Optim Eng (2016) 17:177–203
DOI 10.1007/s11081-015-9281-2

MISO: mixed-integer surrogate optimization framework

Juliane Müller[1]

Received: 12 August 2014 / Revised: 5 January 2015 / Accepted: 7 May 2015 /
Published online: 20 June 2015

Abstract We introduce MISO, the mixed-integer surrogate optimization framework. MISO aims at solving computationally expensive black-box optimization problems with mixed-integer variables. This type of optimization problem is encountered in many applications for which time consuming simulation codes must be run in order to obtain an objective function value. Examples include optimal reliability design and structural optimization. A single objective function evaluation may take from several minutes to hours or even days. Thus, only very few objective function evaluations are allowable during the optimization. The development of algorithms for this type of optimization problems has, however, rarely been addressed in the literature. Because the objective function is black-box, derivatives are not available and numerically approximating the derivatives requires a prohibitively large number of function evaluations. Therefore, we use computationally cheap surrogate models to approximate the expensive objective function and to decide at which points in the variable domain the expensive objective function should be evaluated. We develop a general surrogate model framework and show how sampling strategies of well-known surrogate model algorithms for continuous optimization can be modified for mixed-integer variables. We introduce two new algorithms that combine different sampling strategies and local search to obtain high-accuracy solutions. We compare MISO in numerical experiments to a genetic algorithm, NOMAD version 3.6.2, and SO-MI. The results show that MISO is in general more efficient than NOMAD and the genetic algorithm with respect to

Electronic supplementary material The online version of this article (doi:10.1007/s11081-015-9281-2) contains supplementary material, which is available to authorized users.

✉ Juliane Müller
juliane.mueller2901@gmail.com

[1] Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

finding improved solutions within a limited budget of allowable evaluations. The performance of MISO depends on the chosen sampling strategy. The MISO algorithm that combines a coordinate perturbation search with a target value strategy and a local search performs best among all algorithms.

Keywords Mixed-integer optimization · Surrogate models · Black-box optimization · Radial basis functions · Derivative-free · Global optimization

Abbreviations

DYCORS	DYnamic COordinate search using response surface models (Regis and Shoemaker 2013)
EGO	Efficient global optimization (Jones et al. 1998)
MISO	Mixed-integer surrogate optimization
MISO-CP	MISO-coordinate perturbation
MISO-EI	MISO-expected improvement
MISO-RS	MISO-random sampling
MISO-SM	MISO-surface minimum
MISO-TV	MISO-target value
MISO-CPTV	MISO with combination of CP and TV
MISO-CPTV-local	MISO with combination of CP, TV, and a local search
MISO-CPTV-l(f)	MISO-CPTV-local that uses fmincon as local optimizer
MISO-CPTV-l(o)	MISO-CPTV-local that uses ORBIT (Wild et al. 2007) as local optimizer
NOMAD	Nonlinear optimization by mesh-adaptive direct search (Le Digabel 2011), version 3.6.2
RBF	Radial basis function
SO-MI	Surrogate optimization-mixed integer (Müller et al. 2013b)
SRBF	Stochastic radial basis function algorithm (Regis and Shoemaker 2007)
$f(\cdot)$	Computationally expensive objective function
d	Problem dimension
d_1	Number of integer variables
d_2	Number of continuous variables
$\mathbf{z}$	Variable vector
z_i^l, z_i^u	Lower and upper variable bounds of the ith variable
$\mathcal{Z}$	Set of evaluated points, $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$
$\mathcal{Z}_l$	Set of points evaluated during the local search (l-step)
n_0	Number of points in the initial experimental design
n_1	Number of function evaluations done in the local search (l-step)
n	Number of already evaluated points
$s_n(\cdot)$	Surrogate model fit to n data points
$\mathbb{I}$	Indices of the integer variables

1 Introduction and motivation

In engineering optimization problems, evaluating the objective function often requires a computationally expensive computer simulation that approximates the physical behavior of the system under consideration. These simulation models are black-box, and thus the analytical description and derivatives are not available. Automatic differentiation is in many cases not applicable due to confidentiality restrictions of the simulation codes. Numerical differentiation requires many computationally expensive objective function evaluations and is therefore inefficient. Thus, derivative-free methods (Conn et al. 2009) are widely used.

When optimizing computationally expensive black-box problems, the goal is to find near optimal solutions within only few (often only few hundred) expensive objective function evaluations in order to keep the optimization time acceptable. Surrogate models (also known as response surface models or metamodels) have been developed to efficiently solve this type of optimization problems (Forrester et al. 2008; Giunta et al. 1997; Glaz et al. 2008; Koziel and Leifsson 2013; Marsden et al. 2004; Simpson et al. 2001). Surrogate models are computationally cheap approximations of the expensive objective function (Booker et al. 1999). During the iterative optimization routine, the information from the surrogate model is exploited in order to select promising sample points in the variable domain. Hence, the computationally expensive objective function is evaluated only at carefully selected points, and thus near optimal solutions can be found efficiently.

Surrogate model algorithms have mainly been developed for continuous optimization problems (Gutmann 2001; Jones et al. 1998; Müller and Piché 2011; Müller and Shoemaker 2014; Regis and Shoemaker 2007, 2013; Wild et al. 2007). Only recently have surrogate model algorithms been devised for optimization problems that have integer constraints for some or all variables (Davis and Ierapetritou 2009; Holmström 2008b; Müller et al. 2013a, b; Rashid and Cetinkaya 2012) and where the integer variables may assume a large range of values rather than only binary values (Müller et al. 2013a, b). The goal of this paper is to develop an algorithm framework for computationally expensive black-box mixed-integer optimization problems where the variables are not restricted to binary values. This framework allows the choice of various surrogate models and sampling strategies and is a major contribution to the area of algorithms for mixed-integer computationally expensive global optimization. The MATLAB codes of our mixed-integer surrogate optimization (MISO) framework are available from the author upon request.

We consider optimization problems of the following form:

$$\min \quad f(\mathbf{z}) \tag{1}$$

$$\text{s.t.} -\infty < z_i^l \leq z_i \leq z_i^u < \infty, \quad i = 1, \ldots, d \tag{2}$$

$$\mathbf{z} \in \mathbb{Z}^{d_1} \times \mathbb{R}^{d_2}, \quad d_1 + d_2 = d, \tag{3}$$

where $f(\cdot)$ denotes the computationally expensive black-box objective function, and z_i^l and z_i^u denote the lower and upper bounds of variable i. We assume that the lower and upper bounds of the integer variables are integer values. d is the problem dimension, d^1 denotes the number of the integer variables, and d^2 denotes the number of continuous variables. For real world applications where a single function evaluation may require several hours or even days, only few hundred evaluations of $f(\mathbf{z})$ are allowable, and thus algorithms that are able to find a (near) optimal solution within a limited budget of function evaluations are needed.

In Sect. 2, we briefly describe different surrogate model types. We give a general surrogate model optimization algorithm description in Sect. 3 and we briefly review widely-used surrogate model algorithms for continuous optimization and the few developments for mixed-integer problems. In Sect. 4, we describe the MISO framework and show how the sampling strategies of existing continuous surrogate model algorithms can be modified for mixed-integer optimization problems. We also introduce a new memetic algorithm, i.e., a combination of local and global search, in order to find solutions of higher accuracy.

MISO differs from our previous mixed-integer algorithm SO-MI (Müller et al. 2013b). In SO-MI, sample points were generated by perturbing the variables of the best point found so far with constant perturbation probability. In MISO, we allow various sampling strategies. SO-MI does not contain a local search which makes it difficult for SO-MI to find high accuracy solutions. SO-MI can generally be considered as part of MISO and we show in the numerical experiments that we can improve upon the performance of SO-MI by using an additional local search.

In Sect. 5, we compare various algorithms that follow the MISO framework with SO-MI (Müller et al. 2013b), NOMAD version 3.6.2 (Abramson et al. 2009; Le Digabel 2011), and MATLAB’s genetic algorithm on a set of benchmark problems and applications arising in reliability-redundancy optimization and structural design optimization. We show that the algorithms following the MISO framework are more efficient when the goal is to find good solutions within a limited number of function evaluations. Sect. 6 concludes the paper.

2 Surrogate models

Various surrogate model types have been used in the literature within optimization frameworks. Radial basis functions (RBFs) (Gutmann 2001; Müller et al. 2013b; Powell 1992; Regis and Shoemaker 2007, 2009; Wild and Shoemaker 2013) and kriging (Davis and Ierapetritou 2009; Forrester et al. 2008; Jones et al. 1998; Simpson et al. 2001) are interpolating models, whereas polynomial regression models (Myers and Montgomery 1995) and multivariate adaptive regression splines (Friedman 1991) are non-interpolating. Moreover, there are mixture models (also known as ensemble models) that exploit information from several different surrogate model types (Goel et al. 2007; Müller and Piché 2011; Müller and Shoemaker 2014; Viana et al. 2009).

Although in general any type of surrogate model (ensemble) can be used within our MISO framework, we focus here on RBFs because they have been shown most successful in comparison to other surrogate model types (Müller and Shoemaker 2014). An RBF interpolant is defined as follows:

$$s(\mathbf{z}) = \sum_{\iota=1}^{n} \lambda_\iota \phi(\|\mathbf{z} - \mathbf{z}_\iota\|) + p(\mathbf{z}), \tag{4}$$

where $\phi(\cdot)$ is a radial basis function (here we use the cubic function $\phi(r) = r^3$ because Wild and Shoemaker (2013) showed that it performs better than other types), $\mathbf{z}_\iota, \iota = 1,\ldots,n$, denotes the points at which the objective function value is known (already sampled points), and $p(\cdot)$ denotes the polynomial tail whose order depends on the chosen RBF type (for the cubic RBF we need at least a linear polynomial tail $p(\mathbf{z}) = a + \mathbf{b}^T\mathbf{z}$). The parameters $\lambda_\iota \in \mathbb{R}, \iota = 1,\ldots,n$, and the parameters $a \in \mathbb{R}$ and $\mathbf{b} = [b_1,\ldots,b_d]^T \in \mathbb{R}^d$ are determined by solving the following linear system of equations

$$\begin{bmatrix} \boldsymbol{\Phi} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \end{bmatrix}, \tag{5}$$

where $\Phi_{\iota\nu} = \phi(\|\mathbf{z}_\iota - \mathbf{z}_\nu\|)$, $\iota, \nu = 1,\ldots,n$, $\mathbf{0}$ is a matrix with all entries 0 of appropriate dimension, and

$$\mathbf{P} = \begin{bmatrix} \mathbf{z}_1^T & 1 \\ \mathbf{z}_2^T & 1 \\ \vdots & \vdots \\ \mathbf{z}_n^T & 1 \end{bmatrix}, \quad \boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_d \\ a \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} f(\mathbf{z}_1) \\ f(\mathbf{z}_2) \\ \vdots \\ f(\mathbf{z}_n) \end{bmatrix}. \tag{6}$$

The matrix in (5) is invertible if and only if $\mathrm{rank}(\mathbf{P}) = d + 1$ (Powell 1992).

3 Review of surrogate model algorithms

3.1 General surrogate model algorithm

Surrogate model algorithms generally follow the same steps. First, an initial experimental design is created and the computationally expensive objective function is evaluated at the selected points. The objective function value predictions of the surrogate model at unsampled points are then used to select the next evaluation point. After the new function value has been obtained, the surrogate model is updated if the stopping criterion has not been satisfied (for example, the budget of function evaluations has not been exhausted) and a new point is selected for evaluation. Otherwise the algorithm stops and returns the best solution found during the optimization.

This surrogate optimization framework has been adopted in several well-known algorithms for continuous optimization such as EGO (Jones et al. 1998), Gutmann's RBF method (Gutmann 2001), DYCORS (Regis and Shoemaker 2013), SRBF (Regis and Shoemaker 2007), and SO-M-s (Müller and Shoemaker 2014). The major differences between these algorithms are the type of surrogate model used to approximate the expensive objective function and the method for selecting a new evaluation point.

3.2 Previous surrogate model algorithms for continuous optimization

Several surrogate model algorithms have been introduced in the literature for computationally expensive black-box optimization problems with continuous variables. The efficient global optimization algorithm (EGO) by Jones et al. (1998) uses a kriging surrogate model. A new decision variable vector is selected based on the solution of an auxiliary optimization problem that aims at maximizing the expected improvement that is computed based on the error estimate of the kriging surface.

Gutmann (2001) uses RBF surrogate models and selects the next evaluation point based on a target value strategy. A target value is defined and a computationally cheap auxiliary optimization problem that aims at minimizing a bumpiness measure is solved on the RBF model.

SO-M-s (Müller and Shoemaker 2014) does the next computationally expensive function evaluation at the minimum point of the surrogate surface. Any type of surrogate model may be used within the SO-M-s framework, but the authors showed that RBFs and ensembles containing RBFs perform generally well. EGO, Gutmann's RBF method, and SO-M-s have in common that an auxiliary optimization problem is solved on the computationally cheap surrogate model in order to select the next evaluation point.

The algorithm SRBF by Regis and Shoemaker (2007) uses an RBF model and a stochastic sampling approach. A large set of candidates for the next evaluation point is generated by adding random perturbations to *all* variables of the best point found so far. Two scores are computed for each candidate point and the candidate with the best weighted sum of these scores is selected as new evaluation point.

Regis and Shoemaker (2013) suggested a second stochastic sampling approach called DYCORS. The generation of candidate points in DYCORS is similar to SRBF, except that the probability of perturbing each variable of the best point found so far decreases with the number of realized expensive objective function evaluations. The search thus becomes more local. DYCORS has been shown to be more efficient for large-dimensional problems.

3.3 Previous surrogate model algorithms for mixed-integer optimization

Surrogate model algorithms for mixed-integer optimization of computationally expensive black-box problems are scarce and implementations of the algorithms are hardly available. SO-MI (Müller et al. 2013b; Müller 2014) is the first surrogate model based algorithm for mixed-integer optimization that is able to address problems with large numbers of variables that may have a large range and are not restricted to binary values. SO-MI uses a cubic RBF model and a stochastic sampling strategy in which four points are evaluated in parallel in each iteration. The MATLAB implementation is open source and available from the authors.

Holmström's radial basis function algorithm for mixed-integer problems (Holmström 2008b) uses an adaptive version of Gutmann's target value sampling strategy. The algorithm was shown to perform well for low-dimensional problems (up to 11 dimensions) with up to six integer variables of which most were binary. The implementation is contained in the commercial TOMLAB toolbox for MATLAB.

Davis and Ierapetritou (Davis and Ierapetritou 2009) developed a surrogate model algorithm for mixed-integer problems with binary variables. The authors combine a branch and bound algorithm with a kriging surface and show the effectiveness of the algorithm on two application examples from process synthesis. Hemker et al. (2008) introduced a mixed-integer surrogate optimization approach for water resources management.

Surrogate models have been used in connection with evolutionary strategies in order to address problems with computationally expensive objective functions. Zhuang et al. (2013) developed an algorithm that combines radial basis function neural networks and mixed-integer evolutionary strategies. Similarly, Li et al. (2008) used metamodels within mixed-integer evolutionary strategies and showed their applicability to ultrasound image analysis.

4 MISO framework

The algorithms for continuous optimization briefly reviewed in Sect. 3.2 can be easily modified for mixed-integer optimization problems. Only the initial experimental design and the strategy for selecting new sample points must be adjusted. The goal is to find a near-optimal solution within a limited number of function evaluations. Hence, no computationally expensive evaluations should be wasted at points that do not satisfy the integrality constraints. Also, for many application problems, the black-box simulation model may crash when continuous values are used for integer variables. This makes the application of methods such as branch and bound, that depend on solving relaxed subproblems, impossible. Thus, the goal is to evaluate the expensive objective function only at integer-feasible points. The general surrogate model framework can thus be modified to the mixed-integer surrogate optimization (MISO) framework shown in Algorithm 1.

Algorithm 1 General MISO Framework

```
1:  Initialization
2:  Create an initial experimental design. Ensure that the integer variables of
    the points in the design assume integer values. Do the expensive objective
    function evaluations at the selected points.
3:  Surrogate Model
4:  Fit the chosen surrogate model to the data in Step 2.
5:  Sampling
6:  Use the information from the surrogate model to select the point for doing
    the next expensive function evaluation. Ensure with the sampling strategy
    that the newly selected point $\mathbf{z}_{\text{new}}$ satisfies the integrality constraints.
7:  Do the expensive evaluation at $\mathbf{z}_{\text{new}}$: $f_{\text{new}} = f(\mathbf{z}_{\text{new}})$.
8:  if Stopping criterion is not met then
9:      Update the surrogate model and go to Step 6.
10: else
11:     Return the best solution found during the optimization.
12: end if
```

In Step 2 of Algorithm 1, when creating the initial experimental design, we only select points that satisfy the integrality constraints. We use a symmetric Latin hypercube design and round the values of the integer variables. The computationally expensive objective function evaluations are done at the selected points and the surrogate model is fit to this data in Step 4. When fitting the surrogate model, we assume that all variables are continuous in order to obtain a smooth surface. However, in Step 6 we have to guarantee that each newly selected evaluation point satisfies the integer constraints.

Our previous algorithm SO-MI (Müller et al. 2013b) fits into the MISO framework and may thus be considered part of it. SO-MI generates four groups of integer-feasible sample points in each iteration by (a) perturbing only the integer variables, (b) perturbing only the continuous variables, (c) perturbing both integer and continuous variables of the best point found so far, and (d) uniformly generating integer-feasible points from the whole variable domain. In contrast to MISO, we select in SO-MI always four new points in each iteration for doing the expensive function evaluation. If parallel computing resources are available, one can thus do almost four times as many expensive evaluations within the same number of iterations as MISO. On the other hand, if it is not possible to do several function evaluations in parallel, SO-MI may have a disadvantage because in MISO we update the surrogate model after each evaluation and thus we have more information available for selecting the next sample point, whereas in SO-MI we select always four points at once given the current information. MISO's sampling strategies can easily be extended to methods that select more than one new point in each iteration. Parallel sampling strategies are, however, beyond the scope of the present paper and we will investigate them in the future.

4.1 Modifications of continuous surrogate model algorithms for mixed-integer problems

We adapted SRBF (Regis and Shoemaker 2007), DYCORS (Regis and Shoemaker 2013), Gutmann's RBF method (Gutmann 2001), SO-M-s (Müller and Shoemaker 2014), and Forrester's implementation of EGO (Forrester et al. 2008) according to the MISO framework for mixed-integer problems. We will denote the algorithms as follows:

- MISO-CP (MISO-coordinate perturbation): the mixed-integer version of DYCORS;
- MISO-RS (MISO-random sampling): the mixed-integer version of SRBF;
- MISO-EI (MISO-expected improvement): the mixed-integer version of Forrester's EGO implementation (Forrester et al. 2008);
- MISO-TV (MISO-target value): the mixed-integer version of Gutmann's RBF method;
- MISO-SM (MISO-surface minimum): the mixed-integer version of SO-M-s.

MISO-CP and MISO-RS both follow the steps of the continuous algorithms described by Regis and Shoemaker (2007, 2013). Both methods generate candidate points by perturbing the best point found so far ($\mathbf{z}_{\text{best}}$). For mixed-integer problems, we use the perturbation $r = \text{sgn}(\rho)\max\{1, [|\sigma\rho|]\}$ for the integer variables, where

$\rho \sim \mathcal{N}(0, 1)$, σ denotes the perturbation radius of the current iteration, $[\cdot]$ denotes the nearest integer, and $\mathrm{sgn}(\cdot)$ denotes the sign function. Thus, the integer perturbation is at least one unit and integer which is either added or subtracted to the parameter value depending on the sign of ρ. When perturbing the continuous variables of $\mathbf{z}_{\text{best}}$, we add $r = \sigma\rho$ to the parameter value to be perturbed where the value of $\rho\sigma$ is, depending on ρ, either positive or negative. In particular, denote the candidate point in the following by $\mathbf{z}_{\text{cand}}$ and let $z_{\text{cand},i}$ and $z_{\text{best,i}}$ be the ith variable of $\mathbf{z}_{\text{cand}}$ and $\mathbf{z}_{\text{best}}$, respectively. Then

$$z_{\text{cand,i}} = \begin{cases} \max\{z_i^l, \min\{z_{\text{best,i}} + r, z_i^u\}\} & \text{if } i \text{ is chosen for perturbation} \\ z_{\text{best,i}} & \text{if } i \text{ is not chosen for perturbation} \end{cases} \tag{7}$$

where

$$r = \begin{cases} \rho\sigma, & \text{if } i \text{ is a continuous variable} \\ \mathrm{sgn}(\rho)\max\{1, [|\sigma\rho|]\} & \text{if } i \text{ is an integer variable} \end{cases}. \tag{8}$$

The probability of perturbing the ith variable of $\mathbf{z}_{\text{best}}$ is different for MISO-RS (all variables are perturbed) and MISO-CP (each variable is perturbed with decreasing probability, see Eq. (9) below).

For algorithms that solve an auxiliary optimization problem on the surrogate model in order to select new sample points such as MISO-EI, MISO-TV, and MISO-SM, we can substitute the optimization routine used for solving the auxiliary problem with a mixed-integer global optimization algorithm. Finding the optimum of the auxiliary problem is in general itself a global optimization problem. Thus, we can use, for example, a mixed-integer genetic algorithm for minimizing the bumpiness measure in MISO-TV, for finding the minimum point of the surrogate surface in MISO-SM, and for finding the maximum of the expected improvement in MISO-EI, respectively. Hence, the newly selected point $\mathbf{z}_{\text{new}}$ (the optimum of the auxiliary problem) will satisfy the integer constraints. Except for the optimization subroutine used for solving the auxiliary problems, MISO-EI, MISO-TV, and MISO-SM follow the steps of the algorithms described by Forrester et al. (2008), Gutmann (2001), Holmström (2008b), and Müller and Shoemaker (2014), respectively.

Table 1 gives an overview of all MISO algorithms that we examine in this paper. The table contains information about the algorithms' abbreviations, the meaning of the abbreviation and the type of search method that is used, and a literature reference in which the sampling strategy has been introduced in the continuous optimization literature (if applicable). The table also shows the type of surrogate model that is used and the number of the algorithm in which the steps are explained in this paper.

Table 1 MISO-based algorithms

#	MISO-	Full name/ search method/ reference	Surrogate	Alg. no.
1	CP	Coordinate perturbation (Regis and Shoemaker 2013), *c*-step	Cub. RBF	4
2	RS	Random sampling (Regis and Shoemaker 2007)	Cub. RBF	
3	SM	Surface minimum (Müller and Shoemaker 2014)	Cub. RBF	
4	TV	Target value (Gutmann 2001), *t*-step	Cub. RBF	5
5	EI	Expected improvement (Forrester et al. 2008)	Kriging	
6	CPTV	Combination of #1 and #4	Cub. RBF	3
7	CPTV-l(f)	Combination of #6 and gradient based local search on continuous variables	Cub. RBF	3
8	CPTV-l(o)	Combination of #6 and gradient-free local search on continuous variables	Cub. RBF	3

Abbreviation, full name, literature references, type of surrogate model used, and index of the algorithm (Alg. no.) in which the steps are described in this paper

4.2 MISO-CPTV and MISO-CPTV-local

We developed two new algorithms, namely MISO-CPTV and MISO-CPTV-local, that combine a coordinate perturbation search (*c*-step, stochastic) with a target value search (*t*-step, minimizing an auxiliary objective function on the surrogate model). MISO-CPTV-local is a memetic algorithm, i.e., we combine global (*c*- and *t*-step) and local optimization search (*l*-step), which aims at improving the solution found by MISO-CPTV locally for higher accuracy. A general overview of the individual steps of MISO-CPTV and MISO-CPTV-local is shown in Algorithm 2.

Algorithm 2 MISO-CPTV/MISO-CPTV-local: general overview

```
1: Initialization
2: Create an initial experimental design and fit the surrogate model to the data.
3: while Stopping criterion not met do
4:     Compute $\mathbf{z}_{\mathbf{new}}$:
5:     if c-Step then
6:         Create a large set of candidate points by adding random perturbations to randomly selected variables of the best point found so far. Use scoring criteria based on the surrogate model to select the best candidate point.
7:     else if t-Step then
8:         Define a target value for the objective function and minimize a bumpiness measure on the surrogate model.
9:     else if l-Step (for MISO-CPTV-local only) then
10:        Fix the integer variables of the best point found so far and do a local search on the continuous variables only.
11:    end if
12:    Do the expensive evaluation at $\mathbf{z}_{\mathbf{new}}$.
13:    Update the surrogate model with the new data.
14: end while
```

Algorithm 3 contains a detailed description of MISO-CPTV and MISO-CPTV-local. Details of the *c*-, *t*-, and *l*-steps are described in Algorithms 4, 5, and 6, respectively. Both algorithms require the following parameters, where the parameter settings 3–10 related to the *c*-step are adopted from Regis and Shoemaker (2013).

1. The number of points in the initial experimental design $n_0 = 2(d+1)$.
2. A maximum number of allowed function evaluations $n_{\max}$.
3. The initial perturbation radius $\sigma_0 = 0.2l(\mathcal{D})$, where $l(\mathcal{D})$ is the shortest side of the hyper-rectangle $\mathcal{D}$ defined by the variables' upper and lower bounds.
4. A minimum $\sigma_l = 2^{-6}\sigma_0$ for the perturbation radius σ ($\sigma_l \leq \sigma$).
5. The number of candidate points generated in each iteration $N = \min\{500d, 5000\}$.
6. A threshold for the number of allowed consecutive successful improvement trials in the *c*-step (coordinate search step) $T_s^c = 3$.
7. A threshold for the number of allowed consecutive unsuccessful improvement trials in the *c*-step $T_f^c = \max\{5, d\}$.
8. A threshold for the number of times the perturbation radius in the *c*-step can be decreased $T_r^c = 5$.
9. A weight pattern $\mathcal{W} = <0.3, 0.5, 0.8, 0.95>$ for computing the weighted sum of scores for the candidate points.
10. A function to determine the perturbation probability
$$q(n) = \min\{20/d, 1\}(1 - \log(n - n_0 + 1)/\log(n_{\max} - n_0)), \tag{9}$$
where n denotes the number of function evaluations done so far.
11. A pattern for the target value strategy stage $G = <0, 1, \ldots, P, P+1>$, where $P = 10$.
12. A threshold for the number of consecutive unsuccessful improvement trials in the *t*-step $T_f^t = |G|$, where $|G|$ denotes the cardinality of the set G.
13. A surrogate model constructed by using n evaluation points $s_n(\cdot)$.
14. A mixed-integer genetic algorithm MI-GA.
15. A threshold distance δ below which two points are considered equal. The distance $\|\cdot\|$ between two points is the Euclidean distance.
16. For MISO-CPTV-local only: a local optimization algorithm for continuous problems.

The number of points in the initial experimental design has to be large enough to fit the chosen surrogate model. For the cubic RBF with linear polynomial tail we need at least $d = 1$ points. The more points that are contained in the initial design, the better is the global fit of the initial surrogate model. However, the larger this number is, the fewer evaluations can be done during the iterative optimization. Hence, depending on whether insight into the global or the local behavior of the objective function is desired, the number of points in the initial design should be adjusted. If more points are allowed for the iterative improvement trials, then the local search will be more thorough and better objective function values can be found. The maximum number of allowed function evaluations depends on the time a single objective function evaluation requires and how much time there is to obtain the solution. If an evaluation takes, for example, approximately one hour, and a result has to be obtained within 30 days, then (neglecting the algorithm's own computational overhead, which is negligible in comparison to the time needed for the function evaluation), at most 720 evaluations can be done.

The parameters related to the perturbation radius (σ_0, σ_l, T_r^c) and the thresholds T_f^c and T_s^c, respectively, influence how thorough the local search is. Increasing T_f^c, T_s^c, and T_r^c and decreasing σ_l and σ_0 leads to a more thorough local search during the c-step because the generated candidate points will be closer to the best point found so far and the number of attempts to improve the current best point within the same perturbation radius is larger. Thus, more accurate solutions may be found during the c-step. If the budget of allowable function evaluations is small in comparison to the problem dimension d, we will have to be satisfied with a good approximation of a local minimum. The goal may be to find a good solution fast and searching locally is more efficient than using many evaluations for the global search in order to find various other promising parameter regions which can then not be explored because the budget of evaluations is exhausted. The choice of these parameters should depend on the problem dimension d, the number of allowable function evaluations, and the desired accuracy level. In MISO we adjusted these values following Regis and Shoemaker (2013). In connection with the perturbation parameters is the adjustment of the threshold distance δ. This value should be decreased if higher accuracy solutions are desired.

The weight pattern $\mathcal{W}$ may be adjusted depending on the desired search type. If the budget of allowable function evaluations is low relative to the problem dimension d (for example, $d = 30$, $n_{\max} = 100$) and a local search is the best chance of finding an improved solution, then using a weight pattern only with values close to zero is more efficient because preference will then be given to candidate points that have low predicted objective function values which is likely to be close to the best solution found so far and the search will therefore be more local. If the budget of allowable evaluations is large enough to cycle between local and global search, then the weight pattern should contain values from the whole range between zero and one. The adjustment of T_f^c should depend on the length of $\mathcal{W}$ in order to allow cycling at least once through the whole weight pattern after an improved solution has been found.

As shown by Holmström (2008a), the adjustment of the pattern for the target value sampling strategy influences the solution quality and a variety of target values is needed to make the sampling strategy effective for a wide range of problems. The length of the target value pattern should also depend on the number of allowable function evaluations. If the budget is low, then emphasis should be on the local search, and therefore fewer cycles in the global search step (smaller value for P) should be used. If there are sufficiently many function evaluations for a global search, P should be increased. We set the value of T_f^t equal to the length of the target value pattern in order to allow at least one cycle through all target values.

Algorithm 3 MISO-CPTV/MISO-CPTV-local

```
1: Initialization
2: $c$-Step $\leftarrow$ true, $t$-Step $\leftarrow$ false, $l$-Step $\leftarrow$ false.
3: $C_f^c \leftarrow 0, C_s^c \leftarrow 0, C_r^c \leftarrow 0$, $C_i^c \leftarrow 0$, $C_f^t \leftarrow 0, C_s^t \leftarrow 0$, $C_i^t \leftarrow 0$.
4: Create a symmetric Latin hypercube design with $n_0 = 2(d+1)$ points and round the integer variables. Do the computationally expensive function evaluations at the generated points. Denote the set of sampled points by $\mathcal{Z}$, i.e., $\mathcal{Z} \leftarrow \{\mathbf{z}_1, \ldots, \mathbf{z}_{n_0}\}$.
5: $n \leftarrow n_0$.
6: Fit the surrogate model $s_n(\cdot)$ to the data.
7: while Stopping criterion not met ($n < n_{\max}$) do
8:     Compute $\mathbf{z}_{\mathbf{new}}$:
9:     if $c$-Step then
10:        $c$-Step
11:        $C_i^c \leftarrow C_i^c + 1$.
12:        Use the sampling strategy described in Algorithm 4.
13:        $n \leftarrow n + 1$, $\mathcal{Z} \leftarrow \mathcal{Z} \cup \mathbf{z}_{\text{new}}$.
14:    else if $t$-Step then
15:        $t$-Step
16:        $C_i^t \leftarrow C_i^t + 1$.
17:        Use the sampling strategy described in Algorithm 5.
18:        $n \leftarrow n + 1$, $\mathcal{Z} \leftarrow \mathcal{Z} \cup \mathbf{z}_{\text{new}}$.
19:    else if $l$-Step (for MISO-CPTV-local only) then
20:        $l$-Step
21:        Use the local search strategy described in Algorithm 6.
22:        $n \leftarrow n + n_l$, $\mathcal{Z} \leftarrow \mathcal{Z} \cup \mathcal{Z}_l$.
23:    end if
24:    Update the surrogate model with the new data.
25: end while
```

In Algorithm 3, we first initialize the search type. We initialize the counters C_f^c and C_f^t for the number of consecutive failed improvement trials in the c-and t-step, respectively. We initialize C_s^c and C_s^t for counting the number of consecutive successful improvement trials in the c- and t-step, respectively, and C_r^c for counting the number of times we decreased the perturbation radius σ in the c-step. We also initialize the iteration counters C_i^c and C_i^t for the c- and t-step, respectively. We use the same approach for creating the initial experimental design as for the other MISO algorithms, i.e., a symmetric Latin hypercube design in which we round the integer variables. The evaluation of the computationally expensive objective function at the points in the initial design can be done in parallel if the necessary computing resources are available and the function evaluations are independent of each other. We fit a cubic RBF model with linear polynomial tail to the data and select $\mathbf{z}_{\text{new}}$ either by the coordinate perturbation strategy (c-step) or the target value strategy (t-step) in MISO-CPTV (see Algorithms 4 and 5, respectively). In MISO-CPTV-local, an additional l-step may be used (a local search described in Algorithm 6). The number of function evaluations and the set of points evaluated during the *l*-step are denoted by n_l and $\mathcal{Z}_l$, respectively. MISO-CPTV and MISO-CPTV-local both start with the *c*-step.

4.2.1 c-Step: coordinate perturbation search

The details of the *c*-step are given in Algorithm 4. The goal of using the *c*-step is to explore the whole variable domain and detect promising valleys in which the global minimum may be located.

Algorithm 4 *c*-Step details

```
1: Coordinate perturbation strategy
2: Determine the best point found so far: $\mathbf{z}_{\text{best}} \in \arg\min\{f(\mathbf{z}), \mathbf{z} \in \mathcal{Z}\}$, $f_{\text{best}} = f(\mathbf{z}_{\text{best}})$.
3: Determine the probability $q_n = q(n)$ of perturbing each variable of $\mathbf{z}_{\text{best}}$.
4: Generate candidate points
5: Create a set of $N$ candidate points by perturbing each variable of $\mathbf{z}_{\text{best}}$ with probability $q_n$ according to (7) and (8). If no variable is selected for perturbation by using probability $q_n$, randomly select one variable.
6: Score the candidate points
7: Use the surrogate model $s_n$ to predict the objective function values at the candidate points. Scale the predicted values to the interval $[0, 1]$ (surrogate model score, $S_s$).
8: Compute the distance of each candidate point to the set $\mathcal{Z}$ and scale the values to [0,1] (distance score, $S_d$).
9: Compute the weighted sum of the two scores, $S_t = w_s S_s + w_d S_d$, where $w_d \in \mathcal{W}$ and $w_s = 1 - w_d$. Select the candidate point with the best score (lowest value) as new evaluation point ($\mathbf{z}_{\text{new}}$).
10: Function evaluation and updates
11: Do the expensive function evaluation $f_{\text{new}} = f(\mathbf{z}_{\text{new}})$.
12: if $f_{\text{best}} < f_{\text{new}}$ (no improvement found) then
13:     $C_f^c \leftarrow C_f^c + 1, C_s^c \leftarrow 0$.
14:     if $C_f^c > T_f^c$ then
15:         if $C_r^c > T_r^c$ then
16:             c-Step $\leftarrow$ false, t-Step $\leftarrow$ true, $C_r^c \leftarrow 0$, $C_f^c \leftarrow 0$.
17:         end if
18:     else
19:         $C_r^c \leftarrow C_r^c + 1, \sigma \leftarrow \max\{\sigma_l, \sigma/2\}, C_f^c \leftarrow 0$.
20:     end if
21: else
22:     $f_{\text{best}} \leftarrow f_{\text{new}}$, $\mathbf{z}_{\text{best}} \leftarrow \mathbf{z}_{\text{new}}$, $C_s^c \leftarrow C_s^c + 1, C_f^c \leftarrow 0$.
23:     if $C_s^c > T_s^c$ then
24:         $\sigma \leftarrow \min\{\sigma_0, 2\sigma\}$, $C_s^c \leftarrow 0$.
25:     end if
26: end if
```

We generate a set of N candidates for the next sample point by adding random perturbations to randomly selected variables of the best point found so far ($\mathbf{z}_{\text{best}}$). Each candidate point is initially set equal to $\mathbf{z}_{\text{best}}$ and a uniform random number $\upsilon_i \sim \mathcal{U}(0,1)$ is drawn for each variable $i = 1, \ldots, d$. If $\upsilon_i < q(n)$ (the perturbation probability computed in Step 3), we add a random perturbation to that variable. We use the perturbation probability in Eq. (9) which decreases as the number of function evaluations increases. Hence, initially the perturbation probability is large and therefore the search is more global because the more variables that are perturbed, the more likely it is that the distance to $\mathbf{z}_{\text{best}}$ is large. As the algorithm progresses, the probability of perturbing the variables of $\mathbf{z}_{\text{best}}$ decreases, and thus the search becomes more local. If no variable is selected for perturbation, one variable is selected for perturbation at random.

We compute two scores for each candidate point. First, we use the surrogate surface to predict the objective function values of the candidate points (Step 7). We scale the values to [0,1] where low predicted objective function values will have a score S_s close to zero and large values will have a value S_s close to one. Secondly, we compute the distance of each candidate point to the set $\mathcal{Z}$ (Step 8) and scale these values to [0,1] such that points far away from $\mathcal{Z}$ (points in relatively unexplored regions of the variable domain) obtain a value S_d close to zero, and points that are close to $\mathcal{Z}$ obtain a score S_d close to one.

We compute a weighted sum of both scores in Step 9. The weights w_s and w_d for the surrogate surface criterion and the distance criterion, respectively, are adjusted in a cycling manner, i.e., in each iteration, w_d is selected as

$$w_d = \begin{cases} \mathcal{W}[k] & \text{if } k \equiv C_i^c \bmod |\mathcal{W}| > 0 \\ \mathcal{W}[|\mathcal{W}|] & \text{otherwise} \end{cases}, \tag{10}$$

where $|\cdot|$ denotes the cardinality of a set and $\mathcal{W}[j]$ denotes the jth element of $\mathcal{W}$ (see input 9). The candidate point with the lowest total score S_t is selected for evaluation.

The computationally expensive objective function is evaluated at the newly selected point ($f_{\text{new}} = f(\mathbf{z}_{\text{new}})$) in Step 11 and depending on whether or not f_{new} is better than f_{best}, C_f^c, C_s^c, C_r^c, and f_{best} are updated (see Steps 12–26). The c-step ends when the threshold of failed improvement trials T_f^c has been reached T_r^c times and the perturbation radius σ has been decreased T_r^c times (Steps 15–17).

4.2.2 *t-Step: target value search*

The details of the t-step are described in Algorithm 5. The goal of the t-step is to explore the vicinity of promising valleys in the variable domain in a more systematic way (define a target value and examine in which valley of the variable domain the resulting surface is least bumpy). Hence, the c-step helps us to find promising regions of the variable domain and the t-step is used to find improved solutions in these regions more efficiently.

Algorithm 5 t-Step details

```
1: Target value sampling
2: Select sample stage $g \in G$.
3: if g=0 then
4:     Inf-Step
5:     Use MI-GA to solve (12) and obtain $\mathbf{z}_{\text{new}}$.
6: else if $1 \leq g \leq P$ (Cycle step) then
7:     Global Search
8:     Use MI-GA to solve (15) and obtain $(\mathbf{z}_s, s_s)$.
9:     $w_g \leftarrow (1 - g/|G|)^2$.
10:    Define the target value $t \leftarrow s_s - w_g(\max\{f(\mathbf{z}_i), \mathbf{z}_i \in \mathcal{Z}\} - s_s)$.
11:    Use MI-GA to solve (16) and obtain $\mathbf{z}_{\text{new}}$.
12: else (Cycle step)
13:    Local Search
14:    Use MI-GA to solve (15) and obtain $(\mathbf{z}_s, s_s)$.
15:    if $s_s < f_{\text{best}} - 10^{-6}|f_{\text{best}}|$ then
16:        $\mathbf{z}_{\text{new}} \leftarrow \mathbf{z}_s$.
17:    else
18:        Define the target value $t \leftarrow f_{\text{best}} - 10^{-2}|f_{\text{best}}|$.
19:        Use MI-GA to solve (16) and obtain $\mathbf{z}_{\text{new}}$.
20:    end if
21: end if
22: if $\|\mathbf{z}_{\text{new}} - \mathbf{z}_i\| \leq \delta$ for any $i \in \{1, \ldots, n\}$ then
23:    repeat Randomly select a new point $\mathbf{z}_{\text{new}}$ from $\mathcal{D}$.
24:    until $\|\mathbf{z}_{\text{new}} - \mathbf{z}_i\| > \delta$ for all $i \in \{1, \ldots, n\}$.
25: end if
26: Function evaluation and updates
27: Do the expensive function evaluation $f_{\text{new}} = f(\mathbf{z}_{\text{new}})$.
28: if $f_{\text{best}} < f_{\text{new}}$ (no improvement found) then
29:    $C_f^t \leftarrow C_f^t + 1, C_s^t \leftarrow 0$.
30:    if $C_f^t > T_f^t$ then
31:        $t$-Step $\leftarrow$ false, $c$-Step $\leftarrow$ true, $C_f^t \leftarrow 0$.
32:    end if
33: else
34:    $f_{\text{best}} \leftarrow f_{\text{new}}$, $\mathbf{z}_{\text{best}} \leftarrow \mathbf{z}_{\text{new}}$, $C_s^t \leftarrow C_s^t + 1, C_f^t \leftarrow 0$.
35: end if
```

There are three different cases (Steps 3–21) in which an auxiliary optimization problem is solved to determine the next sample point (see also Holmström (2008a), AlgorithmRBF). We use the same notation as Gutmann (2001) and Holmström (2008a) for minimizing the bumpiness measure. Throughout the t-step, we have to solve one of the following computationally cheap auxiliary minimization problems depending on the type g of the target value step the algorithm is currently in (Step 2). The step type g is defined by

$$g = \begin{cases} G[k] & \text{if } k \equiv C_l^t \bmod |G| > 0 \\ G[|G|] & \text{otherwise} \end{cases}. \tag{11}$$

In the "Inf-Step" (Steps 3–5), we select as new evaluation point

$$\mathbf{z}_{\text{new}} \in \arg\min_{\mathbf{z}\in\mathcal{D}} \mu_n(\mathbf{z}), \tag{12}$$

where $\mu_n(\mathbf{z})$ corresponds to the (n + 1)th value of $\mathbf{v}$ when solving the augmented linear system

$$\begin{bmatrix} \mathbf{\Phi}_z & \mathbf{P}_z \\ \mathbf{P}_z^T & \mathbf{0} \end{bmatrix} \mathbf{v} = \begin{bmatrix} \mathbf{0}_n \\ 1 \\ \mathbf{0}_d \end{bmatrix}, \tag{13}$$

where $\mathbf{0}_n$ and $\mathbf{0}_d$ denote vectors with n and d zeros, respectively, and

$$\mathbf{\Phi}_z = \begin{bmatrix} \mathbf{\Phi} & \boldsymbol{\phi}_z \\ \boldsymbol{\phi}_z^T & 0 \end{bmatrix}, \quad \mathbf{P}_z = \begin{bmatrix} \mathbf{P} & \\ \mathbf{z}^T & 1 \end{bmatrix}, \quad \text{and} \quad (\boldsymbol{\phi}_z)_i = \phi(\|\mathbf{z} - \mathbf{z}_i\|), i = 1, \ldots, n. \tag{14}$$

In the "Cycle step - global search" (Steps 6–11), the goal is to first find the minimum point of the surrogate surface (Step 8):

$$\mathbf{z}_{\text{s}} \in \arg\min_{\mathbf{z}\in\mathcal{D}} s_n(\mathbf{z}), \tag{15}$$

and we denote $s_s = s_n(\mathbf{z}_s)$. Based on the value of s_s and a target value t that is computed based on the pattern G (Steps 9–10), we determine

$$\mathbf{z}_{\text{new}} \in \arg\min_{\mathbf{z}\in\mathcal{D}} \mu_n(\mathbf{z})[s_n(\mathbf{z}) - t]^2, \tag{16}$$

where $\mu_n(\mathbf{z})$ is defined as for (12).

In the "Cycle step - local search" (Steps 12–20), we first find the minimum of the surrogate surface by solving (15) (Step 14). If the value $s_s = s_n(\mathbf{z}_s)$ is a relative improvement of f_{best} of at least 10^{-6}, we use the minimum point of the surrogate surface as new evaluation point (Steps 15–16). Otherwise, we define a target value that corresponds to a 1 % improvement of the best function value found so far and we solve (16) (Steps 18–19).

For each of the three cases, if the newly determined point $\mathbf{z}_{\text{new}}$ is closer than the threshold distance δ to an already evaluated point, we repeatedly uniformly select a random point from $\mathcal{D}$ until the selected point has a distance larger than δ to the set of already evaluated points $\mathcal{Z}$ (Steps 22–25). When generating the random point, we ensure that the integrality constraints are satisfied. We do the computationally expensive function evaluation at the newly selected point (Step 27). If the new function value is not an improvement of f_{best}, we update the counters C_f^t and C_s^t, respectively (Steps 28–29). If the fail counter C_f^t exceeds the threshold of allowable failed improvement trials T_f^t, we leave the t-step and go back to the c-step (Steps 30–32). If we found an improvement, we update $f_{\text{best}}, \mathbf{z}_{\text{best}}, C_s^t$, and C_f^t (Steps 33–35).

4.2.3 l-Step: local search

The local search step is only used in MISO-CPTV-local. While MISO-CPTV alternates only between the c-step and the t-Step until the maximum number of function evaluations has been reached, MISO-CPTV-local enters a local search phase whenever the sequence <c-Step, t-Step, c-Step> did not lead to any improvement. The goal of using this sequence is to first explore the whole variable domain using the c-step and to find promising regions of the variable domain. Initially, the c-step search is more global because the perturbation probability of the variables of the best point found so far is large (see Eq. 9). Denote the best point found during the c-step by $\mathbf{z}^c_{\text{best}}$. In the t-step, we aim at finding an improvement of $\mathbf{z}^c_{\text{best}}$ by applying the target value strategy. It is likely that improvements found during the t-step are in promising valleys of the variable domain that were detected during the c-step because the predicted objective function values are lower. Denote the best point found during the t-step by $\mathbf{z}^t_{\text{best}}$. The goal of using the c-step again after the t-step is to explore the neighborhood of $\mathbf{z}^t_{\text{best}}$ because the perturbation probability of the variables of $\mathbf{z}^t_{\text{best}}$ is now lower (see Eq. (9)) and the search is therefore more local. Furthermore, it is possible that $\mathbf{z}^t_{\text{best}}$ is in a different region of the variable domain than $\mathbf{z}^c_{\text{best}}$, and thus, exploring the close vicinity of $\mathbf{z}^t_{\text{best}}$ will improve the response surface in that region and a better starting guess for the following local search may be found. If all three improvement attempts <c-step, t-step, c-step> fail, it is an indicator that improvements by a global search and random perturbations cannot be found anymore and a more thorough and accurate local search is necessary. The goal of the local search step is to further improve the accuracy of the best solution found so far. Thus, during the local search we only consider the continuous variables.

In general, if $|M|$ denotes all possible combinations of integer variable values for a given problem, then there is for each such combination a global minimum with respect to the continuous variables. During the c- and t-step we determined the best point found so far $\mathbf{z}_{\text{best}}$ by searching over the integer *and* continuous variables. In the local search we now try to improve the objective function value by fixing the integer variables of $\mathbf{z}_{\text{best}}$ and doing a local optimization only with respect to the continuous variables:

$$\mathbf{z}_l \in \arg\min_{\mathbf{z}\in\mathcal{D}} \{f(\mathbf{z}|z_i), z_i = z_{\text{best},i} \forall i \in \mathbb{I}\}, \tag{17}$$

where $\mathbb{I}$ denotes the indices of the integer variables and $z_{\text{best},i}$ denotes the ith variable of $\mathbf{z}_{\text{best}}$. Hence, we will be able to find at least a local minimum associated with the integer variables of $\mathbf{z}_{\text{best}}$. If the best objective function value $f_l = f(\mathbf{z}_l)$ found by the local search is better than f_{best}, we update the best solution found so far (Algorithm 6, Steps 3–4). If the budget of allowed function evaluations has not been exhausted during the l-step, we go back to the c-step (Step 6).

Algorithm 6 l-Step (Local Search Step)

```
1: Fix the integer variables of $\mathbf{z}_{\text{best}}$.
2: Use a local search algorithm on the true objective function starting from
   the best solution found so far to solve (17) and obtain $(\mathbf{z}_l, f_l)$ (the best
   solution found by the local search).
3: if $f_l < f_{\text{best}}$ (improvement found) then
4:     $f_{\text{best}} \leftarrow f_l$, $\mathbf{z}_{\text{best}} \leftarrow \mathbf{z}_l$.
5: end if
6: $c$-Step $\leftarrow$ true, $l$-Step $\leftarrow$ false, $t$-Step $\leftarrow$ false.
```

We consider two options of local search algorithms in the l-step for searching on the true objective function, namely the MATLAB built-in optimizer fmincon that numerically computes derivatives and the derivative-free algorithm ORBIT (Wild et al. 2007) that uses a cubic radial basis function surrogate model. In the latter case, after ORBIT has finished, we use fmincon in an attempt to further improve the solution. The incentive behind using first ORBIT and then fmincon is that ORBIT might be able to find a better starting guess for fmincon and hence fewer expensive function evaluations may be needed in the fmincon stage. We call the algorithm that uses fmincon only for the local search MISO-CPTV-l(f) and the algorithm that uses ORBIT we call MISO-CPTV-l(o).

5 Numerical experiments

5.1 Experimental setup

Algorithms for computationally expensive black-box optimization problems with integrality constraints are scarce. In the numerical experiments we compare the performance of the MISO algorithms introduced in Sect. 4 to SO-MI (Müller et al. 2013b; Müller 2014), nonlinear optimization by mesh adaptive direct search (NOMAD) (Le Digabel 2011), and MATLAB's genetic algorithm (GA). We include GA because it is a widely used algorithm for mixed-integer black-box problems, but we do not expect it to perform very well for computationally expensive problems where only few hundred function evaluations are allowable.

We use a cubic RBF model in SO-MI as done in Müller et al. (2013b). Note that SO-MI is contained in MATSuMoTo (MATLAB Surrogate Model Toolbox (Müller 2014)) and can in general be used with any other surrogate model. NOMAD is a mesh-adaptive direct search method developed for computationally expensive black-box optimization problems and is, although not primarily developed for problems with integrality constraints, applicable to mixed-integer problems (Abramson et al. 2009). We use NOMAD version 3.6.2 in the numerical experiments with the setting `VNS 0.75` (variable neighborhood search method in an attempt to escape from local minima), which is contained in the OPTI Toolbox v2.05 (Currie and Wilson 2012).

The goal of this paper is to develop algorithms that are able to find near optimal solutions for computationally expensive optimization problems efficiently. We limit the number of allowed function evaluations to 500 for all test problems since in practice often only few hundred function evaluations are allowable. We compare the algorithms based on the best objective function value found after an equal number of function evaluations. In practice, the computational expense is caused mainly by the objective function evaluations and the computational overhead of the optimization algorithms themselves is in comparison negligible. We do 20 trials with each algorithm for each problem.

In order to facilitate a fair comparison, all algorithms use the same initial experimental design for the same trial of the same problem. NOMAD starts the systematic search from the best point contained in the initial design. For the genetic algorithm, we give the best point from the initial design as partial initial population. The remaining individuals in the initial population are generated with default MATLAB settings. We use a population size of 20. We cannot use all points from the initial experimental design as starting population since the number of points depends on the number of variables ($2(d+1)$) and is generally not equal to 20.

5.2 Test problems

We compared the algorithms on ten numerically inexpensive test problems, four problems arising in reliability redundancy engineering, and a problem arising in the optimal design of truss structures. For the computationally cheap test problems, we know the analytical description of the objective function (see the online supplement). However, we treat the problems as black-boxes in order to examine the efficiency of the algorithms for problems with different characteristics such as multimodality, convexity, and binary variables. The test problems have been derived from benchmark problems that are often used in continuous global optimization and we impose integer constraints for some of the variables.

In reliability-redundancy optimization, the goal is to maximize the reliability of a system (the mean time to failure) given restrictions on, for example, the total costs and weight of the system. A system consists of several components. The reliability of the system can be increased by either increasing each component's reliability (continuous variables) or by adding redundancy (integer variables). See the online supplement for further details.

The second application problem arises in optimal design. The goal is to minimize the weight of a truss dome subject to a displacement constraint. The dome consists of tubular members whose lengths (continuous variables) and wall thicknesses (integer variables, production restrictions do not allow arbitrary wall thicknesses) are the decision variables. The nodal displacement under loading is computed by a finite element analysis. The structure consists of 24 elements (24 integer variables) and the location of 7 nodes can be adjusted (7 continuous variables).

Table 2 gives an overview over the test problems. The table shows the problem number (column "ID"), the number of integer variables (column "d_1"), the number of continuous variables (column "d_2"), and the variable ranges. Problems 1–10 are the computationally cheap test problems. Problems 11–14 are the reliability redundancy optimization problems, and problem 15 is the structural optimization problem.

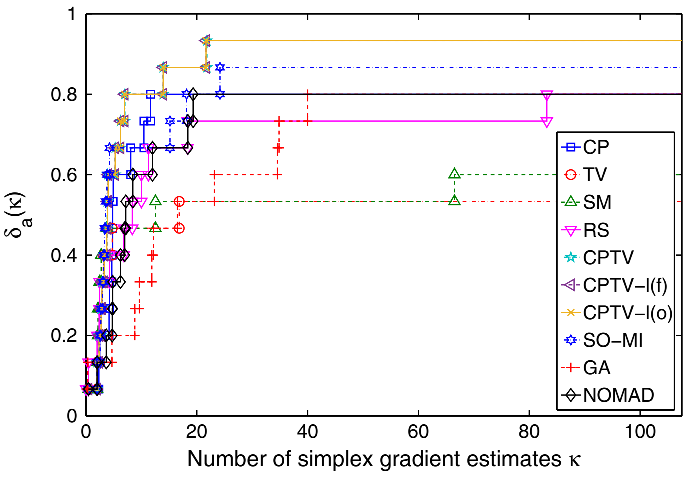
Table 2 Test problems for algorithm comparison

ID	d_1	d_2	Variable range
1	5	7	$\{-1,3\}^5 \times [-1,3]^7$
2	4	4	$\{-10,10\}^4 \times [-10,10]^4$
3	2	3	$\{-100,100\}^2 \times [-100,100]^3$
4	2	3	$\{0,10\}^2 \times [0,10] \times [0,1]^2$
5	5	5	$\{3,9\}^5 \times [3,9]^5$
6	6	9	$\{-15,30\}^6 \times [-15,30]^9$
7	1	1	$\{-5,10\} \times [0,15]$
8	10	5	$\{-15,30\}^{10} \times [-15,30]^5$
9	1	2	$\{0,1\} \times [0,1]^2$
10	30	30	$\{-15,30\}^{30} \times [-15,30]^{30}$
11	5	5	$\{1,10\}^5 \times [0.5,0.999999]^5$
12	4	4	$\{1,10\}^4 \times [0.5,0.999999]^4$
13	5	5	$\{1,10\}^5 \times [0.5,0.999999]^5$
14	5	5	$\{1,10\}^5 \times [0.5,0.999999]^5$
15	24	7	$\{1,10\}^{24} \times [0,1000]^7$

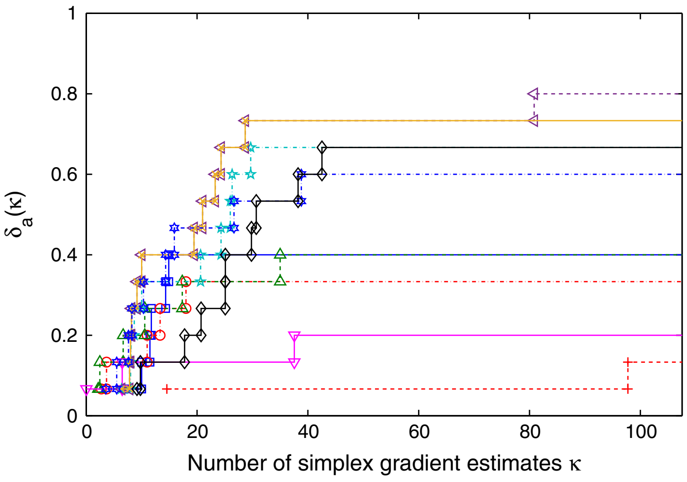
5.3 Numerical results and discussion

At this point we want to note that the computational effort of MISO-EI is considerably larger than that of all other algorithms (as observed also by Müller and Shoemaker (2014)). MISO-EI needs on average 500 times more computation time than MISO-CPTV-l(f) (more than 120 h versus 0.2 h) which is due to the computation of the kriging parameters. Since MISO-EI does not appear to be efficient, we only examined its performance for the first five test problems (the results are summarized in Table A1 in the online supplement where the average best solution and standard deviations over 20 trials found by each algorithm are shown). The results for these test problems show that MISO-EI is not promising and performs worst for two of the problems. One reason for the worse performance of MISO-EI may be related to the difficulty of finding the maximum of the expected improvement function as the number of variables and sample points increases. The number of local maxima generally increases as the number of evaluated points increases, and hence finding the global optimum of the expected improvement function becomes increasingly difficult. One could possibly adjust the parameter settings of the subsolver that maximizes the expected improvement function to explore local maxima more thoroughly, but this would come at a cost of computation time which is larger for kriging than for the other surrogate models in general. Based on these preliminary results and the computational cost of MISO-EI, we decided to not use MISO-EI for the remaining problems and we do not include it in the following analysis.

We summarize the results of the numerical experiments in form of data and performance profiles as suggested by Moré and Wild (2009) (the numerical results are summarized in Tables A2 and A3 in the online supplement where we show the average best objective function values and standard deviations). We use the MATLAB codes provided on http://www.mcs.anl.gov/ ~ more/dfo/ for creating Figs. 1 and 2. We create the profile plots based on the average objective function value found over all 20 trials by each algorithm.

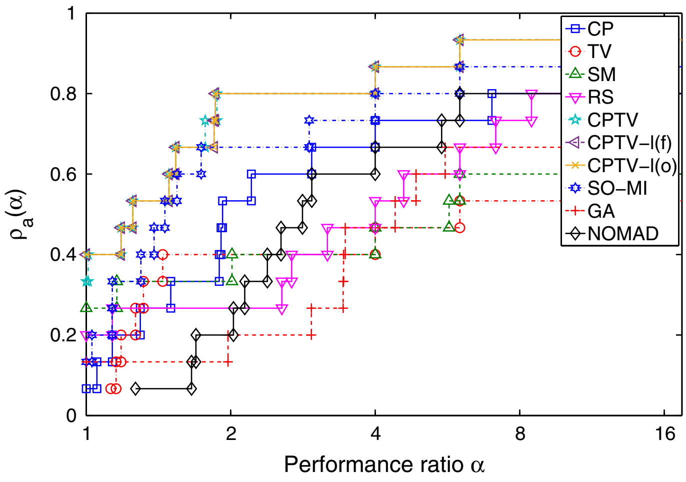


(a) Data profile for accuracy level $\tau = 10^{-1}$.

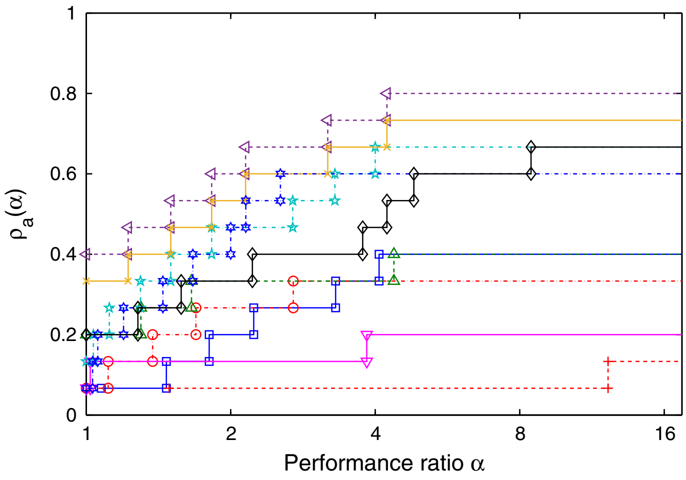


(b) Data profile for accuracy level $\tau = 10^{-3}$.

Fig. 1 Data profiles. Both *figures* share the same legend. The algorithms following the MISO framework are abbreviated with their sampling strategies



(a) Performance profile for accuracy level $\tau = 10^{-1}$.



(b) Performance profile for accuracy level $\tau = 10^{-3}$.

Fig. 2 Performance profiles. Both *figures* share the same legend. The algorithms following the MISO framework are abbreviated with their sampling strategies

We denote the set of problems and the set of algorithms in the comparison by $\mathcal{P}$ and $\mathcal{A}$, respectively. Let $l_{\gamma,a}$, where $\gamma \in \mathcal{P}$ and $a \in \mathcal{A}$, be the used performance measure. Then the performance ratio is defined as (Moré and Wild 2009)

$$r_{\gamma,a} = \frac{l_{\gamma,a}}{\min\{l_{\gamma,a} : a \in \mathcal{A}\}}. \tag{18}$$

The performance profile of algorithm $a \in \mathcal{A}$ shows the fraction of problems where the performance ratio is at most α:

$$\rho_a(\alpha) = \frac{1}{|\mathcal{P}|}\text{size}\{\gamma \in \mathcal{P} : \mathrm{r}_{\gamma,\mathrm{a}} \leq \alpha\}. \tag{19}$$

Here $|\mathcal{P}|$ denotes the cardinality of the set $\mathcal{P}$. High values for $\rho_a(\alpha)$ are better. The performance profile reflects how well an algorithm performs relative to the other algorithms. Data profiles on the other hand show the raw data. They illustrate the percentage of problems solved for a given tolerance τ within a given number of simplex gradient estimates $\kappa = n/(d+1)$, where n denotes the number of function evaluations. If $l_{\gamma,a}$ denotes the number of function evaluations needed to satisfy a convergence test with tolerance τ, then the percentage of problems that can be solved within κ simplex gradient estimates is defined as

$$\delta_a(\kappa) = \frac{1}{|\mathcal{P}|}\text{size}\left\{\gamma \in \mathcal{P} : \frac{\mathrm{l}_{\gamma,\mathrm{a}}}{\mathrm{d}_\gamma + 1} \leq \kappa\right\}, \tag{20}$$

where d_γ denotes the dimension of problem $\gamma \in \mathcal{P}$.

Figure 1 shows data profiles for all algorithms for accuracy level $\tau = 10^{-1}$ and $\tau = 10^{-3}$. In practice, one is often satisfied with an accuracy of $\tau = 10^{-3}$ since the simulation models themselves are approximations of physical phenomena and therefore inaccurate. For reasons of space considerations, we abbreviate the algorithms following the MISO framework by their sampling strategy in Figs. 1 and 2. For example, CPTV stands for MISO-CPTV, etc.

For both accuracy levels, we observe that except for GA all algorithms perform initially (up to 10 simplex gradient estimates) similarly. However, for $\tau = 10^{-1}$, after about 10 simplex gradient estimates, we can see that MISO-CPTV, MISO-CPTV-l(o), and MISO-CPTV-l(f) outperform the other algorithms. In fact, there is no difference between the performance of these algorithms. Similarly, for the accuracy $\tau = 10^{-3}$, MISO-CPTV-l(o) and MISO-CPTV-l(f) find better solutions than the other algorithms. MISO-CPTV-l(o) and MISO-CPTV-l(f) perform better than MISO-CPTV, which shows that the local search leads to higher-accuracy solutions. GA is able to outperform MISO-TV and MISO-SM after about 25 simplex gradient estimates for the accuracy level $\tau = 10^{-1}$. If solutions of higher accuracy are required, we can see that GA performs worst among all algorithms.

The performance profiles in Fig. 2 show similar results. MISO-CPTV, MISO-CPTV-l(o), and MISO-CPTV-l(f) perform equally well for $\tau = 10^{-1}$, whereas MISO-CPTV-l(f) performs better than all other algorithms for $\tau = 10^{-3}$. Figure 2b shows, for example, that for the performance ratio of $\alpha = 4$ there is a performance difference between NOMAD and MISO-CPTV-l(f) of about 25 %, which means that for 25 % of the problems, NOMAD needs four times as many function evaluations to reach the same accuracy as MISO-CPTV-l(f).

In summary, the results of the numerical experiments show that the MISO algorithms that combine coordinate search with target value and local search (MISO-CPTV, MISO-CPTV-l(o), MISO-CPTV-l(f)) perform better than the algorithms that use only a single sampling method (MISO-SM, MISO-RS, MISO-TV, MISO-CP). One reason why MISO-CPTV performs better than MISO-CP may be that MISO-CPTV initially searches globally for improvements (the perturbation probability of each variable of the best point found so far is large) and then switches to the target value strategy. The target value strategy then explores promising regions of the variable domain found during the c-step more systematically (low target values are more likely to be assumed in regions where low objective function values have been observed) than when only random candidates are generated within a decreasing perturbation radius. On the other hand, using only the target value sampling (MISO-TV) may lead to a very thorough exploration of some local minimum which does not improve the surrogate model globally, and thus other promising regions of the variable domain may be missed. We can also see that, similar to the results for continuous problems reported in Regis and Shoemaker (2013), the coordinate perturbation strategy (MISO-CP), which only perturbs a fraction of the variables of the best point found so far for creating candidate points, performs better than the random strategy (MISO-RS), which perturbs all variables, especially for lower tolerance levels τ. In comparison to our previous algorithm SO-MI, the results show that MISO-CPTV, MISO-CPTV-l(o), and MISO-CPTV-l(f) are an improvement.

The comparison of MISO-CPTV-l(f) and MISO-CPTV-l(o) shows that for the low accuracy $\tau = 10^{-1}$ both versions perform equally well. For the higher accuracy $\tau = 10^{-3}$, MISO-CPTV-l(f) performs slightly better, indicating that for our approach of fixing the integer variables and locally searching for improvements only with respect to the continuous variables, a derivative-free local search does not have an advantage over immediately using a local search that numerically computes derivatives.

6 Conclusions

The goal of this paper was to introduce the mixed-integer surrogate optimization (MISO) framework, a new algorithm framework for solving computationally expensive black-box optimization problems with mixed-integer variables that may have large ranges and are not restricted to binary values. The MISO framework ensures that all sample points satisfy the integer constraints, and thus no computationally expensive function evaluations are wasted on evaluating points that do not satisfy the integer constraints. This is a great advantage over algorithms that are based on solving relaxed subproblems such as branch and bound methods, especially for black-box simulations that crash when integer variables take on real values.

We used the MISO framework in combination with several well-known sampling strategies from the continuous optimization literature that we modified for mixed-integer problems such as Gutmann's target value strategy (Gutmann 2001), DYCORS (Regis and Shoemaker 2013), SRBF (Regis and Shoemaker 2007), expected improvement (Forrester et al. 2008), and SO-M-s (Müller and Shoemaker 2014). We also introduced two new MISO algorithms, namely MISO-CPTV that combines a coordinate perturbation search with a target value strategy, and MISO-CPTV-local that uses in addition a local search to further improve the solution accuracy.

We compared MISO in numerical experiments to our previous algorithm SO-MI (Müller et al. 2013b), NOMAD version 3.6.2 (Le Digabel 2011), and MATLAB's genetic algorithm. The numerical comparison on ten benchmark problems, four application problems arising in reliability optimization, and one structural optimization application shows that the MISO algorithms that use combinations of sampling strategies, namely MISO-CPTV and MISO-CPTV-local, find improved solutions much more efficiently than all other algorithms. Hence, MISO is a promising approach to solving computationally expensive mixed-integer black-box optimization problems.

Finally, we want to remark that we can develop a framework similar to MISO for pure integer problems where the integer variables have large ranges and are not restricted to binary values only. For the random sampling methods such as the coordinate perturbation strategy, one has to guarantee that all candidate points' variables are integer. For sampling strategies that solve an auxiliary optimization problem on the surrogate surface, one has to choose a subsolver that is able to address pure integer global optimization problems (for example, genetic algorithms or, depending on the range of the variables, complete enumeration may be possible). This, however, will be the topic of future research.

Acknowledgments This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under contract number DE-AC02005CH11231. Partial support was also provided by NSF CISE 1116298 to Prof. Shoemaker at Cornell University, School of Civil and Environmental Engineering.

References

Abramson M, Audet C, Chrissis J, Walston J (2009) Mesh adaptive direct search algorithms for mixed variable optimization. Optim Lett 3:35–47

Booker A, Dennis J Jr, Frank P, Serafini D, Torczon V, Trosset M (1999) A rigorous framework for optimization of expensive functions by surrogates. Struct Multidiscip Optim 17:1–13

Conn A, Scheinberg K, Vicente L (2009) Introduction to Derivative-Free Optimization. SIAM

Currie J, Wilson D (2012) Foundations of computer-aided process operations. OPTI: lowering the barrier between open source optimizers and the industrial MATLAB user. Savannah, Georgia, USA

Davis E, Ierapetritou M (2009) Kriging based method for the solution of mixed-integer nonlinear programs containing black-box functions. J Glob Optim 43:191–205

Forrester A, Sóbester A, Keane A (2008) Engineering design via surrogate modelling—a practical guide. Wiley, Chichester

Friedman J (1991) Multivariate adaptive regression splines. Ann Stat 19:1–141

Giunta A, Balabanov V, Haim D, Grossman B, Mason W, Watson L, Haftka R (1997) Aircraft multidisciplinary design optimisation using design of experiments theory and response surface modelling. Aeronaut J 101:347–356

Glaz B, Friedmann P, Liu L (2008) Surrogate based optimization of helicopter rotor blades for vibration reduction in forward flight. Struct Multidiscip Optim 35:341–363

Goel T, Haftka RT, Shyy W, Queipo NV (2007) Ensemble of surrogates. Struct Multidiscip Optim 33:199–216

Gutmann H (2001) A radial basis function method for global optimization. J Glob Optim 19:201–227

Hemker T, Fowler K, Farthing M, von Stryk O (2008) A mixed-integer simulation-based optimization approach with surrogate functions in water resources management. Optim Eng 9:341–360

Holmström K (2008a) An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization. J Glob Optim 41:447–464

Holmström K (2008b) An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer global optimization. J Glob Optim 9:311–339

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

Koziel S, Leifsson L (2013) Surroagte-based modeling and optimization: applications in engineering. Springer, New York

Le Digabel S (2011) Algorithm 909: NOMAD: nonlinear optimization with the mads algorithm. ACM Trans Math Softw 37:44

Li R, Emmerich M, Eggermont J, Bovenkamp E, Back T, Dijkstra J, Reiber H (2008) Metamodel-assisted mixed-integer evolution strategies and their applications to intravascular ultrasound image analysis. In: IEEE World Congress on Computational Intelligence, IEEE, pp 2764–2771

Marsden A, Wang M, Dennis J Jr, Moin P (2004) Optimal aeroacoustic shape design using the surrogate management framework. Optim Eng 5:235–262

Moré J, Wild S (2009) Benchmarking derivative-free optimization algorithms. SIAM J Optim 20:172–191

Müller J (2014) MATSuMoTo: The MATLAB surrogate model toolbox for computationally expensive black-box global optimization problems. arXiv:14044261

Müller J, Shoemaker C, Piché R (2013a) SO-I: a surrogate model algorithm for expensive nonlinear integer programming problems including global optimization applications. J Glob Optim 59:865–889. doi:10.1007/s10,898-013-0101-y

Müller J, Shoemaker C, Piché R (2013b) SO-MI: a surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. Comput Oper Res 40:1383–1400

Müller J, Piché R (2011) Mixture surrogate models based on Dempster–Shafer theory for global optimization problems. J Glob Optim 51:79–104

Müller J, Shoemaker C (2014) Influence of ensemble surrogate models and sampling strategy on the solution quality of algorithms for computationally expensive black-box global optimization problems. J Glob Optim 60:123–144. doi:10.1007/s10898-014-0184-0

Myers R, Montgomery D (1995) Response surface methodology: process and product optimization using designed experiments. Wiley-Interscience Publication, Hoboken

Powell M (1992) Advances in numerical analysis, vol. 2: wavelets, subdivision algorithms and radial basis functions. In: Light WA (ed) The theory of radial basis function approximation in 1990. Oxford University Press, Oxford, pp 105–210

Rashid S, Ambani S, Cetinkaya E (2012) An adaptive multiquadric radial basis function method for expensive black-box mixed-integer nonlinear constrained optimization. Eng Optim 45:185–206. doi:10.1080/0305215X.2012.665450

Regis R, Shoemaker C (2007) A stochastic radial basis function method for the global optimization of expensive functions. INFORMS J Comput 19:497–509

Regis R, Shoemaker C (2009) Parallel stochastic global optimization using radial basis functions. INFORMS J Comput 21:411–426

Regis R, Shoemaker C (2013) Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. Eng Optim 45:529–555

Simpson T, Mauery T, Korte J, Mistree F (2001) Kriging metamodels for global approximation in simulation-based multidisciplinary design optimization. AIAA J 39:2233–2241

Viana F, Haftka R, Steffen V (2009) Multiple surrogates: how cross-validation errors can help us to obtain the best predictor. Struct Multidiscip Optim 39:439–457

Wild S, Regis R, Shoemaker C (2007) ORBIT: optimization by radial basis function interpolation in trust-regions. SIAM J Sci Comput 30:3197–3219

Wild S, Shoemaker C (2013) Global convergence of radial basis function trust-region algorithms for derivative-free optimization. SIAM Rev 55:349–371

Zhuang L, Tang K, Jin Y (2013) Metamodel assisted mixed-integer evolution strategies based on Kendall rank correlation coefficient. In: Hea Yin (ed) IDEAL 2013. Springer-Verlag, Berlin, pp 366–375