constraint-handling technique and an adaptive metamodeling strategy. Structural and Multidisciplinary Optimization pp 1–23, DOI 10.1007/s00158-016-1482-6, URL `http://dx.doi.org/10.1007/s00158-016-1482-6`

Liuzzi G, Lucidi S, Sciandrone M (2010) Sequential penalty derivative-free methods for nonlinear constrained optimization. SIAM Journal on Optimization 20(5):2614–2635

Ollar J, Mortished C, Jones R, Sienz J, Toropov V (2016) Gradient based hyper-parameter optimisation for well conditioned kriging metamodels. Structural and Multidisciplinary Optimization pp 1–16

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Rettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12:2825–2830

Picheny V, Ginsbourger D, Roustant O, Haftka RT, Kim NH (2010) Adaptive Designs of Experiments for Accurate Approximation of a Target Region

Powell M (1994) A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Advances in optimization and numerical analysis: Proceedings of the Sixth Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico, Kluwer Academic Pub, p 51

Regis R (2011) Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. Computers and Operations Research 38(5):837–853

Regis RG (2014) Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. Engineering Optimization 46(2):218–243

Regis RG, Wild SM (2017) CONORBIT: Constrained optimization by radial basis function interpolation in trust regions. Optimization Methods and Software 32(3):552–580

Sacks J, Welch WJ, Mitchell WJ, Wynn HP (1989) Design and Analysis of Computer Experiments. Statistical Science 4(4):409–435

Sasena M, Papalambros P, Goovaerts P (2002) Exploration of metamodeling sampling criteria for constrained global optimization. Engineering optimization 34(3):263–278

Sasena MJ (2002) Flexibility and efficiency enhancements for constrained global design optimization with Kriging approximations. PhD thesis, University of Michigan

Schonlau M (1998) Computer experiments and global optimization. PhD thesis, University of Waterloo, Canada

Sendin JOH, Banga T J Rand Csendes (2009) Extensions of a multistart clustering algorithm for constrained global optimization problems. Industrial and Engineering Chemistry Research 6(48):3014–3023

Simon D (2013) Evolutionary Optimization Algorithms. Wiley

Singh P, Couckuyt I, Ferranti F, Dhaene T (2014) A constrained multi-objective surrogate-based optimization algorithm. In: 2014 IEEE Congress on Evolution-

ary Computation (CEC), pp 3080–3087, DOI 10.1109/CEC.2014.6900581
Ugray Z, Lasdon L, Plummer J, Glover F, Kelly J, Martí R (2007) Scatter search and local nlp solvers: A multistart framework for global optimization. INFORMS J on Computing 19(3):328–340, DOI 10.1287/ijoc.1060.0175, URL http://dx.doi.org/10.1287/ijoc.1060.0175
Watson AG, Barnes RJ (1995) Infill sampling criteria to locate extremes. Mathematical Geology 27(5):589–608
Wold H (1966) Estimation of Principal Components and Related Models by Iterative Least squares, Academic Press, New York, pp 391–420