Noname manuscript No.
(will be inserted by the editor)

MISO: Mixed-Integer Surrogate Optimization Framework

Juliane Müller

Received: date / Accepted: date

Abstract We introduce MISO, the Mixed-Integer Surrogate Optimization framework. MISO aims at solving computationally expensive black-box optimization problems with mixed-integer variables. Although encountered in many applications, such as optimal reliability design or structural optimization, for example, where time consuming simulation codes have to be run in order to obtain an objective function value, the development of algorithms for this type of optimization problem has rarely been addressed in the literature. 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. 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 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, and SO-MI. The results show that MISO is in general very efficient with respect to finding improved solutions within very few function evaluations. The performance of MISO depends on the chosen sampling strategy. The MISO algorithm that combines a dynamic coordinate search with a target value strategy and a local search performs best among all algorithms.

J. Müller
Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, Berkeley, CA, 94720
E-mail: juliane.mueller2901@gmail.com

Abbreviations and Notation

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-CS	MISO - Coordinate Search
MISO-EI	MISO - Expected Improvement
MISO-RS	MISO - Random Sampling
MISO-SM	MISO - Surface Minimum
MISO-TV	MISO - Target Value
MISO-CSTV	MISO with combination of CS and TV
MISO-CSTV-local	MISO with combination of CS, TV, and a local search
MISO-CSTV-l(f)	MISO-CSTV-local that uses fmincon as local optimizer
MISO-CSTV-l(o)	MISO-CSTV-local that uses ORBIT (Wild et al, 2007) as local optimizer
NOMAD	Nonlinear Optimization by Mesh-Adaptive Direct search (Le Digabel, 2011)
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, $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$
n_0	Number of points in the initial experimental design
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 such computationally expensive black-box problems, the goal is to find near optimal solutions within only very few 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 detect promising sample points in the variable domain. Hence, the computationally expensive objective function is evaluated only at very few 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 very 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.

We consider optimization problems of the following form:

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

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

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

where $f(\cdot)$ denotes the computationally expensive black-box objective function, z_i^l and z_i^u denote the lower and upper bounds of variable i, 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, often only few hundred evaluations of $f(\mathbf{z})$ are allowable, and thus algorithms that are able to find a (near) optimal solution within a very limited number of function evaluations are needed.

In Section 2, we briefly describe different surrogate model types. We give a general surrogate model optimization algorithm description in Section 3 and we briefly review widely-used surrogate model algorithms for continuous optimization and the few developments for mixed-integer problems. In Section 4 we describe the Mixed-Integer Surrogate Optimization (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 that combines local and global searches in order to find solutions of higher accuracy. In Section 5, we compare various algorithms that follow the MISO framework with SO-MI (Müller et al, 2013b), NOMAD (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 very efficient when the goal is to find good solutions within very few function evaluations. Section 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 $\phi(r) = r^3$), $\mathbf{z}_\iota, \iota = 1, \ldots, n$, 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 $\mathbf{b} = [b_1, \dots, b_d]^T \in \mathbb{R}^d$ 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 $\Phi_{\iota\nu} = \phi(\|\mathbf{z}_\iota - \mathbf{z}_\nu\|)$, $\iota, \nu = 1, \dots, n$, $\mathbf{0}$ 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 $\text{rank}(\mathbf{P}) = d + 1$ (Powell, 1992).

3 Review of Surrogate Model Algorithms

3.1 General Surrogate Model Algorithm

Surrogate model based optimization algorithms consist in general of the following steps:

Algorithm 1 General Surrogate Model Algorithm

```
1: Create an initial experimental design and do the expensive objective function evaluations at the selected points.
2: Fit the chosen surrogate model to the data in Step 1.
3: Use the information from the surrogate model to select the point $\mathbf{z}_{\text{new}}$ for doing the next expensive function evaluation.
4: Do the expensive evaluation at $\mathbf{z}_{\text{new}}$: $f_{\text{new}} = f(\mathbf{z}_{\text{new}})$.
5: if Stopping criterion is not met then
6:     Update the surrogate model and go to Step 3.
7: else
8:     Return the best solution found during the optimization.
9: end if
```

In Step 1, an initial experimental design is created and the computationally expensive objective function is evaluated at the selected points. In general, any initial design strategy may be used, but it has to be ensured that there are sufficiently many points to fit the chosen surrogate model in Step 2. The objective function value predictions of the surrogate model at unsampled points are used in Step 3 to select the next evaluation point. After the new function value has been obtained in Step 4, the surrogate model is updated in Step 6 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 in Step 8.

This 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 in Step 2;
– the method for selecting a new evaluation point in Step 3.

3.2 Previous Surrogate Model Algorithms for Continuous Optimization

Several surrogate model algorithms have been introduced in the literature for addressing computationally expensive black-box optimization problems with continuous variables. The EGO algorithm (Efficient Global Optimization) 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 so-called 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 and 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, 2014; Müller et al, 2013b) 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 adaptive 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.

4 MISO Framework

The algorithms for continuous optimization briefly reviewed in Section 3.2 can be easily modified for mixed-integer optimization problems. Only Steps 1 and 3 of Algorithm 1 (the initial experimental design and the selection of new sample points) have to be adjusted. The goal is to find a near-optimal solution within a very restricted 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 which 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 shown in Algorithm 1 can thus be modified to the MISO (Mixed-Integer Surrogate Optimization) framework shown in Algorithm 2.

Algorithm 2 General MISO Framework

```
1: 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.
2: Fit the chosen surrogate model to the data in Step 1.
3: 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.
4: Do the expensive evaluation at $\mathbf{z}_{\text{new}}$: $f_{\text{new}} = f(\mathbf{z}_{\text{new}})$.
5: if Stopping criterion is not met then
6:     Update the surrogate model and go to Step 3.
7: else
8:     Return the best solution found during the optimization.
9: end if
```

In Step 1 of Algorithm 2, when creating the initial experimental design, we create only 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 2. When fitting the surrogate model, we assume that all variables are continuous in order to obtain a smooth surface. However, in Step 3 we have to guarantee that each newly selected evaluation point satisfies the integer constraints.

4.1 Modifications of Continuous Surrogate Model Algorithms for Mixed-Integer Problems

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

– MISO-CS (MISO - Coordinate Search): 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.

For algorithms that follow a stochastic sampling approach such as MISO-CS and MISO-RS, we generate integer-feasible random sample points. MISO-CS 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 $\rho \sim \mathcal{N}(0, 1)$ and σ denotes the perturbation radius of the current iteration. Thus, the integer perturbation is at least one unit. When perturbing the continuous variables of $\mathbf{z}_{\text{best}}$, we add $\sigma\rho$ to the value.

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 optimizing 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.

4.2 MISO-CSTV and MISO-CSTV-local

We developed two new algorithms, namely MISO-CSTV and MISO-CSTV-local, that combine a coordinate search (stochastic sampling) with a target value strategy (minimizing an auxiliary objective function on the surrogate model). MISO-CSTV-local is a memetic algorithm that uses in addition a local search in order to improve the solution accuracy. The algorithms' steps are described in Algorithm 3. 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 $T_h^c = 5$.
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 $q(n) = \min\{20/d, 1\}(1 - \log(n - n_0 + 1)/\log(n_{\max} - n_0))$, where n denotes the number of function evaluations done so far.
11. A pattern for the target value strategy stage $G =< 0, 1, \ldots, 10, 11 >$.
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-CSTV-local only: a local optimization algorithm for continuous problems.

In Algorithm 3, we first initialize the counters C_f^c (for counting the number of consecutive failed improvement trials in the c-Step), C_s^c (for counting the number of consecutive successful improvement trials in the c-Step), and C_r^c (for counting the number of times we decreased the perturbation radius σ in the c-Step). We also initialize the counters I_c (for counting the iterations in the c-Step), C_f^t (the number of consecutive failed improvement trials in the t-Step), C_s^t (the number of consecutive successful improvement trials in the t-Step), and C_g^t (the iteration counter for the t-Step). We use the same approach for creating the initial experimental design as for the other MISO algorithms. 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. We fit a cubic RBF model with linear polynomial tail to the data and select $\mathbf{z}_{\text{new}}$ either by the coordinate search strategy (c-Step) or the target value strategy (t-Step) in MISO-CSTV (see Algorithms 4 and 5, respectively). In MISO-CSTV-local an additional l-Step may be used (a local search described in Algorithm 6). MISO-CSTV and MISO-CSTV-local both start with the c-Step.

4.2.1 c-Step: Coordinate Search

In the c-Step described in Algorithm 4, we generate a set of N candidates for the next sample point (Step 3) by adding random perturbations to randomly

Algorithm 3 MISO-CSTV/MISO-CSTV-local

```
1: Initialize $c$-Step $\leftarrow$ true, $t$-Step $\leftarrow$ false, $l$-Step $\leftarrow$ false.
2: Initialize counters $C_f^c \leftarrow 0, C_s^c \leftarrow 0, C_r^c \leftarrow 0$, $I_c \leftarrow 0$, $C_f^t \leftarrow 0, C_s^t \leftarrow 0$, and
   $C_g^t \leftarrow 1$.
3: 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., $\mathcal{Z} \leftarrow \{\mathbf{z}_1, \dots, \mathbf{z}_{n_0}\}$.
4: Set $n \leftarrow n_0$.
5: Fit the surrogate model $s_n(\cdot)$ to the data.
6: while Stopping criterion not met ($n < n_{\max}$) do
7:     Determine $\mathbf{z}_{\text{new}}$ as follows
8:     if $c$-Step then
9:         Update the $c$-Step iteration counter $I_c \leftarrow I_c + 1$.
10:        Use the coordinate search strategy described in Algorithm 4.
11:        Update $c$-Step, $t$-Step, and $l$-Step if necessary.
12:        Update $n \leftarrow n + 1$, $\mathcal{Z} \leftarrow \mathcal{Z} \cup \mathbf{z}_{\text{new}}$.
13:    else if $t$-Step then
14:        Use the target value search strategy described in Algorithm 5.
15:        Update $c$-Step and $t$-Step if necessary.
16:        Update $n \leftarrow n + 1$, $\mathcal{Z} \leftarrow \mathcal{Z} \cup \mathbf{z}_{\text{new}}$.
17:    else if $l$-Step (for MISO-CSTV-local only) then
18:        Use the local search strategy described in Algorithm 6.
19:        Update $c$-Step and $l$-Step if necessary.
20:        Update $n \leftarrow n + n_l$, $\mathcal{Z} \leftarrow \mathcal{Z} \cup \mathcal{Z}_l$, where $n_l$ denotes the number
   of function evaluations done by the local search and $\mathcal{Z}_l$ denotes the set of
   points evaluated during the local search.
21:    end if
22:    Update the surrogate model with the new data.
23: end while
```

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 $v_i \sim \mathcal{U}(0, 1)$ is drawn for each variable $i = 1, \dots, d$. If $v_i < q(n)$ (the perturbation probability computed in Step 2), we add a random perturbation to that variable. 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 4). 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 5) and scale these values to [0,1] such that points far away from

Algorithm 4 c-Step (Coordinate Search Step)

```
1: Determine the best point found so far: $\mathbf{z}_{\text{best}} = \arg\min\{f(\mathbf{z}), \mathbf{z} \in \mathcal{Z}\}$, $f_{\text{best}} = f(\mathbf{z}_{\text{best}})$.
2: Determine the probability $q_n = q(n)$ of perturbing each variable of $\mathbf{z}_{\text{best}}$.
3: Create a set of $N$ candidate points by perturbing each variable of $\mathbf{z}_{\text{best}}$ with probability $q_n$ by adding $\sigma\rho$ for continuous variables and $\text{sgn}(\rho)\max\{1, |\sigma\rho|\}$ for integer variables. If no variable is selected for perturbation by using probability $q_n$, randomly select one variable.
4: 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, $S_s$).
5: Compute the distance of each candidate point to the set $\mathcal{Z}$ and scale the values to [0,1] (distance score, $S_d$).
6: 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 ($\mathbf{z}_{\text{new}}$).
7: Do the expensive function evaluation $f_{\text{new}} = f(\mathbf{z}_{\text{new}})$.
8: if $f_{\text{best}} < f_{\text{new}}$ (no improvement found) then
9:     Update counters $C^c_f \leftarrow C^c_f + 1, C^c_s \leftarrow 0$.
10:    if $C^c_f > T^c_f$ then
11:        if $C^c_r > T^c_r$ then
12:            Set $c$-Step ← false, $t$-Step ← true, reset $C^c_r \leftarrow 0$, $C^c_f \leftarrow 0$.
13:        end if
14:    else
15:        Update $C^c_r \leftarrow C^c_r + 1, \sigma \leftarrow \max\{\sigma_l, \sigma/2\}, C^c_f \leftarrow 0$.
16:    end if
17: else
18:    Update $f_{\text{best}} \leftarrow f_{\text{new}}$, $\mathbf{z}_{\text{best}} \leftarrow \mathbf{z}_{\text{new}}$. Update $C^c_s \leftarrow C^c_s + 1, C^c_f \leftarrow 0$.
19:    if $C^c_s > T^c_s$ then
20:        Update $\sigma \leftarrow \min\{\sigma_u, 2\sigma\}$. Update $C^c_s \leftarrow 0$.
21:    end if
22: end if
```

$\mathcal{Z}$ (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 6. 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

$$w_d = \begin{cases} \mathcal{W}[k] & \text{if } k \equiv I_c \bmod |\mathcal{W}| > 0 \\ \mathcal{W}[|\mathcal{W}|] & \text{otherwise} \end{cases}, \tag{7}$$

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. With large values for w_s, preference is given to candidates that have low predicted objective function values. In this case the search tends to be more local since low predicted objective function values are likely to be in the vicinity of $\mathbf{z}_{\text{best}}$. For large values of w_d, preference is given to points that are in rather unexplored regions of the variable domain (global search). By repeatedly cycling through the weight pattern $\mathcal{W}$, a repeated transition from local to global search is achieved, and thus the algorithm is able to escape from local minima (Regis and Shoemaker, 2007).

The computationally expensive objective function is evaluated at the newly selected point ($f_{\text{new}} = f(\mathbf{z}_{\text{new}})$) in Step 7 and depending on whether or not f_{new} is better than f_{best}, the counters C_f^c, C_s^c, and C_r^c are updated as well as f_{best} (see Steps 8-22). 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 11-13).

4.2.2 t-Step: Target Value Search

The t-Step is described in Algorithm 5 and has three different cases (Steps 2-17) in which an auxiliary optimization problem is solved to determine the next sample point (see also (Holmström, 2008a, Algorithm RBF)). 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. The step type g is defined by

$$g = \begin{cases} G[k] & \text{if } k \equiv C_g^t \bmod |G| > 0 \\ G[|G|] & \text{otherwise} \end{cases}. \tag{8}$$

In the "Inf-Step" (Steps 2-3), we select as new evaluation point

$$\mathbf{z}_{\text{new}} = \arg\min_{\mathbf{z}\in\mathcal{D}} \mu_n(\mathbf{z}), \tag{9}$$

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

$$\begin{bmatrix} \boldsymbol{\Phi}_z & \mathbf{P}_z \\ \mathbf{P}_z^T & \mathbf{0} \end{bmatrix} \mathbf{v} = \begin{bmatrix} \mathbf{0}_n \\ 1 \\ \mathbf{0}_d \end{bmatrix}, \tag{10}$$

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

$$\boldsymbol{\Phi}_z = \begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{\phi}_z \\ \boldsymbol{\phi}_z^T & 0 \end{bmatrix}, \mathbf{P}_z = \begin{bmatrix} \mathbf{P} \\ \mathbf{z}^T \ 1 \end{bmatrix}, \text{ and } (\boldsymbol{\phi}_z)_i = \phi(\|\mathbf{z} - \mathbf{z}_i\|), i = 1, \ldots, n. \tag{11}$$

Algorithm 5 t-Step (Target Value Search Step)

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

In the "Cycle step - global search" (Steps 4-8), the goal is to first find the minimum point of the surrogate surface (Step 5):

$$\mathbf{z}_\text{s} = \arg\min_{\mathbf{z} \in \mathcal{D}} s_n(\mathbf{z}), \tag{12}$$

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 6-7), we determine

$$\mathbf{z}_{\text{new}} = \arg\min_{\mathbf{z} \in \mathcal{D}} \mu_n(\mathbf{z}) \left[s_n(\mathbf{z}) - t \right]^2, \tag{13}$$

where $\mu_n(\mathbf{z})$ is defined as for (9).

In the "Cycle step - local search" (Steps 13-16), we first find the minimum of the surrogate surface by solving (12) (Step 10). 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 11-12). Otherwise, we define a target value that corresponds to a 1% improvement of the best function value found so far and we solve (13) (Steps 14-15).

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 18-20). 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 22). If the new function value is not an improvement of f_{best}, we update the fail and success counters C_f^t and C_s^t, respectively (Steps 23-24). If the fail counter exceeds the threshold of allowable failed improvement trials, we leave the t-Step and go back to the c-Step (Steps 25-26). If we found an improvement of the best solution encountered so far, we update f_{best}, C_s^t, and C_f^t (Steps 28-30).

4.2.3 l-Step: Local Search

The local search step is only used in MISO-CSTV-local. While MISO-CSTV alternates only between the c-Step and the t-Step until the maximum number of function evaluations has been reached, MISO-CSTV-local enters a local search phase whenever the sequence $<$ c-Step, t-Step, c-Step $>$ did not lead to any improvement. 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:

$$\mathbf{z}_l = \arg\min_{\mathbf{z}\in\mathcal{D}}\{f(\mathbf{z}|z_\iota), z_\iota = z_{\text{best},\iota}\forall\iota \in \mathbb{I}\}, \tag{14}$$

where $\mathbb{I}$ denotes the indices of the integer variables and $z_{\text{best},\iota}$ denotes the ιth variable of $\mathbf{z}$. 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 (Steps 3-4). If the budget of allowed function values has not been used up 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 (14) 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:     Update $f_{\text{best}} \leftarrow f_l$, $\mathbf{z}_{\text{best}} \leftarrow \mathbf{z}_l$.
5: end if
6: Set $c$-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-CSTV-l(f) and the algorithm that uses ORBIT we call MISO-CSTV-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 Section 4 to SO-MI (Müller, 2014; Müller et al, 2013b), NOMAD (Nonlinear Optimization by Mesh Adaptive Direct Search) (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 applicable to mixed-integer problems. 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).

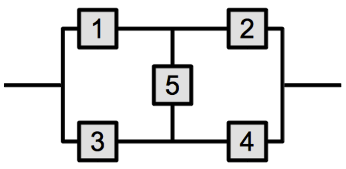
The goal of this paper is to develop efficient algorithms that are able to find near optimal solutions for computationally expensive optimization problems within very few function evaluations. We limit the number of allowed function evaluations to 500 for all test problems since in practice often only 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 majorly 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 average out the random component.

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 in the initial design depends on the number of variables ($2(d+1)$) and is generally not equal to 20.

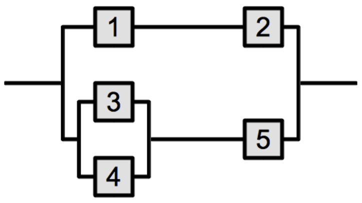
5.2 Test Problems

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, but 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 problems. 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. Each component of type j has the reliability r_j. There are different system configurations such as, for example, the bridge network or the series-parallel system shown in Figure 1.



(a) Bridge network with five components.



(b) Series parallel system with five components.

Fig. 1: Examples of system configurations in reliability redundancy optimization

There are two possibilities to increase the reliability of a system. We can either increase each component's reliability r_j (continuous variables) or we can add redundancy by adding a component of type j (integer variables). The component costs increase exponentially after r_j exceeds a certain threshold, and thus it may be cheaper to include components of lower reliability but to have several backup components. Hence, there is a trade-off between increasing component reliability and adding redundancy.

The second application problem we consider 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 1 gives an overview over the test problems we use for comparing the algorithms. 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.

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-CSTV-l(f) (more than 120 hours versus 0.2 hours). Since MISO-EI does not appear to be efficient, we only examined

Table 1: 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	$\{1,10\}^{24} \times [0,1000]^7$

its performance for the first five test problems. The results are summarized in Table 2 where the average best solution over 20 trials found by each algorithm is shown. The results for these test problems show that MISO-EI is not very promising and performs worst for two of the problems. 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 comparison.

For the remaining algorithms we summarize the results of the numerical experiments in form of data and performance profiles as suggested by Moré and Wild (2009). We use the MATLAB codes provided on `http://www.mcs.anl.gov/~more/dfo/` for creating Figures 2 and 3. We create the profile plots based on the average objective function value found over all 20 trials by each algorithm.

Denote in the following 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)

$$r_{\gamma,a} = \frac{l_{\gamma,a}}{\min\{l_{\gamma,a} : a \in \mathcal{A}\}}. \tag{15}$$

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

$$\rho_a(\alpha) = \frac{1}{|\mathcal{P}|}\text{size}\{\gamma \in \mathcal{P} : r_{\gamma,a} \leq \alpha\}. \tag{16}$$

Table 2: Comparison of the best objective function value found by MISO-EI (expected improvement sampling) for test problems 1-5 to all other algorithms.

Problem ID Algorithm	1	2	3	4	5
MISO-CS	-9.8457	0.0128	-484.2190	-0.4978	-43.1015
MISO-CSTV-l(f)	-11.2390	0.0005	-506.6590	-0.5000	-43.1343
MISO-CSTV-l(o)	-11.1453	0.0004	-503.6524	-0.5000	-43.1343
MISO-CSTV	-11.1931	0.0046	-504.9472	-0.4996	-43.1235
MISO-TV	-7.9072	0.0764	-473.4385	-0.4999	-43.1328
MISO-SM	-10.5764	0.0100	-418.3756	-0.5000	-43.1325
MISO-RS	-7.4715	6.7507	-409.0911	0.5099	-39.1662
MISO-EI	-5.6389	4.1557	-331.2325	-0.4991	-43.1256
SO-MI	-8.1837	0.0222	-474.1491	-0.5000	-43.1343
GA	-8.0020	15.8860	-477.0198	-0.2990	-40.5000
NOMAD	-11.3336	0.0001	-477.3689	-0.5000	-43.1343

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

$$\delta_a(\kappa) = \frac{1}{|\mathcal{P}|}\text{size}\{\gamma \in \mathcal{P} : \frac{l_{\gamma,a}}{d_\gamma + 1} \leq \kappa\}, \tag{17}$$

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

Figure 2 shows data profiles for all algorithms for two levels of accuracy, namely $\tau = 10^{-1}$ 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 Figures 2 and 3. For example, CSTV stands for MISO-CSTV, SM stands for MISO-SM, 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-CSTV, MISO-CSTV-l(o), and MISO-CSTV-l(f) outperform the other algorithms. In fact, there is no difference between the performance of MISO-

CSTV, MISO-CSTV-l(o), and MISO-CSTV-l(f). Similarly, for the accuracy $\tau = 10^{-3}$, MISO-CSTV-l(o) and MISO-CSTV-l(f) find better solutions than the other algorithms. MISO-CSTV-l(o) and MISO-CSTV-l(f) perform better than MISO-CSTV, 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, for example $\tau = 10^{-3}$, we can see that GA performs worst among all algorithms.

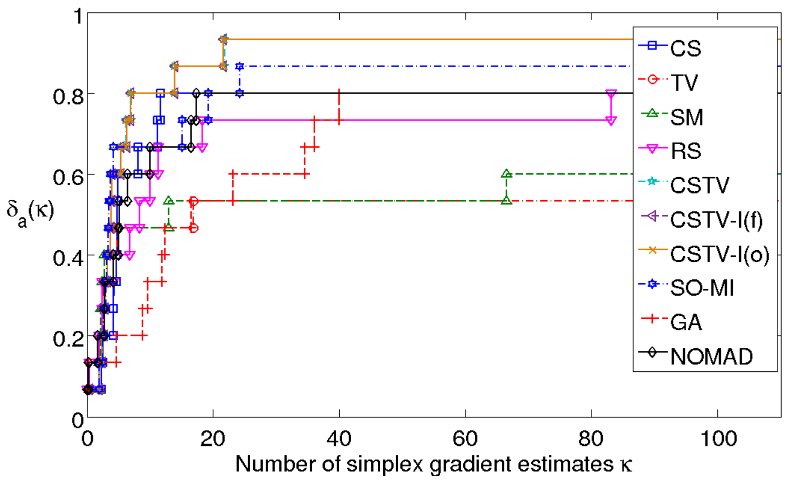
The performance profiles in Figure 3 show similar results. MISO-CSTV, MISO-CSTV-l(o), and MISO-CSTV-l(f) perform equally well for $\tau = 10^{-1}$, whereas MISO-CSTV-l(f) performs better than all other algorithms for $\tau = 10^{-3}$. Figure 3(b) shows, for example, that for the performance ratio of $\alpha = 4$ there is a performance difference between NOMAD and MISO-CSTV-l(f) of about 20%, which means that for 20% of the problems, NOMAD needs four times as many function evaluations to reach the same accuracy as MISO-CSTV-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-CSTV, MISO-CSTV-l(o), MISO-CSTV-l(f)) perform better than the algorithms that use only one sampling method (MISO-SM, MISO-RS, MISO-TV). We can also see that, similar to the results for continuous problems reported in Regis and Shoemaker (2013), the coordinate search strategy (MISO-CS), 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 of the best point found so far. In comparison to our previous algorithm SO-MI, the results show that MISO-CSTV, MISO-CSTV-l(o), and MISO-CSTV-l(f) are an improvement.

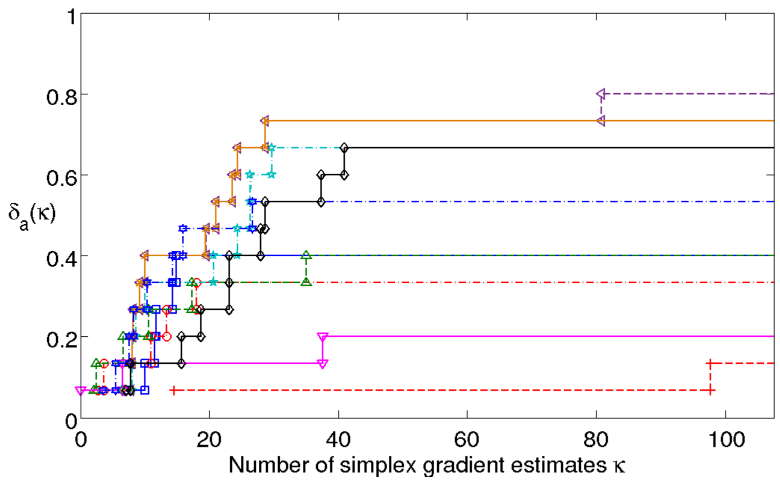
The comparison of MISO-CSTV-l(f) and MISO-CSTV-l(o) shows that for the low accuracy $\tau = 10^{-1}$ both versions perform equally well. For the higher accuracy $\tau = 10^{-3}$, MISO-CSTV-l(f) performs slightly better, indicating that for our approach of fixing the integer variables and locally searching 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 MISO (Mixed-Integer Surrogate Optimization) 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.



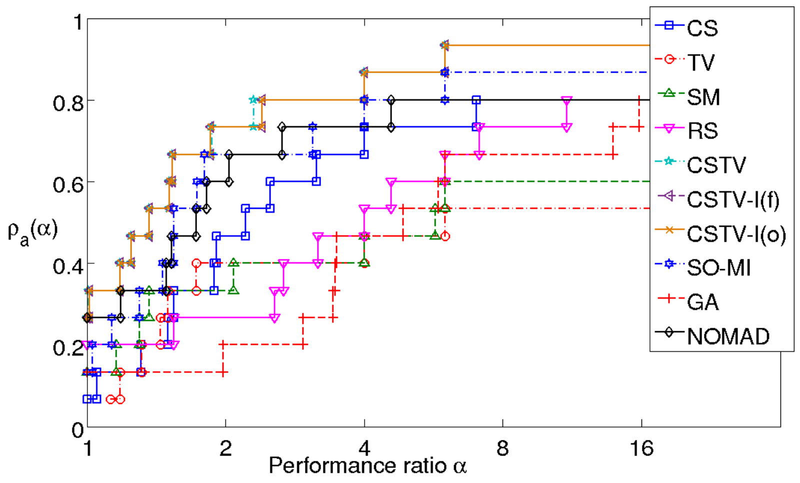
(a) Data profile for accuracy level $\tau = 10^{-1}$.



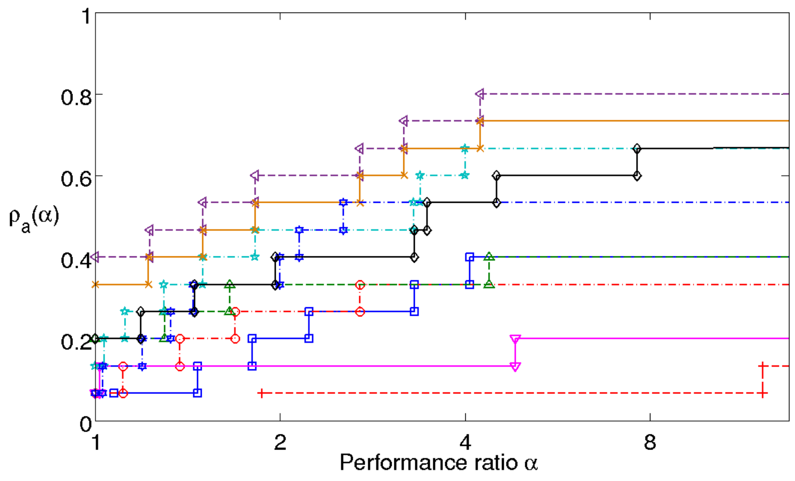
(b) Data profile for accuracy level $\tau = 10^{-3}$.

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

The MISO framework ensures that by generation all sample points satisfy the integer constraints, and thus no computationally expensive function evaluations are wasted by 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



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



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

Fig. 3: Performance profiles. Both figures share the same legend. The algorithms following the MISO framework are abbreviated with their sampling strategies.

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-CSTV that combines a dynamic coordinate search with a target value strategy, and MISO-CSTV-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 (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-CSTV and MISO-CSTV-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 search strategy, one has to guarantee that all candidate points 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.

Acknowledgements 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

Booker A, Dennis Jr J, Frank P, Serafini D, Torczon V, Trosset M (1999) A rigorous framework for optimization of expensive functions by surrogates. Structural Multidisciplinary Optimization 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. Journal of Global Optimization 43:191–205

Forrester A, Sóbester A, Keane A (2008) Engineering Design via Surrogate Modelling - A Practical Guide. Wiley

Friedman J (1991) Multivariate adaptive regression splines. The Annals of Statistics 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. Aeronautical Journal 101:347–356

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

Goel T, Haftka RT, Shyy W, Queipo NV (2007) Ensemble of surrogates. Structural Multidisciplinary Optimization 33:199–216

Gutmann H (2001) A radial basis function method for global optimization. Journal of Global Optimization 19:201–227

Holmström K (2008a) An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization. Journal of Global Optimization 41:447–464

Holmström K (2008b) An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer global optimization. Journal of Global Optimization 9:311–339

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

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

Le Digabel S (2011) Algorithm 909: NOMAD: Nonlinear optimization with the mads algorithm. ACM Transactions on Mathematical Software 37

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

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

Müller J (2014) MATSuMoTo: The MATLAB Surrogate Model Toolbox for Computationally Expensive Black-Box Global Optimization Problems. arXiv:14044261

Müller J, Piché R (2011) Mixture surrogate models based on Dempster-Shafer theory for global optimization problems. Journal of Global Optimization 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. Journal of Global Optimization pp DOI: 10.1007/s10,898–014–0184–0

Müller J, Shoemaker C, Piché R (2013a) SO-I: a surrogate model algorithm for expensive nonlinear integer programming problems including global optimization applications. Journal of Global Optimization pp 10.1007/s10,898–013–0101–y, journal of Global

Müller J, Shoemaker C, Piché R (2013b) SO-MI: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. Computers and Operations Research 40:1383–1400

Myers R, Montgomery D (1995) Response Surface Methodology, Process and Product Optimization using Designed Experiments. Wiley-Interscience Publication

Powell M (1992) The Theory of Radial Basis Function Approximation in 1990. Advances in Numerical Analysis, vol. 2: wavelets, subdivision algorithms and radial basis functions. Oxford University Press, Oxford, pp. 105-210

Rashid S Kand Ambani, Cetinkaya E (2012) An adaptive multiquadric radial basis function method for expensive black-box mixed-integer nonlinear constrained optimization. Engineering Optimization p 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 Journal on Computing 19:497–509

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

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

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

Viana F, Haftka R, surrogates: how cross-validation errors can help us to obtain the best predictor VS (2009) Multiple surrogates: how cross-validation errors can help us to obtain the best predictor. Structural and Multidisciplinary Optimization 39:439–457

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

Wild S, Regis R, Shoemaker C (2007) ORBIT: Optimization by radial basis function interpolation in trust-regions. SIAM Journal on Scientific Computing 30:3197–3219