Computers & Operations Research 40 (2013) 1383–1400

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

SO-MI: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems

Juliane Müller [a,b,*], Christine A. Shoemaker [b], Robert Piché [a]

[a] Tampere University of Technology, Department of Mathematics, P.O. Box 553, 33101 Tampere, Finland
[b] Cornell University, School of Civil and Environmental Engineering, School of Operations Research and Information Engineering, Center of Applied Mathematics, 220 Hollister Hall, Ithaca, NY 14853-3501, United States

ARTICLE INFO

Available online 11 September 2012

Keywords:
Surrogate model
Mixed-integer optimization
Multimodal
Black-box
Nonlinear
Global optimization
Radial basis functions
Derivative-free

ABSTRACT

This paper introduces a surrogate model based algorithm for computationally expensive mixed-integer black-box global optimization problems with both binary and non-binary integer variables that may have computationally expensive constraints. The goal is to find accurate solutions with relatively few function evaluations. A radial basis function surrogate model (response surface) is used to select candidates for integer and continuous decision variable points at which the computationally expensive objective and constraint functions are to be evaluated. In every iteration multiple new points are selected based on different methods, and the function evaluations are done in parallel. The algorithm converges to the global optimum almost surely. The performance of this new algorithm, SO-MI, is compared to a branch and bound algorithm for nonlinear problems, a genetic algorithm, and the NOMAD (Nonsmooth Optimization by Mesh Adaptive Direct Search) algorithm for mixed-integer problems on 16 test problems from the literature (constrained, unconstrained, unimodal and multimodal problems), as well as on two application problems arising from structural optimization, and three application problems from optimal reliability design. The numerical experiments show that SO-MI reaches significantly better results than the other algorithms when the number of function evaluations is very restricted (200–300 evaluations).

1. Introduction

Mixed-integer optimization problems naturally arise in many application areas such as logistics, engineering design, portfolio optimization or energy generation. These problems are in general NP-hard and difficult to solve. Most algorithms for solving mixed-integer optimization problems are based on branch and bound methods, or some evolutionary strategy such as genetic algorithms [1–3] or ant colony optimization (for example MIDACO [4]). While for solving problems with convex objective and constraint functions several software packages are available (for example DICOPT [5] or BONMIN [6]), algorithms for solving nonconvex problems are rather scarce and are usually based on problem reformulation and convexification strategies [7–9]. However, when dealing with black-box objective and constraint functions, where all that is known about the problem is the deterministic output for a given input, reformulating the problem is not possible, and other methods must be developed. Moreover, if a single function evaluation requires a time consuming simulation (from several minutes to several hours or even days), the number of function evaluations for finding a good approximation of the global optimum has to be as low as possible.

The branch and bound algorithm is based on recursively dividing the set of possible solutions into smaller sets (nodes) during the so-called branching step, and a tree structure evolves. During the bounding step an upper and lower bound on the objective function value is calculated for every solution subset. This requires minimizing the costly objective function at least once because in order to obtain the lower bound, a relaxed problem (where certain integer variables are assumed to be continuous) must be minimized. For solving computationally expensive black-box functions, however, branch and bound might not be suitable because to obtain valid lower bounds for multimodal problems the global optimum of the relaxed problem must be found, which requires the application of a global optimization algorithm. Furthermore, some application problems may not allow a relaxation of the integer variables because this may cause the code evaluating the objective function to fail.

Evolutionary algorithms such as genetic algorithms [10] mimic the natural process of evolution by survival of the fittest. An initial population of individuals (solutions) is generated randomly.

* Corresponding author. Permanent address: Tampere University of Technology, Department of Mathematics, P.O. Box 553, 33101 Tampere, Finland.
Tel.: +358 40 8490 580.
E-mail address: juliane.mueller@tut.fi (J. Müller).

0305-0548/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2012.08.022

Nomenclature

$\mathbf{z}$	mixed-integer decision variables, see Eq. (5)
$\mathbf{x}$	continuous decision variables, see Eq. (5)
$\mathbf{u}$	discrete decision variables, see Eq. (5)
i_1	index for continuous variables, see Eq. (3)
i_2	index for discrete variables, see Eq. (4)
$f(\mathbf{z})$	objective function value, see Eq. (1)
y_i	ith objective function value
$\mathcal{Z}_n$	the set of n already evaluated points
$f_p(\mathbf{z})$	objective function value augmented with penalty term, see Eq. (10)
$\tilde{f}_p(\mathbf{z})$	objective function value augmented with penalty term, see Eq. (12)
$c_j(\mathbf{z})$	the jth constraint function, see Eq. (2)
k_1	dimension of the continuous variables, see Eq. (3)
k_2	dimension of the discrete variables, see Eq. (4)
k	problem dimension ($k=k_1+k_2$)
Ω_b	box-constrained variable domain
Ω^D	discrete box-constrained variable domain
Ω^C	continuous box-constrained variable domain
Ω	variable domain defined by box-constraints, linear, nonlinear, and integrality constraints
$s_f(\mathbf{z})$	radial basis function interpolant, see Eq. (6)
$\mathbf{z}_{\min}$	best feasible point found so far
$f_{\min}$	objective function value of the best feasible point found so far
$f_{\max}$	objective function value of the worst feasible point found so far
n	number of already sampled points
n_0	number of points in initial experimental design

The fittest individuals are chosen for reproduction. Their offspring are generated by random crossover and mutation operations. The advantage of this algorithm type is that it is possible to escape from a local optimum. In general, a large number of function evaluations must be done to find a good approximation of the global optimum due to the number of individuals in the population and the number of generations. Thus, genetic algorithms may not be suitable for solving computationally expensive optimization problems.

Another derivative-free algorithm often referred to in the literature is NOMAD (Nonsmooth Optimization by Mesh Adaptive Direct Search) [11–13]. NOMAD uses a mesh adaptive direct search algorithm designed for solving constrained black-box optimization problems. The mesh adaptive direct search is an extension of generalized pattern search algorithms for which superior convergence properties can be shown [14].

NOMAD is applicable for mixed-variable problems, and by using the variable neighborhood search [15,16] option in the C++ implementation, it is able to escape from local minima [17]. The mixed-variable pattern search can be shown to guarantee first-order optimality conditions with respect to the continuous variables, and local optimality in the mixed-integer case is defined by user-specified neighboring points [18]. Although the C++ implementation of NOMAD is able to use surrogate models during the search, the user has to implement the desired model him/herself.

Furthermore, there are no extensive numerical studies of using NOMAD for solving constrained mixed-integer problems. Liuzzi et al. [19] compared their derivative-free algorithms to NOMAD on many low-dimensional mixed-integer local optimization problems and few global optimization test problems with at most 20 dimensions that have only bound constraints, but application problems were not considered in [19].

Recently, surrogate model based algorithms have been a focus of interest in the global optimization literature [20–28]. Surrogate models have proved successful for finding the global optima of computationally expensive continuous optimization problems, and have been employed for solving real world application problems [29–32].

Despite their success in solving continuous optimization problems, surrogate model algorithms have only started to be considered for solving mixed-integer optimization problems [33–36]. Davis and Ierapetritou [33] suggest an algorithm that is able to handle problems with noisy data, and that have continuous and binary variables. A branch and bound framework is coupled with Kriging response surfaces and response surface methodology to obtain a balance of local and global search while using a very restricted number of function evaluations.

Holmström et al. [35] describe in their paper an adaptive radial basis function algorithm for solving mixed-integer optimization problems. The algorithm uses mixed-integer subsolvers from the commercial TOMLAB optimization environment to solve an auxiliary problem to determine the points where to evaluate the expensive objective function in every iteration of the algorithm. Holmström et al. [35] considered mainly low dimensional problems (one test problem with 11 dimensions and eight integer variables, and other test problems with six or fewer dimensions with one to four integer variables). The response surface is not necessarily unimodal, and solving the auxiliary problem may require itself the application of a global optimization algorithm. Furthermore, constraints have to either be computationally cheap, or otherwise added as penalty term to the objective function value. The adjustment of the penalty factor is however a delicate task, and there are no recommendations on how to set it.

Similarly, Rashid et al. [36] use a MINLP subsolver to solve two auxiliary optimization problems for determining the next sample site(s). In addition to the computationally expensive objective function, Rashid et al. [36] consider problems that may have computationally inexpensive as well as expensive constraints. Although a multistart method is applied when optimizing the auxiliary problems, there is no guarantee that the actual global optimum of the subproblems will be found, and the chosen sample points may therefore be derived from local optima of the auxiliary problems.

The present work is intended as a contribution to the new area of applying surrogate models for solving larger dimensional mixed-integer computationally expensive black-box problems *without* using mixed-integer subsolvers, and where the integer variables may take on a wide range of values rather than only binary values. A theoretical analysis of convergence properties is also presented.

The remainder of this paper is organized as follows. The problem formulation is given in Section 2, and the concept of surrogate models using radial basis functions is briefly outlined in Section 3. The new algorithm, SO-MI, for solving constrained mixed-integer optimization problems using surrogate models is described in Section 4, and its theoretical properties are described in Theorem 4.1. In Section 5 the set up of the numerical experiments is described. The performance of SO-MI is compared to a genetic algorithm, a branch and bound algorithm, and NOMAD on 16 test problems from the literature, and five application problems that are briefly described in Section 6. A more thorough description of the test problems is given in

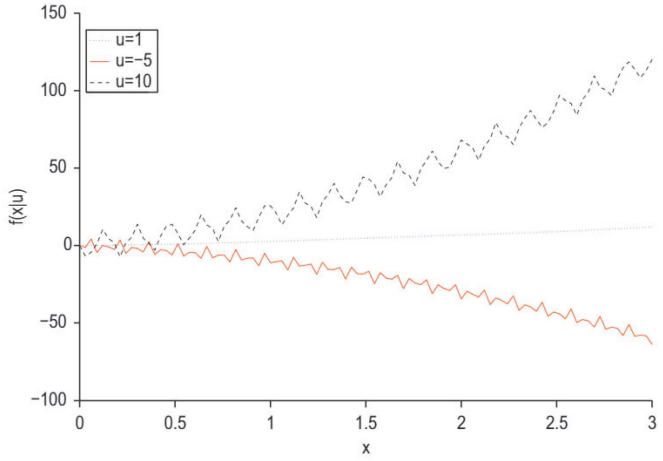

Fig. 1. Illustrated is the function $f(x,u)=ux+u\sin^3(u^3x)+ux^2$ for three different fixed values of u. The function may be multimodal depending on the discrete variable values.

the Online Supplement. The numerical results are summarized in Section 7, and Section 8 concludes the paper.

2. Mixed-integer optimization problem

In the following let $f:\mathbb{R}^{k_1}\times\mathbb{Z}^{k_2}\mapsto\mathbb{R}$ denote the costly black-box objective function that may in general be nonlinear or multimodal. Denote the decision variables by $\mathbf{z}^T=(\mathbf{x}^T,\mathbf{u}^T)$, where $\mathbf{x}\in\mathbb{R}^{k_1}$ denote the continuous variables, and $\mathbf{u}\in\mathbb{Z}^{k_2}$ denote the discrete variables. The optimization problem considered in this paper is

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

$$\text{s.t.}\quad c_j(\mathbf{z})\le 0\ \forall j=1,\dots,m \tag{2}$$

$$-\infty<x_{i_1}^l\le x_{i_1}\le x_{i_1}^u<\infty\ \forall i_1=1,\dots,k_1 \tag{3}$$

$$-\infty<u_{i_2}^l\le u_{i_2}\le u_{i_2}^u<\infty\ \forall i_2=1,\dots,k_2 \tag{4}$$

$$\mathbf{x}\in\mathbb{R}^{k_1},\quad \mathbf{u}\in\mathbb{Z}^{k_2},\quad \mathbf{z}^T=(\mathbf{x}^T,\mathbf{u}^T) \tag{5}$$

where $x_{i_1}^l$ and $x_{i_1}^u$ denote the lower and upper bounds on the continuous variables x_{i_1}, $i_1=1,\dots,k_1$, respectively, and where $u_{i_2}^l$ and $u_{i_2}^u$ denote the lower and upper bounds on the discrete variables u_{i_2}, $i_2=1,\dots,k_2$, respectively. The jth computationally expensive constraint is denoted by $c_j(\mathbf{z})$. Inequalities (3) and (4) are the box constraints. Denote the box-constrained discrete variable domain by $\Omega^D\subset\mathbb{Z}^{k_2}$, and the box-constrained continuous variable domain by $\Omega^C\subset\mathbb{R}^{k_1}$. The mixed-integer box-constrained variable domain is denoted by $\Omega_b=\Omega^C\times\Omega^D$, and it holds that $|\Omega^D|=\kappa<\infty$, i.e. Ω^D is a finite set. The dimension of the mixed-integer optimization problem is denoted by $k=k_1+k_2$. Throughout this paper it is assumed that every point $\mathbf{z}$ satisfies the integrality constraints imposed on $\mathbf{u}$, and that $f(\mathbf{z})$ and $c_j(\mathbf{z})$, $j=1,\dots,m$, are continuous when the discrete variables are fixed.

The objective functions of the optimization problems considered in this paper may in general look as illustrated in Fig. 1 (constraints left out for simplicity). Illustrated are the graphs of a function with one discrete and one continuous variable when the discrete variable is fixed to three different values. As can be seen, the function depends now only on the continuous variable and may be multimodal.

3. Surrogate models and radial basis functions

Surrogate models, also known as response surface models or metamodels, are approximations of simulation models [20]. Simulation models are used to capture the behavior of physical processes. A single simulation may however require several hours of computation time. Therefore, it is often useful to approximate the output of a computationally expensive simulation model with a surrogate model, i.e. $f(\mathbf{x})=s_f(\mathbf{x})+\epsilon(\mathbf{x})$, where $f(\mathbf{x})$ is the simulation output, $s_f(\mathbf{x})$ denotes the surrogate model output, and $\epsilon(\mathbf{x})$ is the difference between both. Surrogate models can be non-interpolating (for example multivariate adaptive regression splines [37], regression polynomials [38]), or interpolating (for example Kriging [24,39], radial basis functions [22,27,40]). Furthermore, mixture surrogate models [25] have appeared in the literature and can be, depending on the models in the mixture, either interpolating or non-interpolating. For a review of different approaches for using surrogate models in optimization see [23]. Surrogate models have been used, amongst others, for solving groundwater remediation problems and design optimization problems [27–32,41,42].

Denote in the following $\mathcal{Z}_n=\{\mathbf{z}_1,\dots,\mathbf{z}_n\}$ the set of n sample points where the objective function has been evaluated. The objective function values at these points are denoted by $y_1,\dots,y_n$. In this paper radial basis function (RBF) models are used as surrogate models. The RBF interpolant can be represented as

$$s_f(\mathbf{z})=\sum_{\iota=1}^{n}\lambda_\iota\phi(\|\mathbf{z}-\mathbf{z}_\iota\|)+p(\mathbf{z}) \tag{6}$$

where s_f denotes the response surface, $\phi:\mathbb{R}_+\mapsto\mathbb{R}$ denotes the radial basis function, and $p(\mathbf{z})=\mathbf{b}^T\mathbf{z}+a$ is the polynomial tail. Here, $\mathbf{b}=[b_1,\dots,b_k]^T\in\mathbb{R}^k$ and $a\in\mathbb{R}$. In this paper the cubic radial basis function $\phi(r)=r^3$ is used. The parameters λ_ι, $\iota=1,\dots,n$, b_i, $i=1,\dots,k$, and a are determined by solving a linear system of equations [22]:

$$\begin{bmatrix}\boldsymbol{\Phi} & \mathbf{P}\\ \mathbf{P}^T & \mathbf{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{\lambda}\\ \mathbf{c}\end{bmatrix}=\begin{bmatrix}\mathbf{F}\\ \mathbf{0}\end{bmatrix}, \tag{7}$$

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

$$\mathbf{P}=\begin{bmatrix}\mathbf{z}_1^T & 1\\ \mathbf{z}_2^T & 1\\ \vdots & \vdots\\ \mathbf{z}_n^T & 1\end{bmatrix},\quad \boldsymbol{\lambda}=\begin{bmatrix}\lambda_1\\ \lambda_2\\ \vdots\\ \lambda_n\end{bmatrix}\quad \mathbf{c}=\begin{bmatrix}b_1\\ b_2\\ \vdots\\ b_k\\ a\end{bmatrix},\quad \mathbf{F}=\begin{bmatrix}y_1\\ y_2\\ \vdots\\ y_n\end{bmatrix}. \tag{8}$$

The matrix in (7) is invertible if and only if $\text{rank}(\mathbf{P})=k+1$ [43].

4. Surrogate model algorithm for mixed-integer black-box optimization problems

4.1. General structure of surrogate model algorithms for continuous problems

Surrogate model algorithms work iteratively and have in general the same structure. Initially, several sample points are chosen for evaluating the costly objective function. Finding these sample points can, for example, be done by Latin hypercube sampling [21] or some other experimental design strategy [38]. Given that data, the parameters of the chosen response surface model are determined. Then, the next sample point where to evaluate the expensive objective function is chosen according to some strategy. Gutmann's strategy [22], for example, is based on

solving an auxiliary optimization problem where a 'bumpiness' measure is minimized. Jones et al. [24], on the other hand, maximize the expected improvement in order to decide where to do the next expensive function evaluation. Regis and Shoemaker [27] suggested a so-called candidate point strategy that does not require solving an auxiliary optimization problem. Note however that these are all strategies for continuous optimization problems.

After the next sample point has been determined and the expensive objective function evaluated at this point, the response surface parameters are updated. The surrogate model algorithm iteratively determines the next sample site and updates the response surface parameters until a predefined stopping criterion has been met. The following are the components of a general surrogate optimization method:

1. Build an initial experimental design and do expensive function evaluations.
2. Use the data from 1 to compute the parameters of the surrogate model.
3. Select the next sample point and do the expensive function evaluation at this point.
4. Update the surrogate model parameters.
5. Iterate through 3 and 4 until a predefined stopping criterion has been met.

The surrogate model algorithm presented in this paper has been developed for finding (near) optimal solutions of mixed-integer optimization problems. It will in the following be referred to as SO-MI (**S**urrogate **O**ptimization-**M**ixed-**I**nteger). The single steps of SO-MI are described in detail in the following sections.

4.2. The initial experimental design for SO-MI and penalty functions

The initial experimental design consists of $2k+1$ points that are generated using a symmetric Latin hypercube design where the integer constraints are satisfied by rounding the corresponding variable values to the closest integers.

It is assumed that one feasible point is known, and this point is added to the initial experimental design. In practice, it can be assumed that at least one feasible solution is known due to experience and a general understanding of the problem at hand. Thus, totally $n_0 = 2(k+1)$ points are in the initial experimental design, which is twice the minimum number of points required to fit the RBF model of dimension k. Note that there is a potential problem of the rank of the matrix $\mathbf{P}$ in (7) being less than $k+1$ when rounding the integer variables. This has not been a problem in the computational experiments, but, in order to circumvent this problem, the rank of the matrix $\mathbf{P}$ is computed, and if $\text{rank}(\mathbf{P}) < k+1$, a new initial experimental design is generated until $\text{rank}(\mathbf{P}) = k+1$. Next, the computationally expensive objective function and the constraints are evaluated at these points.

The constraints $c_j(\mathbf{z})$ in (2) are integrated during the optimization phase in the objective function by a penalty term. A constraint violation function

$$v(\mathbf{z}) = \sum_{j=1}^{m} [\max\{0, c_j(\mathbf{z})\}]^2 \tag{9}$$

is used, where it is assumed that $\mathbf{z} \in \Omega_b$, i.e. only points within the box-constrained domain are considered. If $v(\mathbf{z}) = 0$, the point $\mathbf{z}$ satisfies every constraint $c_j(\mathbf{z}), j = 1, \ldots, m$. The penalty for constraint violations is incorporated in two different ways. At the beginning of the algorithm when there are possibly only one or very few feasible points known, it is preferable to stay within the feasible region and explore it, i.e. to find further feasible points, in order to improve the accuracy of the response surface within the feasible region. To achieve this goal the following penalty augmented objective function is used

$$f_p(\mathbf{z}) = \begin{cases} f_{\max} + c_p v(\mathbf{z}) & \text{if } \mathbf{z} \text{ is not feasible} \\ f(\mathbf{z}) & \text{otherwise} \end{cases} \tag{10}$$

where c_p denotes the penalty factor and $f_{\max}$ is the worst feasible objective function value found so far. This definition guarantees that the penalty augmented function values of the infeasible points are larger than the worst feasible objective function value. Otherwise, if defining the penalty function for example by $f(\mathbf{z}) + c_p v(\mathbf{z})$, it may be possible that for a constant penalty factor c_p infeasible points reach despite penalty a better objective function value than the feasible points, and the search might be drawn into the infeasible region and many unnecessary objective function evaluations could be done. To prevent numerical instabilities, high function values f_p are replaced by the median. The data $(\mathbf{z}_i, f_p(\mathbf{z}_i))$, $i = 1, \ldots, n_0$, from the initial experimental design are then used to solve the linear system of Eq. (7) in Section 3 to obtain the response surface parameters.

Since the goal is to find accurate approximations of the global optimum within as few function evaluations as possible, it is desired to do only very few function evaluations at infeasible points. By adding large values to infeasible points as done in (10) the response surface is likely to predict high objective function values for points in the vicinity of infeasible points. The response surface is used to approximate $f_p(\mathbf{z})$, and the penalty term acts similarly to a barrier method with the goal to stay initially within the feasible region. The disadvantage is however that the accuracy of the response surface near the boundary of the feasible domain is decreased. To overcome this drawback the penalty term is changed after 100 function evaluations.[1]

The penalty augmented objective function used for the remaining function evaluations is defined as follows. First the sum of squared constraint violations (9) is scaled to the interval [0, 1]

$$v_s(\mathbf{z}) = \frac{v(\mathbf{z}) - v_{\min}}{v_{\max} - v_{\min}} \tag{11}$$

if $v_{\min} \neq v_{\max}$, and $v_s(\mathbf{z}) = 0$ otherwise (if $v_{\min} = v_{\max}$, then there are no infeasible points). Here $v_{\min} = \min\{v(\mathbf{z}_i), i = 1,2,\ldots,n\}$ and $v_{\max} = \max\{v(\mathbf{z}_i), i = 1,2,\ldots,n\}$. Thus, feasible points (including the points on the boundary) obtain the value $v_s = 0$, whereas points that violate the constraints obtain positive values for v_s depending on how much these points violate the constraints. The new penalty augmented objective function values are then defined by

$$\tilde{f}_p(\mathbf{z}) = f(\mathbf{z}) + v_s(\mathbf{z}) f_{\max}. \tag{12}$$

Thus, the worst feasible objective function value $f_{\max}$ is added to the function value of the point with the largest constraint violation, whereas only a fraction of $f_{\max}$ is added as penalty to the objective function values of points with lower constraint violations. This penalty function does not guarantee that infeasible points have worse objective function values than the best feasible point. The parameters of the response surface are then computed using the data $(\mathbf{z}_i, \tilde{f}_p(\mathbf{z}_i))$, and may predict better objective function values for infeasible points than for feasible points, and the probability of sampling points at the boundary of the feasible region is larger than when using (10).

[1] Numerical experiments showed that for the problems in Section 6 100 evaluations were sufficient to find several feasible points. The number may be changed depending on the size of the initial experimental design and the total number of allowed function evaluations.

4.3. Selecting the next sample site in SO-MI

Preliminary numerical experiments showed that a random sampling strategy for determining the next sample site is more successful than optimizing some auxiliary function based on the response surface. The response surface is in general not unimodal, and the efficiency of methods such as minimizing a bumpiness measure [22] or maximizing the expected improvement [24] are highly dependent on the subsolver used for finding the optimum of these auxiliary problems. The auxiliary problem itself may be a multimodal optimization problem, and if the global optimum of this problem is not found, then the next sample site may be derived from a local optimum or some other stationary point only. Therefore, a random sampling strategy is employed in this paper.

In every iteration of the algorithm four new sample sites are chosen for evaluating the expensive objective and constraint functions. These four points are determined by generating four groups of candidate points. The first group of points is generated by perturbing only the continuous variables of the best feasible point found so far (this point is in the following denoted by $\mathbf{z}_{\min}$). The discrete variable values of $\mathbf{z}_{\min}$ are kept constant. Randomly chosen continuous variables of $\mathbf{z}_{\min}$ are perturbed by randomly adding or subtracting small, medium and large perturbations. If $k>5$, then each variable is perturbed with a probability of $5/k$. Otherwise, every variable is perturbed. Considering high-dimensional problems, this is a practical approach because the goal of a local search is to stay within the vicinity of $\mathbf{z}_{\min}$, and perturbing each variable of, for example, a 30-dimensional problem may cause candidate points to be far away from $\mathbf{z}_{\min}$ [26]. By perturbing only the continuous variables it is possible to find the minimum of the function $f(\mathbf{x}|\mathbf{u}$ fixed), which is at least a local optimum of $f(\mathbf{z})$.

For generating the candidate points in the second group, the continuous variables of $\mathbf{z}_{\min}$ are kept constant, and only randomly chosen discrete variables are perturbed by randomly adding or subtracting small, medium or large discrete perturbations that depend on the minimum range of the variables, and that are at least one unit. Thus, it is possible to explore how much the objective function values change when transitioning from one integer variable vector to another (compare for example the three objective function values of the three graphs in Fig. 1 when the continuous variable is kept constant). If, for example, the global optima for each function $f(\mathbf{u}|\mathbf{x}$ fixed) are at the same point $\mathbf{x}$ for all $\mathbf{u}$, it is possible to find the global optimum of $f(\mathbf{z})$ very efficiently.

The candidate points in the third group are generated by perturbing randomly chosen discrete *and* continuous variables of $\mathbf{z}_{\min}$ by randomly adding or subtracting small, medium and large perturbations that are defined as for the first two groups. The candidate points in the fourth group are generated by uniformly sampling points from the whole box-constrained variable domain Ω_b, ensuring that every point in Ω_b has a positive probability of becoming a candidate and therefore also sample point. The integrality constraints are satisfied by rounding the corresponding variables to the closest integers. If any candidate point in the first three groups exceeds the lower or upper variable bounds $x^l_{i_1}, u^l_{i_2}$ or $x^u_{i_2}, u^u_{i_2}$ in (3) and (4), respectively, then these variable values are set to the value of the corresponding bound that has been exceeded.

The standard deviations of the perturbations of the continuous variables have been set to $h_1=\delta\cdot\min\{\min\{x^u_{i_1}-x^l_{i_1}, i_1=1,\ldots,k_1\},\min\{u^u_{i_2}-u^l_{i_2}, i_2=1,\ldots,k_2\}\}$, where $\delta=0.1$ for large perturbations, $\delta=0.01$ for medium perturbations, and $\delta=0.001$ for small perturbations. The standard deviations of the perturbations for the integer variables have been defined as $h_2=\max\{1,[h_1]\}$, where $[h_1]$ is the value of h_1 rounded to the closest integer. The random perturbations ζh_1 and ζh_2, where $\zeta\sim\mathcal{N}(0,1)$, are then added to the variables chosen to be perturbed, and integer variables are subsequently rounded to the closest integer value. Three standard deviations for the perturbations have been used in order to increase the variability of the generated candidate points.

Each group contains $500k$ candidate points, which is sufficient to create a large diversity of points in each group. The four groups have been generated with the incentive of obtaining points that are close to $\mathbf{z}_{\min}$ (local search), and points that are randomly selected from the variable domain (global search). A balance of local and global search can therefore be established, and it is possible to explore the vicinity of $\mathbf{z}_{\min}$ more thoroughly, as well as search globally for other promising regions of the variable domain.

If it is a priori known that the optimization problem is unimodal and that the location of the minimum is independent of the discrete variable values, it might be sufficient to use only candidate points from the third group, and later switching to using candidate points only from the first group to further refine the search since the perturbations are in general much smaller (perturbing an integer variable results in a perturbation of at least one unit). In general, it is however unknown how many local and/or global optima are present. The candidate points in the fourth group allow the exploration of the whole variable domain, and thus it is possible to escape from local optima.

In order to determine the "best" candidate point in each group, two scoring criteria [27] are used. The first criterion is derived from the objective function value predicted by the response surface. Denote $\boldsymbol{\chi}_j$ the jth candidate point, $j=1,\ldots,t$, in any of the four groups. For every candidate point j the function $s_f(\boldsymbol{\chi}_j)$ in Eq. (6) is evaluated, and $s_{\max}=\max_{j=1,\ldots,t}s_f(\boldsymbol{\chi}_j)$ and $s_{\min}=\min_{j=1,\ldots,t}s_f(\boldsymbol{\chi}_j)$ are determined, i.e. the maximum and the minimum of the predicted objective function value over all candidate points is determined. The score of the jth candidate point derived from the predicted function value is then calculated as

$$V_R(\boldsymbol{\chi}_j)=\frac{s_f(\boldsymbol{\chi}_j)-s_{\min}}{s_{\max}-s_{\min}} \tag{13}$$

if $s_{\min}\neq s_{\max}$, and $V_R(\boldsymbol{\chi}_j)=1$ otherwise. Note that this step is very fast because evaluating the response surface is computationally inexpensive.

The second criterion is derived from the distance of the candidate points to the set of already sampled points $\mathcal{Z}_n$. Denote D the Euclidean distance in $\mathbb{R}^k$. For every candidate point $\Delta(\boldsymbol{\chi}_j)=\min_{i=1,\ldots,n}D(\boldsymbol{\chi}_j,\mathbf{z}_i)$ is computed, where $\mathbf{z}_i\in\Omega_b$ is the ith already sampled point. Also, $\Delta_{\max}=\max_{j=1,\ldots,t}\Delta(\boldsymbol{\chi}_j)$ and $\Delta_{\min}=\min_{j=1,\ldots,t}\Delta(\boldsymbol{\chi}_j)$ are determined, i.e. the maximum and minimum of the distances of all candidate points to $\mathcal{Z}_n$. Then the score for the jth candidate point derived from the distance criterion is calculated as

$$V_D(\boldsymbol{\chi}_j)=\frac{\Delta_{\max}-\Delta(\boldsymbol{\chi}_j)}{\Delta_{\max}-\Delta_{\min}} \tag{14}$$

if $\Delta_{\min}\neq\Delta_{\max}$, and $V_D(\boldsymbol{\chi}_j)=1$ otherwise.

A weighted sum

$$W(\boldsymbol{\chi}_j)=w_RV_R(\boldsymbol{\chi}_j)+w_DV_D(\boldsymbol{\chi}_j) \tag{15}$$

of both criteria is then used to determine the best candidate in each group. Here, w_D is the weight for the distance criterion, w_R is the weight for the predicted function value criterion, and $w_R+w_D=1$. The weights are adjusted in a cycling manner, i.e. starting with a high weight for the distance criterion ($w_D=1$) and a low weight for the response surface criterion ($w_R=0$), the algorithm updates the weights in every iteration, slowly reducing the weight of the distance criterion and increasing the weight of the response surface criterion. After the weight for the distance

criterion has reached zero, it is reset to the initial value, and will be reduced in the following iterations anew.

If w_D is high, then the distance criterion is emphasized, and candidate points that are far away from $\mathcal{Z}_n$ are preferred. In this case the search is more global because the best candidate points will be in regions that are rather unexplored. On the other hand, if w_R is high, then the criterion based on the predicted objective function value will have more influence. The search becomes more local because candidate points that have low predicted objective function values will be preferred, and these points are often in the vicinity of $\mathbf{z}_{\min}$. By adjusting the weights in a cycling manner, a repeated transition from global to local search is achieved.

Based on the final score the best candidate point in each group is chosen for doing the next expensive objective and constraint function evaluations, i.e. one point from each of the four groups is chosen. It is assumed that objective and constraint function values are the output of the same computationally expensive black-box simulation model, i.e. whenever the objective function is evaluated, also the constraint function values are obtained. The four evaluations can thus be done in parallel because they are independent of each other. Of course, the parallelization can be extended to more than four points if one wishes to select more than four candidate points for doing function evaluations.[2] After the function values have been obtained, the four new points are added to $\mathcal{Z}_n$, and, depending on the stage of the algorithm, either f_p in Eq. (10) or $\tilde{f}_p$ in Eq. (12) is computed for each point in $\mathcal{Z}_n$. The parameters of the response surface (6) are updated by solving the system (7) using either the data $(\mathbf{z}_\iota, f_p(\mathbf{z}_\iota))$ or $(\mathbf{z}_\iota, \tilde{f}_p(\mathbf{z}_\iota))$, $\iota = 1,\ldots,n$ (depending on the stage of the algorithm), and the algorithm iterates through generating candidate points, calculating scores, and updating the response surface until a given maximal number of function evaluations has been reached.

4.4. SO-MI algorithm

The specific steps of the surrogate model algorithm SO-MI for computationally expensive mixed-integer black-box optimization problems is given below.

Algorithm 1. SO-MI

1. Repeatedly generate a large initial experimental Latin hypercube design with $2k+1$ points, round the discrete variables to the closest integers, and add a known feasible point to the design until $\text{rank}(\mathbf{P}) = k+1$, where $\mathbf{P}$ is defined in (8). Denote the points by $\mathbf{z}_1,\ldots,\mathbf{z}_{n_0}$, where $n_0 = 2(k+1)$.
2. Do the costly function evaluations to obtain $y_\iota = f(\mathbf{z}_\iota)$, and $c_j(\mathbf{z}_\iota), \iota = 1,\ldots,n_0,\ j = 1\ldots,m$.
3. Find the best feasible point $\mathbf{z}_{\min} = \arg\min\{f(\mathbf{z}_\iota)$ where $c_j(\mathbf{z}_\iota) \le 0\ \forall j = 1\ldots,m\}$ with lowest function value $f_{\min}$, and determine the worst feasible objective function value $f_{\max} = \max\{f(\mathbf{z}_\iota)$ where $c_j(\mathbf{z}_\iota) \le 0\ \forall j = 1\ldots,m\}$.
4. Compute $f_p(\mathbf{z}_\iota), \iota = 1,\ldots,n_0$, the adjusted objective function values according to (10).
5. Use the data $(\mathbf{z}_\iota, f_p(\mathbf{z}_\iota)), \iota = 1,\ldots,n_0$, to calculate the RBF model parameters by solving (7).
6. Iterate until the maximal number of allowed function evaluations has been reached:
   (a) Create four groups of candidate points by randomly (i) perturbing only continuous variable values of $\mathbf{z}_{\min}$, (ii) perturbing only discrete variable values of $\mathbf{z}_{\min}$, (iii) perturbing continuous and discrete variable values of $\mathbf{z}_{\min}$, and (iv) uniformly sampling points from Ω_b.
   (b) Calculate the scoring criteria for every candidate point.
   - Predict the objective function value of each candidate point $\boldsymbol{\chi}_j, j = 1,\ldots,t$, using the response surface, and compute $V_R(\boldsymbol{\chi}_j)$ in (13).
   - Determine the distance of each candidate point $\boldsymbol{\chi}_j, j = 1,\ldots,t$, to $\mathcal{Z}_n$, and compute $V_D(\boldsymbol{\chi}_j)$ in (14).
   - Compute the weighted score for every candidate point $W(\boldsymbol{\chi}_j)$ in (15).

   (c) Choose from each group the candidate point with the best score W.
   (d) Do the expensive function evaluations at these points (in parallel).
   (e) Update best feasible point found so far $\mathbf{z}_{\min}$. Update the worst feasible objective function value $f_{\max}$, and adjust the objective function values according to (10) or (12), depending on the stage of the algorithm.
   (f) Update the RBF model parameters by solving the system (7) using the penalty augmented objective function values from Step 6(e).
7. Return the best feasible solution found $\mathbf{z}_{\min}$.

4.5. Convergence of SO-MI

The algorithm SO-MI is convergent, specifically asymptotically complete [44]. It is assumed that f is a deterministic real-valued function on the compact set $\Omega \subseteq \Omega_b$ defined by the box constraints, and the linear and nonlinear constraints $c_j, j = 1,\ldots,m$, if they exist. It is assumed that f and $c_j, j = 1,\ldots,m$, are continuous when the integer variables are fixed. Denote $f^* = \inf_{\mathbf{z}\in\Omega} f(\mathbf{z}) > -\infty$ the feasible global minimum. Let $\mathbf{z}^*$ be a feasible global minimizer of f over Ω and suppose that f is continuous at $\mathbf{z}^*$.

Theorem 4.1. *The algorithm SO-MI is asymptotically complete, i.e. assuming an indefinitely long run-time and exact computations, a global minimum of the optimization problem (1)–(5) will be found with probability one.*

The proof of the theorem is given in the Online Supplement.

5. Numerical experiments

The SO-MI algorithm has been implemented and tested in Matlab 2010a. The performance of SO-MI is in the following analysis compared in numerical experiments to a branch and bound algorithm for solving nonlinear mixed-integer optimization problems, a genetic algorithm, and the mixed-integer option of the C++ implementation of NOMAD [12,17].

The branch and bound algorithm for nonlinear problems is based on the implementation by Kuipers [45] and applies the trust-region-reflective algorithm (Matlab optimization toolbox function fmincon) when solving the relaxed subproblems in the tree nodes. The maximum number of function evaluations to solve one subproblem has been kept at the default value $100k$. The branch and bound implementation uses a depth-first search with backtracking. The branching variable is chosen such that $|f(x_1,\ldots,x_{k_1},u_1,\ldots,u_{i_2},\ldots,u_{k_2}) - f(x_1,\ldots,x_{k_1},u_1,\ldots,[u_{i_2}],\ldots,u_{k_2})|$ is maximized over all $i_2 \in \{1,\ldots,k_2\}$, where $[u_{i_2}]$ is the value of the i_2th discrete variable rounded to its closest integer.

The branch and bound algorithm has been included in the comparison because it is a widely used algorithm for solving mixed-integer optimization problems. However, as argued before, the performance of branch and bound cannot be expected to be

[2] If objective and constraint function values are the output of separate black-box simulation models, it might be favorable (depending on the number of constraints and their corresponding computational demand) to use $m+1$ processors, and do objective and constraint evaluations for one point at a time in parallel rather than using one processor for each point.

good on multimodal problems due to the computation of the lower bounds in the tree nodes that would require a global optimization algorithm. The comparison with branch and bound is included to show that although the algorithm is suitable for solving mixed-integer problems with special characteristics such as convexity, it is not a feasible option for finding (near) optimal solutions to black-box problems within only very few function evaluations.

The genetic algorithm has a population of 20 individuals, 20 generations, and, following [46], uses real-coded chromosomes. The parents for generating the next generation's individuals are chosen based on their objective function value f_p. Crossover and mutation operations have been applied for generating the offspring. In the crossover operation two parents are chosen and the crossover point is selected randomly. The mutation operator randomly selects variables of the parent and adds a value $d\zeta$, where ζ is a random variable drawn from the normal distribution $\mathcal{N}(0,1)$, and $d=1$ initially and increases if the mutation does not result in new offspring, i.e. variable vectors that are not yet included in the population. The discrete variables are then rounded to the closest integer values, and variables exceeding any upper or lower bounds are replaced by the respective value of the bound that has been exceeded.

As suggested by the literature [47,48] and in order to help the genetic algorithm to perform well within a limited number of function evaluations, a dynamic adjustment of the crossover and mutation probabilities is applied. The probability of using crossover decreases as the number of generations increases, whereas the probability of using mutation increases with the generation number. Hence, a transition from a rather global to a more local search can be achieved as the algorithm advances. The linear and nonlinear constraints have been incorporated in the objective function with a penalty term. The branch and bound algorithm and the genetic algorithm were implemented and tested with Matlab 2010a.

NOMAD 3.5 has been obtained from [12] and can solve mixed-integer problems [17]. The C++ implementation has been used because it incorporates the variable neighborhood search (the setting VNS_SEARCH 0.75 as suggested in the user manual has been used) that enables the algorithm to escape from local optima. The constraints are treated with the progressive barrier approach (setting PB). Although the user manual states that NOMAD can be used with a surrogate model, the software does not include an implementation of a surrogate model. Therefore, NOMAD has not been used with surrogate models in the numerical experiments.

In the SO-MI algorithm, the penalty factor $c_p=100$ in (10) has been used. The factor guarantees that infeasible points will have worse function values than the worst feasible point. By subsequently replacing large function values with the median, numerical instabilities can be prevented. Furthermore, the numerical experiments showed that the strategy of using four candidate point groups with different ranges of perturbations as well as randomly sampled points as described in Section 4.3 is very successful. During the iterations improvements were found by points from all four groups.

The maximum number of allowed function evaluations for SO-MI, the genetic algorithm, and NOMAD has been set to 300 since the algorithm developed in this paper is intended for problems for which only a limited number of function evaluations can be done, and in many real life applications even 100 function evaluations are already computationally infeasible. Branch and bound was run until it stopped, but only the first 300 function evaluations have been considered in the comparison. Thirty trials have been made with every algorithm for every problem, and every algorithm was given the same feasible point for the same trial of a given test problem. This feasible point was either the starting point for the algorithm (NOMAD and branch and bound), or it was added to the initial experimental design/initial generation (SO-MI and genetic algorithm).

The mixed-integer global optimization solver *arbfMIP* contained in the commercial TOMLAB optimization environment [35] has not been included in the comparison for the following reasons. TOMLAB has been developed for unconstrained problems and problems that have computationally cheap constraints. If the constraints are computationally expensive, they have to be incorporated in the objective function with a penalty term. There are however no recommendations on how to adjust the penalty factor. Moreover, the objective and constraint function values must be computed in separate Matlab files, and thus there are difficulties applying the penalty method used in SO-MI as described in Section 3. A fair comparison between SO-MI and TOMLAB for constrained problems is therefore not possible.

Furthermore, *arbfMIP* has several parameters that need to be adjusted (for example, global search type, cycle length, global and local solver, etc.), which may have to be adjusted according to the test problem, and which is without further knowledge of the problem at hand difficult to do such that the comparison with the other algorithms would be fair.

The mixed-integer global optimization solver *arbfMIP* has been applied to a five- and a 30-dimensional unconstrained test problem with computationally cheap objective functions. The calculations for the five-dimensional problem have been interrupted after more than two hours computation time and the algorithm was not close to having done 300 computationally cheap function evaluations. For the 30-dimensional problem, the algorithm required more than 2 h for finding only 50 points for doing the computationally cheap function evaluations, i.e. the *arbfMIP* tends to become itself a computational burden (SO-MI needs less than a tenth of that time). Assuming an allowed maximum number of 300 function evaluations, where each evaluation would take about 2 min, the total time for solving the problem would almost double for the 30-dimensional problem. Thus, the TOMLAB optimization environment seems to be efficient only for problems where the function evaluation requires considerably more time. Also the solver *glcSolve* could be used, which is significantly faster than *arbfMIP*. For *glcSolve* it is however not possible for the user to define a starting point (the given feasible point). Thus, also for this solver a fair comparison of the algorithms would not be possible. For the reasons stated above, TOMLAB has not been included in the computational experiments.

6. Test problems

6.1. Generic test problems

Totally, 16 test problems (which are modifications of literature problems) have been used to examine the efficiency and solution quality of SO-MI compared to branch and bound, the genetic algorithm, and NOMAD. Four test problems used by Koziel and Michalewicz [49] (test problems 5–8, Table 1), that are originally continuous global optimization test problems, have been used and integrality constraints have been imposed on some of the continuous variables.

Four test problems from the Mixed-Integer Nonlinear Programming models library MINLPLib [50] (test problems 3, 9, 11, 12, Table 1), five problems that have only box constraints (test problems 2, 10, 13, 14, 15, Table 1), and three test problems from [51] have been used to compare the algorithms (see also [52–54], integer variables are binary variables in these cases, test problems 1, 4, 16, Table 1). The mathematical description of all test problems

Table 1
Summary of test problems; NLC—nonlinear constraints, LC—linear constraints, FEA—finite element analysis, k—problem dimension, k_2—dimension of integer variables, UM—unimodal, MM—multimodal; see Online Supplement for further information (a) http://www.aridolan.com/ga/gaa/MultiVarMin.html, (b) reliability–redundancy allocation problem.

Problem ID	k	Domain	k_2	Notes	Characteristics
1	11	$[0,1]^8 \times [0,0.997] \times [0,0.9985] \times [0,0.9988]$	4	[52]	3 NLC, 4 LC, MM
2	8	$[-10,10]^8$	4	Convex	No constraints, UM
3	5	$[0,10]^3 \times [0,1]^2$	2	[50]	2 NLC, 3 LC, UM
4	3	$[0,1] \times [0.2,1] \times [-2.22554,-1]$	1	[70]	1 NLC, 2 LC, MM
5	25	$[0,10]^{25}$	6	[49]	1 NLC, 1 LC, MM
6	5	$[78,102] \times [33,45] \times [27,45]^3$	2	[49]	6 NLC, UM/flat
7	2	$[13,100] \times [0,100]$	1	[49]	2 NLC, MM
8	7	$[-10,10]^7$	3	[49]	4 NLC, MM
9	5	$[0,10]^3 \times [0,1]^2$	3	[50], linear	3 LC, UM
10	5	$[-100,100]^5$	2	(a)	No constraints, MM
11	10	$[3,9]^{10}$	5	[50]	No constraints, UM
12	10	$[3,99]^{10}$	5	[50]	No constraints, UM
13	12	$[-1,3]^{12}$	5	[71]	No constraints, MM
14	12	$[-10,30]^{12}$	5	[71]	No constraints, MM
15	30	$[-1,3]^{30}$	10	[71]	No constraints, MM
16	11	$[0,1]^8 \times [0,10]^3$	4	[54]	4 NLC, 9 LC, MM
17	14	$[10,60]^{11} \times [2000,3200]^3$	11	2-dim. truss	FEA
18	31	$[1,10]^{24} \times [0,1000]^7$	24	3-dim. truss	FEA
19	10	$[1,10]^5 \times [0,0.999999]^5$	5	(b), bridge	3 NLC
20	8	$[1,10]^4 \times [0,0.999999]^4$	4	(b), overspeed	3 NLC
21	10	$[1,10]^5 \times [0,0.999999]^5$	5	(b), series-parallel	3 NLC

is for convenience given in the Online Supplement together with the best known solution. Although some of the problems have convex objective functions, the constraints determine the number of local optima. Many problems have therefore several local optima. Furthermore, some problems have flat regions, i.e. regions where several points of the variable domain have the same function value.

6.2. Structural design applications

Two applications from structural design have been examined. Mixed-integer optimization problems are often encountered in this application area, for example when designing truss structures.

The goal is in general to minimize structural costs such as the total weight of the structure while satisfying constraints such as limits on the maximal nodal displacements when loads are applied. A finite element analysis must be done to determine these nodal displacements, and depending on the number of elements involved and the type of finite element analysis required, the computation times may become a considerable burden [55–59].

The need for integer as well as continuous decision variables arises because technological requirements do not allow the production of truss members with arbitrary cross-sectional areas. Rather, catalogues are used from which commercially available member sizes may be chosen. Therefore, discrete variables enter the design problem. On the other hand, as the length of the truss element is variable (one can always weld truss members together), continuous variables are encountered. A more thorough description of the two examined design optimization problems is given in the Online Supplement.

6.3. Reliability–redundancy allocation problems

Three application problems arising from reliability engineering are examined. Reliability engineering is an important topic in fields such as system, mechanical, electronics or software engineering. Depending on the system under consideration different levels of reliability must be guaranteed. The consequences of the failure of a system's reliability may vary significantly between different applications (compare for example the crash of an airplane and the malfunctioning of a coffee machine). Although the reliability requirements for different systems are in general very different, the common goal is to maximize the total system reliability.

The reliability of a system is defined as the probability that a system or device will perform its intended function for a specified time period under given restrictions such as production costs, for example. The most commonly used system reliability measure is the mean time to failure (MTTF). The higher the MTTF is the higher the system's reliability.

The engineer has in general two options for increasing the reliability of a system. On the one hand, it is possible to increase the reliability of single components (continuous variables), and on the other hand redundancy can be provided at various stages of the system (discrete variables). Because the component cost often increases exponentially when the component reliability exceeds a certain limit, it might be cheaper to use components of lower reliability and to provide redundancy, i.e. to add more components of the same kind of lower reliability to the system. Although additional components also incur costs, the cost increase might be less compared to the costs caused by increasing the component reliability. A tradeoff between component reliability increments and component redundancy arises. This problem type is in the literature referred to as reliability–redundancy allocation (see for example [60]).

Since reliability is a probability, it is in practice difficult to test a system's reliability. A single test of the system is in general not representative, and performing multiple tests or tests of systems with a high MTTF may be too expensive. Thus, when adjusting reliability and redundancy parameters for maximizing the total system reliability, it is important to strictly limit the number of reliability tests to a minimum.

In the literature, different heuristic algorithms [61,62], as well as algorithms based on branch and bound [63] and evolutionary algorithms [64,65] have been developed for solving

reliability–redundancy allocation problems. However, these methods require in general many function evaluations which may become infeasible in practice.

Three reliability–redundancy allocation problems have in the following been examined, namely a series-parallel configuration, a bridge system, and an overspeed protection system (see [66–69]). Block diagrams for a simple bridge and series-parallel configuration, and the configuration specific data are given in the Online Supplement.

6.4. Overview of test problems

A summary of all test problems is shown in Table 1. The column k_2 denotes the number of discrete variables, and the column *Notes* gives information about the origin of the problem. The column *Characteristics* contains information about the problem constraints, where NLC stands for nonlinear constraints, and LC for linear constraints, and indicates whether the problem is unimodal (UM) or multimodal (MM). FEA means that a finite element analysis has to be done for evaluating the constraints.

7. Numerical results

In the following the genetic algorithm will be referred to as GA, and the branch and bound algorithm for nonlinear problems will be abbreviated by B&B. The test problems have been divided into four groups as follows. Table 2 shows the results for all test problems that have only box constraints. Table 4 summarizes the results of the algorithms for problems that have in addition to the box constraints linear and/or nonlinear constraints. Table 6 shows the numerical results for the structural design application problems, and the results of the reliability–redundancy allocation problems are given in Table 8 (note that these are maximization problems).

Each table shows the problem ID as defined in Table 1, and the function value achieved by every algorithm after 100, 200, and 300 function evaluations, respectively, averaged over 30 trials together with the corresponding standard errors of the mean (SEM) in columns 100 *eval.*, 200 *eval.*, 300 *eval.* The reported numbers are all for feasible points only. Note that if a test problem has constraints, then the constraints are evaluated if and only if the objective function is evaluated, i.e. every objective function evaluation is followed by the evaluation of all constraints, and constraints are not evaluated at a point without the objective function being evaluated. In many real-life application problems objective and constraint function values are the output of the same computationally expensive black-box simulation, and therefore this assumption has been made in the numerical experiments. The columns *dim.* and $|\Omega^D|$ in all tables give information about the problem dimension and the cardinality of the discrete variable domain. The last column indicates whether the problem is unimodal (UM) or multimodal (MM).

It has to be emphasized that the algorithms are compared with respect to the number of function evaluations needed to find improvements, and that the computation times of the algorithms are neglected because in applications where a function evaluation may take up to several hours or even days, the algorithms' computation times become insignificant. The objective function values reported in the tables are for feasible points only. It shall be noted at this point that the performance of NOMAD, GA, and B&B may be improved by combining these algorithms with surrogate models. However, publicly available codes are rather scarce, and as in the case of the C++ implementation of NOMAD, the developers ask the user to implement the surrogate model him/herself.

In order to better compare the solution quality of the algorithms for the four problem classes, the "score" rows of Tables 2, 4, 6 and 8 summarize how much each algorithm deviates from the best feasible solution averaged over 30 trials (numbers in %). The value is computed for each column (100 eval., 200 eval., and 300 eval.) and each algorithm as

$$\frac{1}{|\mathcal{N}|}\sum_{\pi\in\mathcal{N}}\left|\frac{f_a^n - f_{\text{best}}^n}{f_{\text{best}}^n}\right|\cdot 100, \tag{16}$$

where $\mathcal{N}$ is the set of problems in each category (box-constrained/constrained/structural/reliability-redundancy), π denotes a certain problem within the category, f_{best}^n is the best average feasible objective function value reached for a given problem after n function evaluations (n is 100, 200, or 300), and f_a^n is the average objective function value reached by algorithm a (a is SO-MI, GA, B&B, or NOMAD) after n evaluations for a given problem. The best algorithm for a given problem receives therefore the score 0, whereas all other algorithms with worse results obtain a positive number. Thus, by this definition, the smaller the reported score is, the better is the algorithm's performance because the lower is the deviation from the best solution.

In addition, Tables 3, 5, 7, and 9 contain information of hypothesis testing for differences in means between the algorithms after 100, 200, and 300 function evaluations.

7.1. Results for box-constrained problems

The results for the box-constrained problems in Table 2 show that SO-MI and NOMAD outperform B&B and GA on all problems of this group. Comparing the feasible function values averaged over all 30 trials shows that SO-MI is better than NOMAD for five out of seven problems after 100 function evaluations (column 3), indicating that SO-MI finds improvements faster. SO-MI is also superior after 200 and 300 function evaluations.

B&B did not find any improvements during 300 function evaluations for test problems 11, 12, 13, 14 and 15. GA found improvements for all problems, but could not compete with SO-MI or NOMAD. B&B has in general the worst performance, and is outperformed by *all* other algorithms for five out of seven problems including the unimodal test problems. This fact indicates that the optimization of the subproblems in the leaves of the branch and bound algorithm consumes too many function evaluations, and feasible solutions with respect to the integer constraints cannot be found efficiently.

Problems 13 and 14 are essentially the same problems, but problem 14 has a larger variable domain. The results show that for both problems the results found by SO-MI after 300 evaluations are about equal to the results found by NOMAD, whereas the results of NOMAD for problem 14 are for all evaluations significantly worse. B&B found for problem 13 only very slight improvements of the initially given feasible solution within 300 evaluations, and was not able to find any improvements when the variable domain was increased. Also the performance of GA is significantly worse for the problem with larger variable domain (problem 14) than for problem 13.

Similarly, problems 11 and 12 have the same structure, but problem 12 has a larger variable domain. The results show that SO-MI is able to find for both problems a near optimal feasible solution within fewer than 100 function evaluations, whereas NOMAD reaches competitive results after 300 evaluations. B&B did not find any improvements of the initially given solutions for both problems. GA performs slightly worse for problem 12 (the final solution is about 40% worse than the best feasible solution after 300 evaluations) than for problem 11 (the final solution

Table 2
Mean objective function values (mean) over 30 trials with 100, 200, and 300 function evaluations for box-constrained problems. Best result of all algorithms is marked by bold. Standard errors of means (SEM) are given for each algorithm in italic. All problems are minimization problems. MM—multimodal, UM—unimodal.

Problem	Algorithm	Statistic	100 eval.	200 eval.	300 eval.	Dim.	$\lvert\Omega^D\rvert$	MM/UM
10	SO-MI	Mean	**−386.33**	−420.91	−432.58	5	101^2	MM
		SEM	*16.74*	*14.87*	*15.17*			
	GA	Mean	−240.21	−300.64	−306.96			
		SEM	*18.78*	*18.54*	*18.04*			
	B&B	Mean	644.33	−120.76	−357.94			
		SEM	*107.72*	*95.97*	*19.42*			
	NOMAD	Mean	−380.20	**−460.05**	**−479.98**			
		SEM	*14.49*	*12.41*	*9.37*			
11	SO-MI	Mean	**−42.92**	**−42.99**	**−42.99**	10	7^5	UM
		SEM	*0.19*	*0.13*	*0.13*			
	GA	Mean	−17.00	−26.31	−33.73			
		SEM	*0.78*	*0.76*	*0.77*			
	B&B	Mean	4.23	4.23	4.23			
		SEM	*1.08*	*1.08*	*1.08*			
		Mean	−26.78	−32.99	−38.26			
	NOMAD	SEM	*1.03*	*1.01*	*0.87*			
12	SO-MI	Mean	**−9581.32**	**−9584.62**	**−9584.62**	10	97^5	UM
		SEM	*5.26*	*0.79*	*0.79*			
	GA	Mean	−4931.94	−5648.29	−5825.53			
		SEM	*170.55*	*168.37*	*172.75*			
	B&B	Mean	−1513.54	−1513.54	−1513.54			
		SEM	*131.72*	*131.72*	*131.72*			
	NOMAD	Mean	−5280.95	−7436.47	−9201.87			
		SEM	*184.13*	*222.49*	*75.23*			
13	SO-MI	Mean	−4.89	−8.48	**−9.63**	12	5^5	MM
		SEM	*0.33*	*0.30*	*0.23*			
	GA	Mean	2.69	−1.71	−3.74			
		SEM	*0.60*	*0.34*	*0.28*			
	B&B	Mean	23.88	23.88	23.88			
		SEM	*2.05*	*2.05*	*2.05*			
	NOMAD	Mean	**−7.51**	**−8.66**	−9.27			
		SEM	*0.43*	*0.19*	*0.31*			
14	SO-MI	Mean	27.95	**−5.78**	**−8.27**	12	41^5	MM
		SEM	*2.85*	*0.27*	*0.34*			
	GA	Mean	816.53	510.26	376.38			
		SEM	*40.68*	*26.82*	*22.18*			
	B&B	Mean	2802.79	2802.79	2802.79			
		SEM	*180.28*	*180.28*	*180.28*			
	NOMAD	Mean	**23.10**	19.88	13.23			
		SEM	*1.94*	*1.13*	*2.31*			
15	SO-MI	Mean	**−6.59**	−10.29	−11.80	30	5^{10}	MM
		SEM	*0.37*	*0.36*	*0.47*			
	GA	Mean	25.48	13.83	10.73			
		SEM	*1.46*	*1.09*	*1.07*			
	B&B	Mean	62.03	62.03	62.03			
		SEM	*3.64*	*3.64*	*3.64*			
	NOMAD	Mean	−6.39	**−19.27**	**−19.52**			
		SEM	*2.08*	*0.90*	*0.88*			
Score in %	SO-MI		**9**	**9**	**8**			
	GA		700	1550	827			
	B&B		2325	8282	5830			
	NOMAD		15	81	46			
2 convex	SO-MI	Mean	**0.00**	**0.00**	**0.00**	8	21^4	UM
		SEM	*0.00*	*0.00*	*0.00*			
	GA	Mean	274.17	153.30	82.79			
		SEM	*21.13*	*11.26*	*8.28*			
	B&B	Mean	1471.05	1274.99	4.29			
		SEM	*132.65*	*145.11*	*1.10*			
	NOMAD	Mean	117.70	7.36	0.14			
		SEM	*1.23*	*1.68*	*0.05*			

is about 22% worse than the best feasible solution after 300 evaluations), indicating that the performance of GA decreases when the variable domain gets larger. These four problems (11, 12, 13, and 14) show that the performance of SO-MI is less affected by the size of the variable domain than the other algorithms.

Table 3
Hypothesis testing for differences in means (μ) after 100, 200, and 300 function evaluations for box-constrained problems. (**) denotes significance at the $\alpha = 1$ percentage point for $H_0 : \mu_{SO-MI} = \mu_{\mathcal{A}}, \mathcal{A} \in \{GA, B\&B, NOMAD\}$, and $H_1 : \mu_{SO-MI} < \mu_{\mathcal{A}}$; ($\diamond$) denotes significance at the $\alpha = 5$ percentage point, and ($\diamond\diamond$) denotes significance at the $\alpha = 1$ percentage point for $H_0 : \mu_{SO-MI} = \mu_{NOMAD}$, and $H_1 : \mu_{SO-MI} > \mu_{NOMAD}$.

Problem	Algorithm $\mathcal{A}$	100 evals.	200 evals.	300 evals.
10	GA	(**)	(**)	(**)
	B&B	(**)	(**)	(**)
	NOMAD		($\diamond$)	($\diamond\diamond$)
11	GA	(**)	(**)	(**)
	B&B	(**)	(**)	(**)
	NOMAD	(**)	(**)	(**)
12	GA	(**)	(**)	(**)
	B&B	(**)	(**)	(**)
	NOMAD	(**)	(**)	(**)
13	GA	(**)	(**)	(**)
	B&B	(**)	(**)	(**)
	NOMAD	($\diamond\diamond$)		
14	GA	(**)	(**)	(**)
	B&B	(**)	(**)	(**)
	NOMAD		(**)	(**)
15	GA	(**)	(**)	(**)
	B&B	(**)	(**)	(**)
	NOMAD		($\diamond\diamond$)	($\diamond\diamond$)
2	GA	(**)	(**)	(**)
	B&B	(**)	(**)	(**)
	NOMAD	(**)	(**)	(**)

The scores for each algorithm in the table are computed without taking the convex problem 2 into consideration because SO-MI was developed for non-convex and multimodal problems, and was not expected to work so well on a convex problem. The convex problem was included only to examine how well SO-MI might do on this problem class. The scores in the table show that SO-MI performs better than all other algorithms and has the smallest deviations from the best feasible solution found. Although SO-MI does not continuously perform best on all problems, the cases where it is outperformed by NOMAD show that the differences between the results found by NOMAD and SO-MI are much smaller than the differences for the problems where SO-MI outperforms NOMAD. Compared to GA and B&B, SO-MI and NOMAD perform significantly better.

The standard errors of the means (SEM) given for each algorithm in Table 2 are for SO-MI in general lowest (except for test problem 10). The hypothesis testing for differences in means between the different algorithms in Table 3 shows that except for problems 10, 13, and 15 the average feasible objective function values of SO-MI are significantly lower than those of GA, NOMAD, and B&B at the significance level $\alpha = 0.01$. NOMAD reaches significantly lower values than SO-MI only for test problems 10 and 15 after 200 and more function evaluations, and for problem 13 for up to 100 evaluations. The SEM values reported for B&B for test problems 11, 12, 13, 14, and 15 correspond to the value computed based on the initially given feasible points since B&B was not able to find any improvements for these test problems.

Fig. 2 illustrates the development of the objective function value for feasible points averaged over 30 trials versus the number of function evaluations for all four compared algorithms for test problem 12. Since the problem was about minimization, the best algorithm is the one that achieves the fastest reduction of the objective function value. The figure shows that SO-MI improves the objective function value within the lowest number of function evaluations as compared to GA, NOMAD and B&B.

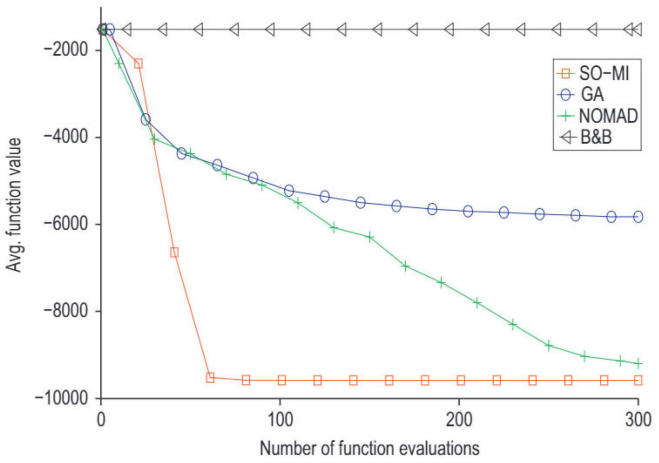

Fig. 2. Objective function value for feasible points averaged over 30 trials vs. number of function evaluations; test problem 12, box-constrained minimization problem.

NOMAD eventually finds a feasible solution that is very similar (less than 4% difference) to the one found by SO-MI, but comparing the graphs of SO-MI and NOMAD after 100 and 200 function evaluations, for example, shows that SO-MI performs significantly better. GA also finds improvements, but is not able to perform nearly as well as SO-MI. B&B did not find any improvements within 300 function evaluations for this test problem.

7.2. Results for constrained problems

The numerical results for the constrained test problems are shown in Table 4. The results show that SO-MI performs best for six out of nine test problems after 100 evaluations, and for five out of nine test problems after 200 and 300 evaluations, respectively. NOMAD on the other hand had the best performance only for one of the nine test problems after 200 function evaluations, respectively, and for two of nine test problems after 300 function evaluations. These results indicate that SO-MI performs in general better than NOMAD also for the constrained problems, and that SO-MI is able to find improvements within fewer function evaluations as compared to NOMAD.

B&B found the best results for test problems three, six, and nine, where test problem six is in particular a convex problem, and test problems three and nine have a linear objective function. Test problems three and nine are essentially the same problem, but the nonlinear constraints have been left out in test problem nine. Moreover, for test problems three and nine the globally optimal point of the continuous relaxation is the same as when integer constraints are imposed (the global optimum of the continuous problem has all variable values zero), which explains the superior performance of B&B on these problems.

For test problem eight the optimization of the relaxed sub-problems in the tree nodes of B&B did not converge and the algorithm stalled. Thus, the reported result is the average of the function values of the initially given feasible points for the 30 trials. Moreover, B&B was not able to find improvements/escape from a local optimum for the remaining five test problems (test problems 1, 4, 5, 7 and 16) within 300 evaluations, and has for these problems the worst performance among all compared algorithms.

GA was able to improve the initially given solution for all but one test problem, and performs in general better than B&B, but significantly worse than SO-MI and NOMAD. Although GA has been developed for solving multimodal optimization problems, the number of function evaluations required is very large due to

Table 4
Mean objective function values (mean) over 30 trials with 100, 200, and 300 function evaluations for problems with box-constraints and additional linear and/or nonlinear constraints. Best result of all algorithms is marked by bold. Standard errors of means (SEM) are given for each algorithm in italic. All problems are minimization problems. B&B * means, that the subsolver for the relaxed problems in the tree nodes did not converge and the algorithm stalled, therefore also SEM is not available. MM—multimodal, UM—unimodal.

Problem	Algorithm	Statistic	100 eval.	200 eval.	300 eval.	Dim.	$\lvert\Omega^D\rvert$	MM/UM
1	SO-MI	Mean	**−0.50**	**−0.66**	**−0.72**	11	2^4	MM
		SEM	*0.03*	*0.02*	*0.02*			
	GA	Mean	0.26	−0.31	−0.33			
		SEM	*0.03*	*0.03*	*0.03*			
	B&B	Mean	−0.09	−0.09	−0.09			
		SEM	*0.02*	*0.02*	*0.02*			
	NOMAD	Mean	−0.47	−0.53	−0.64			
		SEM	*0.02*	*0.02*	*0.02*			
4	SO-MI	Mean	**2.99**	2.95	2.93	3	2	MM
		SEM	*0.02*	*0.02*	*0.01*			
	GA	Mean	3.16	3.16	3.16			
		SEM	*0.02*	*0.02*	*0.02*			
	B&B	Mean	3.18	3.18	3.18			
		SEM	*0.02*	*0.02*	*0.02*			
	NOMAD	Mean	3.03	**2.94**	**2.90**			
		SEM	*0.02*	*0.01*	*0.01*			
5	SO-MI	Mean	**−0.18**	**−0.25**	**−0.32**	25	11^6	MM
		SEM	*0.00*	*0.01*	*0.01*			
	GA	Mean	−0.15	−0.15	−0.16			
		SEM	*0.00*	*0.00*	*0.00*			
	B&B	Mean	−0.09	−0.09	−0.09			
		SEM	*0.00*	*0.00*	*0.00*			
	NOMAD	Mean	−0.17	−0.19	−0.19			
		SEM	*0.01*	*0.01*	*0.01*			
7	SO-MI	Mean	**−4156.44**	**−4182.99**	**−4186.56**	2	14	MM
		SEM	*14.39*	*9.37*	*8.12*			
	GA	Mean	−3266.52	−3380.55	−3616.21			
		SEM	*88.78*	*99.30*	*99.71*			
	B&B	Mean	−3194.01	−3194.01	−3194.01			
		SEM	*93.41*	*93.41*	*93.41*			
	NOMAD	Mean	−3825.44	−3845.79	−3847.36			
		SEM	*48.91*	*47.20*	*47.62*			
8	SO-MI	Mean	**1057.39**	**847.65**	**761.448**	7	21^3	MM
		SEM	*17.81*	*15.68*	*8.63*			
	GA	Mean	1 095 045.15	874 665.90	629 728.43			
		SEM	*446 436.77*	*409 890.30*	*368 763.38*			

	B&B *	Mean	1 436 981.97	1 436 981.97	1 436 981.97			
		SEM	*NA*	*NA*	*NA*			
	NOMAD	Mean	3830.17	1124.20	835.22			
		SEM	*916.38*	*66.73*	*18.79*			
16	SO-MI	Mean	**8.05**	**6.60**	**6.20**	11	2^4	MM/flat
		SEM	*0.22*	*0.10*	*0.06*			
	GA	Mean	10.75	10.60	10.10			
		SEM	*0.33*	*0.29*	*0.31*			
	B&B	Mean	10.93	10.93	10.93			
		SEM	*0.35*	*0.35*	*0.35*			
	NOMAD	Mean	9.09	8.65	7.76			
		SEM	*0.29*	*0.30*	*0.23*			
Score in %	SO-MI		**0**	**0**	**1**			
	GA		17 265	15 351	12 579			
	B&B		22 666	25 226	28 700			
	NOMAD		50	15	11			
3	SO-MI	Mean	0.44	0.09	0.04	5	11^2	UM
		SEM	*0.06*	*0.02*	*0.01*			
	GA	Mean	3.69	2.36	1.03			
		SEM	*0.34*	*0.34*	*0.24*			
	B&B	Mean	**0.00**	**0.00**	**0.00**			
		SEM	*0.00*	*0.00*	*0.00*			
	NOMAD	Mean	0.62	0.21	0.01			
		SEM	*0.11*	*0.07*	*0.01*			
6	SO-MI	Mean	−29 767.98	−29 983.69	−30 078.49	5	975	UM/flat
		SEM	*67.99*	*53.92*	*51.23*			
	GA	Mean	−28 993.04	−29 285.90	−29 461.46			
		SEM	*100.28*	*99.67*	*108.62*			
	B&B	Mean	**−30 665.54**	**−30 665.54**	**−30 665.54**			
		SEM	*0.00*	*0.00*	*0.00*			
	NOMAD	Mean	−29 407.09	−29 998.12	−30 215.53			
		SEM	*129.71*	*66.70*	*45.52*			
9 linear	SO-MI	Mean	0.67	0.35	0.17	5	11^2	UM
		SEM	*0.14*	*0.15*	*0.07*			
	GA	Mean	4.33	1.42	0.67			
		SEM	*0.35*	*0.26*	*0.22*			
	B&B	Mean	**0.00**	**0.00**	**0.00**			
		SEM	*0.00*	*0.00*	*0.00*			
	NOMAD	Mean	0.41	0.16	**0.00**			
		SEM	*0.09*	*0.06*	*0.00*			

Table 5
Hypothesis testing for differences in means (μ) after 100, 200, and 300 function evaluations for box-constrained problems with additional linear and/or nonlinear constraints. ($*$) denotes significance at the $\alpha=5$ percentage point, and ($**$) denotes significance at the $\alpha=1$ percentage point for $H_0:\mu_{SO-MI}=\mu_{\mathcal{A}}, \mathcal{A}\in\{GA,B\&B,NOMAD\}$, and $H_1:\mu_{SO-MI}<\mu_{\mathcal{A}}$; ($\diamond$) denotes significance at the $\alpha=5$ percentage point, and ($\diamond\diamond$) denotes significance at the $\alpha=1$ percentage point for $H_0:\mu_{SO-MI}=\mu_{NOMAD}$, and $H_1:\mu_{SO-MI}>\mu_{NOMAD}$; ($\oplus$) denotes significance at the $\alpha=5$ percentage point, and ($\oplus\oplus$) denotes significance at the $\alpha=1$ percentage point for $H_0:\mu_{SO-MI}=\mu_{B\&B}$, and $H_1:\mu_{SO-MI}>\mu_{B\&B}$.

Problem	Algorithm $\mathcal{A}$	100 evals.	200 evals.	300 evals.
1	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD		($**$)	($**$)
4	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD			($\diamond$)
5	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD		($**$)	($**$)
7	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD	($**$)	($**$)	($**$)
8	GA	($**$)	($*$)	($*$)
	B&B	NA	NA	NA
	NOMAD	($**$)	($**$)	($**$)
16	GA	($**$)	($**$)	($**$)
	B&B	($**$)	($**$)	($**$)
	NOMAD	($**$)	($**$)	($**$)
3	GA	($**$)	($**$)	($**$)
	B&B	($\oplus\oplus$)	($\oplus\oplus$)	($\oplus\oplus$)
	NOMAD			
6	GA	($**$)	($**$)	($**$)
	B&B	($\oplus\oplus$)	($\oplus\oplus$)	($\oplus\oplus$)
	NOMAD	($**$)		($\diamond$)
9	GA	($**$)	($**$)	($*$)
	B&B	($\oplus\oplus$)	($\oplus$)	($\oplus\oplus$)
	NOMAD			($\diamond\diamond$)

the number of individuals in the single generations and the number of generations.

Also for this problem class a score has been computed that measures how much each algorithm deviates from the best feasible average result found after 100, 200, and 300 evaluations. Test problems 3, 6 and 9 have not been included in the calculation because problem 6 is convex, and problems 3 and 9 have linear objective functions and the continuous relaxations lead directly to an integer feasible solution. SO-MI has been developed for multi-modal problems and therefore these problems are left out when computing the scores. The scores show that SO-MI performs also for this problem class better than any of the other algorithms. NOMAD performs better than GA and B&B, and its performance improves as the number of function evaluations increases.

The hypothesis testing for differences in means in Table 5 shows again that SO-MI attains significantly better results (at the level $\alpha=0.01$) than all other algorithms for most problems. For reasons explained above, B&B performs significantly better on the special problems 3, 6, and 9.

Fig. 3 illustrates the development of the function value for feasible points averaged over 30 trials versus the number of function evaluations for all algorithms for test problem five. The figure shows that SO-MI is able to find significantly better feasible solutions than the other three algorithms. It can be seen that all B&B trials got stuck in the same point, and the algorithm did not find any improvements within 300 function evaluations. NOMAD and GA are able to find slight improvements of the initially supplied feasible solutions, but are not able to find any more improvements after about 150 function evaluations.

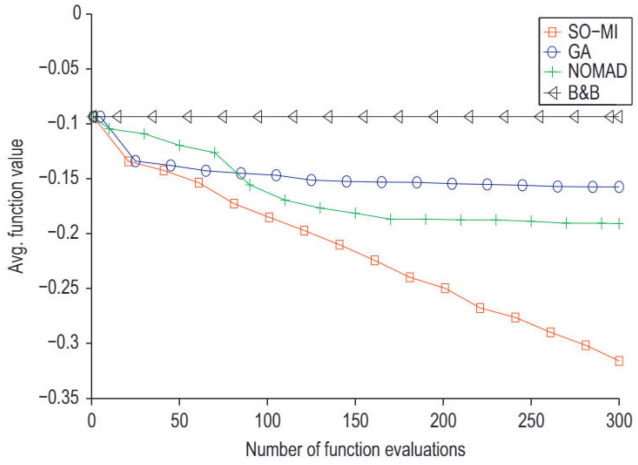

Fig. 3. Objective function value for feasible points averaged over 30 trials vs. number of function evaluations; test problem 5, constrained minimization problem.

7.3. Results for structural design problems

The results for the structural design optimization applications are shown in Table 6. For both problems SO-MI outperforms all other algorithms and reaches significantly better results within the same number of function evaluations. B&B was not able to find any improvements for the three-dimensional truss dome problem. For the two-dimensional truss the optimization of the relaxed subproblems in the tree nodes of B&B did not converge within the allowed $100k$ function evaluations, and therefore the average of the function values of the 30 different initially provided feasible points is given in the table, and the standard error of the mean is therefore not available. B&B had the worst performance for both structural design problems.

GA was able to improve the initially supplied feasible solutions and outperformed NOMAD on the three-dimensional truss dome application (the 31-dimensional problem), but does worse than NOMAD on the smaller problem 17. The scores at the end of the table reflect again that SO-MI performs better than all other algorithms on this problems class. Averaging over both structural design problems shows that GA outperforms NOMAD.

For this problem class the standard errors of the means are for SO-MI at almost all stages of the algorithm lower than for the other algorithms (except for GA after 100 evaluations for test problem 18). Also the results of the statistical tests in Table 7 show that the means of the feasible objective function values attained by SO-MI are significantly lower than the results found by the other algorithms.

The objective function value averaged over all 30 trials versus the number of function evaluations for the three-dimensional truss dome application problem is illustrated for all four algorithms in Fig. 4. The figure shows that SO-MI finds significantly better solutions than all other algorithms. Fig. 5 illustrates the best design obtained for the 2D truss problem. Thicker lines illustrate truss members with larger cross-sectional areas. The arrows indicate the height of nodes 2, 4, and 6. The maximal nodal displacement occurs at node 7 in vertical direction.

7.4. Results for reliability–redundancy problems

Table 8 summarizes the results of the four compared algorithms for the reliability–redundancy allocation application problems.

Table 6
Mean objective function values (mean) over 30 trials with 100, 200, and 300 function evaluations for structural design optimization problems. Best result of all algorithms is marked by bold. Standard errors of means (SEM) are given for each algorithm in italic. B&B* means, that the subsolver for the relaxed problems in the tree nodes did not converge and the algorithm stalled, and therefore also SEM is not available. Both problems are minimization problems.

Problem	Algorithm	Statistic	100 eval.	200 eval.	300 eval.	Dim.	$\lvert\Omega^D\rvert$
17 2D truss	SO-MI	Mean	**102624.50**	**98099.88**	**95902.51**	14	51^{11}
		SEM	*807.45*	*707.95*	*614.30*		
	GA	Mean	112 975.58	112 924.52	112 816.52		
		SEM	*889.47*	*900.90*	*933.85*		
	B&B*	Mean	113 161.21	113 161.21	113 161.21		
		SEM	NA	NA	NA		
	NOMAD	Mean	103 354.72	101 433.69	98 484.86		
		SEM	*815.42*	*858.34*	*932.62*		
18 3D dome	SO-MI	Mean	**186 281.11**	**83 525.43**	**73 010.86**	31	10^{24}
		SEM	*5716.76*	*3007.10*	*1640.95*		
	GA	Mean	277 476.80	239 829.09	212 346.21		
		SEM	*3942.84*	*4613.54*	*3666.28*		
	B&B	Mean	355 160.38	355 160.38	355 160.38		
		SEM	*6786.15*	*6786.15*	*6786.15*		
	NOMAD	Mean	303 995.43	268 712.00	244 911.47		
		SEM	*9193.94*	*7153.09*	*8595.33*		
Score in %	SO-MI		**0**	**0**	**0**		
	GA		30	97	101		
	B&B		50	164	196		
	NOMAD		32	108	115		

Table 7
Hypothesis testing for differences in means (μ) after 100, 200, and 300 function evaluations for structural design optimization problems. (*) denotes significance at the $\alpha=5$ percentage point; (**) denotes significance at the $\alpha=1$ percentage point; $H_0: \mu_{SO-MI} = \mu_{\mathcal{A}}, \mathcal{A} \in \{GA, B\&B, NOMAD\}$, and $H_1: \mu_{SO-MI} < \mu_{\mathcal{A}}$.

Problem	Algorithm $\mathcal{A}$	100 evals.	200 evals.	300 evals.
17	GA	(**)	(**)	(**)
	B&B	NA	NA	NA
	NOMAD		(**)	(*)
18	GA	(**)	(**)	(**)
	B&B	(**)	(**)	(**)
	NOMAD	(**)	(**)	(**)

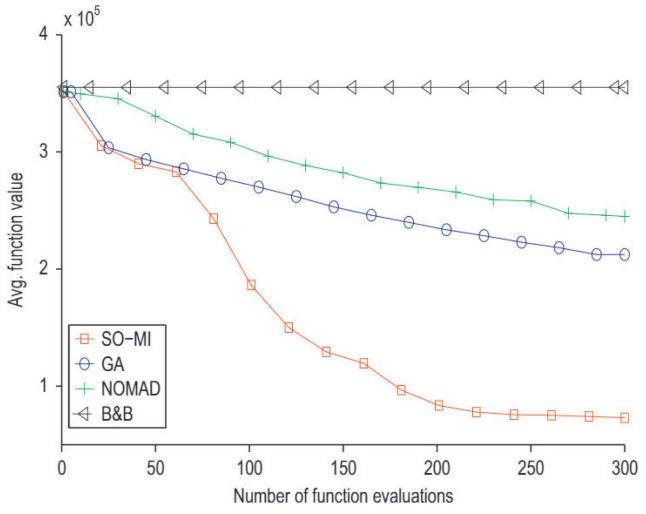

Fig. 4. Objective function value for feasible points averaged over 30 trials vs. number of function evaluations; test problem 18, truss dome design optimization, minimization problem.

The goal is here to maximize the reliability of the system, and therefore large numbers are better. Note that in this application area the digits after the decimal point are important and therefore

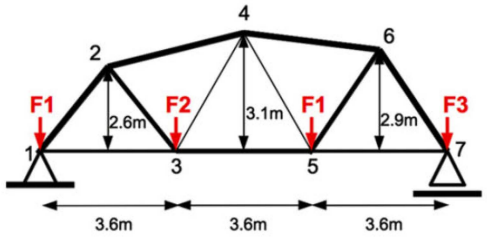

Fig. 5. Best design of the 2D truss problem. Thicker lines symbolize truss members with larger cross-sectional areas. The largest deflection is in vertical direction at node 7. See Online Supplement for the problem description and data.

four decimals are reported. The results show that SO-MI performs best for all three problems for over 200 function evaluations, indicating that SO-MI is successful in finding improvements more efficiently than all other algorithms.

NOMAD achieves only marginally better results for problem 20 after 300 function evaluations. B&B was not able to find any feasible improvements of the initially supplied feasible points, and for the series-parallel system (problem 21) the optimization of the relaxed problems in the tree nodes did not converge. The reported numbers for B&B for this problem are the average of the objective function values of the initially supplied feasible points. GA found slight improvements for all three test problems, but was in general not as good as SO-MI or NOMAD. This fact is also reflected by the scores at the end of the table. SO-MI performs best on this problem class, and the performance of NOMAD improves as the number of function evaluations increases.

The standard errors of the means are for all algorithms very low, but again SO-MI and NOMAD have the lowest values. The statistical test results for differences of means are reported in Table 9, and the results show that the mean of SO-MI is for all problems at almost all stages of the algorithm (100, 200, and 300 evaluations) significantly higher (maximization problem) than for the other algorithms (at levels $\alpha=0.01$ and $\alpha=0.05$, respectively).

Fig. 6 shows the objective function value for feasible points averaged over 30 trials versus the number of function evaluations for all algorithms for the series-parallel system (test problem 21). Note that for the maximization problem the algorithm that

Table 8
Mean objective function values (mean) over 30 trials with 100, 200, and 300 function evaluations for reliability–redundancy allocation problems (maximization problems). Best result of all algorithms is marked by bold. Standard errors of means (SEM) are given for each algorithm in italic. B&B* means, that the subsolver for the relaxed problems in the tree nodes did not converge and the algorithm stalled, and therefore also SEM is not available.

Problem	Algorithm	Statistic	100 eval.	200 eval.	300 eval.	Dim.	$\lvert\Omega^D\rvert$
19 bridge system	SO-MI	Mean	**0.9843**	**0.9974**	**0.9982**	10	5^{10}
		SEM	*0.00*	*0.00*	*0.00*		
	GA	Mean	0.8211	0.8224	0.8369		
		SEM	*0.03*	*0.03*	*0.02*		
	B&B	Mean	0.7573	0.7573	0.7573		
		SEM	*0.03*	*0.03*	*0.03*		
	NOMAD	Mean	0.9640	0.9849	0.9950		
		SEM	*0.01*	*0.00*	*0.00*		
20 over-speed	SO-MI	Mean	**0.9939**	**0.9988**	0.9991	8	4^{10}
		SEM	*0.00*	*0.00*	*0.00*		
	GA	Mean	0.9588	0.9637	0.9646		
		SEM	*0.01*	*0.00*	*0.00*		
	B&B	Mean	0.8060	0.8060	0.8060		
		SEM	*0.03*	*0.03*	*0.03*		
	NOMAD	Mean	0.9884	0.9971	**0.9992**		
		SEM	*0.00*	*0.00*	*0.00*		
21 series parallel	SO-MI	Mean	**0.9618**	**0.9962**	**0.9977**	10	5^{10}
		SEM	*0.01*	*0.00*	*0.00*		
	GA	Mean	0.6452	0.6452	0.6539		
		SEM	*0.04*	*0.04*	*0.04*		
	B&B*	Mean	0.4731	0.4731	0.4731		
		SEM	*NA*	*NA*	*NA*		
	NOMAD	Mean	0.8132	0.9388	0.9833		
		SEM	*0.04*	*0.01*	*0.01*		
Score in %	SO-MI		**0**	**0**	**0**		
	GA		17	18	18		
	B&B*		31	32	32		
	NOMAD		6	2	0		

Table 9
Hypothesis testing for differences in means (μ) after 100, 200, and 300 function evaluations for reliability–redundancy allocation problems. (*) denotes significance at the $\alpha=5$ percentage point; (**) denotes significance at the $\alpha=1$ percentage point; $H_0: \mu_{\text{SO-MI}} = \mu_{\mathcal{A}}, \mathcal{A} \in \{GA, B\&B, NOMAD\}$, and $H_1: \mu_{\text{SO-MI}} > \mu_{\mathcal{A}}$ (maximization problem).

Problem	Algorithm $\mathcal{A}$	100 evals.	200 evals.	300 evals.
19	GA	(**)	(**)	(**)
	B&B	(**)	(**)	(**)
	NOMAD	(*)	(**)	(*)
20	GA	(**)	(**)	(**)
	B&B	(**)	(**)	(**)
	NOMAD		(**)	
21	GA	(**)	(**)	(**)
	B&B	NA	NA	NA
	NOMAD	(**)	(**)	(*)

increases the average feasible objective function value fastest is best. The figure shows that NOMAD finds immediately improvements of the given feasible point because it uses only one point from which the mesh adaptive direct search starts. SO-MI on the other hand generates at first an initial experimental design and starts the "systematic" search for improvements after all points in the starting design have been evaluated. This is reflected by the initially constant average feasible objective function value of SO-MI (the value is constant until evaluation $2(k+1)=22$ because no other feasible points were in the initial experimental design).

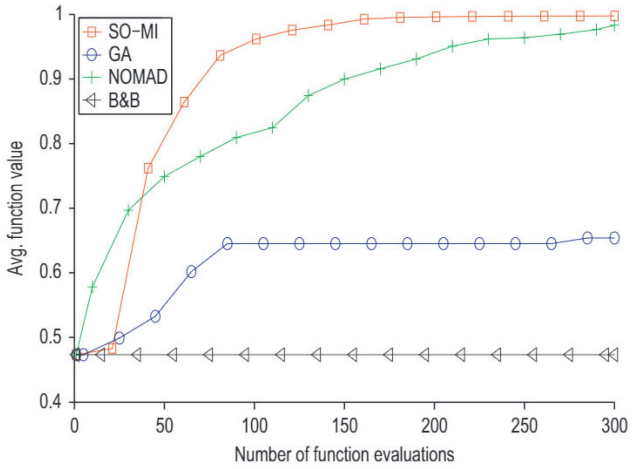

Fig. 6. Objective function value for feasible points averaged over 30 trials vs. number of function evaluations; test problem 21, reliability–redundancy allocation for series-parallel configuration, maximization problem.

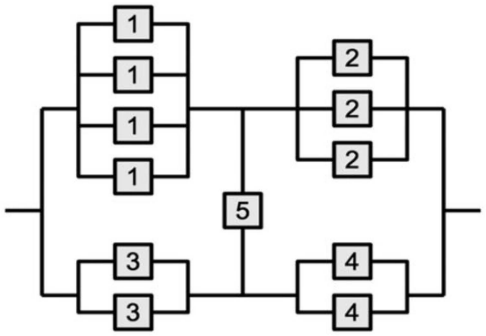

Fig. 7. Best reliability–redundancy allocation for the bridge system. Redundancy has been added at components 1, 2, 3, and 4. See Online Supplement for the problem description and data.

After the points in the initial experimental design have been evaluated, SO-MI immediately finds significantly better feasible results than all other algorithms. Fig. 7 shows the best reliability–redundancy allocation for the bridge system (test problem 19). As can be seen, redundancy has been added at various stages.

8. Conclusions

In this paper a surrogate model approach for finding (near) optimal solutions to black-box mixed-integer global optimization problems within a very restricted number of function evaluations has been presented. The algorithm, SO-MI, iteratively evaluates the computationally expensive simulation model at four chosen points of the variable domain in parallel and updates a radial basis function surrogate model. Four perturbation methods are used to diversify the selection of candidates for the next sample point in every iteration, and based on scoring criteria the best point of each group is chosen. In general, more than four points could be chosen to make use of more processors. The algorithm SO-MI converges to the global optimum almost surely.

SO-MI has been shown to perform very well in comparison to a branch and bound algorithm for nonlinear problems [45], a genetic algorithm, and the NOMAD algorithm for multimodal mixed-integer problems [17] on 16 test problems from the literature containing unconstrained, constrained, unimodal, and

multimodal problems, two application problems arising from structural optimization, and three application problems from optimal reliability design.

The numerical results show that SO-MI is able to find significantly better solutions than the other algorithms for 10 out of 16 literature test problems and all application problems when the number of function evaluations is low (200 to 300 evaluations). On the remaining six problems, SO-MI performed only slightly worse than the best algorithm which is also reflected in a measure that is computed based on the deviation of the results obtained by each algorithm from the best solution. This measure shows that SO-MI performs in general better than all other algorithms for each problem group (unconstrained, constrained, structures problems, and reliability problems). Statistical tests on the differences of means for each problem also support this conclusion.

The genetic algorithm and branch and bound have for almost all problems the worst performance, whereas NOMAD is able to find competitive solutions for some problems. Note however that the results of these algorithms may be further improved by incorporating a surrogate model. The test problems included also a convex test problem (problem 2) for which SO-MI performed best, and two special unimodal problems where the global optimum of the continuous relaxation is at an integer point (problems 3 and 9). As may be expected, branch and bound performed very well on problems 3 and 9, indicating that the algorithm may be preferred if the specific problem structure is known to be mathematically tractable. However, for black-box problems, branch and bound requires in general too many function evaluations when computing lower bounds for the objective function value in the tree nodes. Also, for multimodal problems these lower bounds are not necessarily valid and pruning decisions may eventually be wrong.

Although developed for multimodal problems, the genetic algorithm does not perform well in the computational experiments. Genetic algorithms require in general many function evaluations (=population size $\times$ number of generations) to find good solutions. Thus, when only a very low number of evaluations can be allowed due to restrictions on the computational expense, either the number of individuals in each generation has to be reduced, which restricts the diversity in the population, or the number of generations has to be reduced, which reduces the evolutionary aspect of the algorithm.

The NOMAD algorithm is also a derivative-free method, and the C++ implementation incorporates a mixed-integer version, and the variable neighborhood search that allows the algorithm to escape from local optima. There are local convergence results for NOMAD, but in the numerical experiments NOMAD is in general outperformed by SO-MI, especially on the structural optimization problems where the difficulty was in the black-box constraints rather than the objective function. However, compared to branch and bound and the genetic algorithm, NOMAD performed very well.

In conclusion, the introduced algorithm SO-MI extends the research area of using surrogate models for solving mixed-integer optimization problems. The used candidate point approach works better than solving an auxiliary optimization subproblem for determining the next sample site (which in general requires a mixed-integer global optimization subsolver for the considered problem class). The computational results indicate that SO-MI is a promising algorithm and performs significantly better than commonly used algorithms for mixed-integer problems when dealing with multimodal black-box functions. In addition it can be shown that the SO-MI algorithm converges to the global optimum almost surely for multimodal mixed-integer problems with black-box objective functions. None of the other algorithms have such a proof for this type of problem. It is expected that the performance of SO-MI for black-box constrained problems can be further improved by replacing the penalty approach for example with an approach that uses a response surface for each constraint to eliminate candidate points from consideration. A comparison of various ways for handling constraints exceeds however the scope of this paper and is left for future research.

Acknowledgments

The first author thanks the Finnish Academy of Science and Letters, Vilho, Yrjö and Kalle Väisälä Foundation for the financial support. Prof. Shoemaker has been supported by NSF CISE: AF 1116298 and CPA-CSA-T 0811729. Finally, we would like to thank the anonymous reviewers for their helpful suggestions.

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at http://dx.doi.org.10.1016/j.cor.2012.08.022.

References

[1] Costa L, Oliveira P. Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems. Computers and Chemical Engineering 2001;25:257–66.
[2] Gantovnik V, Gürdal Z, Watson L, Anderson-Cook C. A genetic algorithm for mixed integer nonlinear programming problems using separate constraint approximations. AIAA Journal 2005;43:1844–9.
[3] Haupt R. Antenna design with a mixed integer genetic algorithm. IEEE Transactions on Antennas and Propagation 2007;55:577–82.
[4] Schlüter M, Gerdts M, Rückmann J-J. MIDACO: new global optimization software for MINLPs, 2010. ⟨http://www.midaco-solver.com/about.html⟩.
[5] Viswanathan J, Grossmann I. A combined penalty function and outer-approximation method for MINLP optimization. Computers and Chemical Engineering 1990;14:769–82.
[6] Bonami P, Biegler L, Conn A, Cornuéjols G, Grossmann I, Laird C, et al. An algorithmic framework for convex mixed integer nonlinear programs. Discrete Optimization 2008;5:186–204.
[7] Nowak I. Relaxation and decomposition methods for mixed integer nonlinear programming. Birkhäuser; 2005.
[8] Nowak I, Vigerske S. LaGO—a (heuristic) branch and cut algorithm for nonconvex MINLPs. Central European Journal of Operations Research 2008;16:127–38.
[9] Smith E, Pantelides C. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimization of nonconvex MINLPs. Computers and Chemical Engineering 1999;23:457–78.
[10] Holland J. Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press; 1975.
[11] Abramson M, Audet C, Chrissis J, Walston J. Mesh adaptive direct search algorithms for mixed variable optimization. Optimization Letters 2009;3:35–47.
[12] Abramson M, Audet C, Couture G, Dennis Jr J, Le Digabel S. The NOMAD project. Software available at: ⟨http://www.gerad.ca/nomad⟩, 2011.
[13] Abramson M, Audet C, Dennis Jr J. Filter pattern search algorithms for mixed variable constrained optimization problems. SIAM Journal on Optimization 2004;11:573–94.
[14] Abramson M, Audet C. Convergence of mesh adaptive direct search to second-order stationary points. SIAM Journal on Optimization 2006;17:606–19.
[15] Hansen P, Mladenović N. Variable neighborhood search: principles and applications. European Journal of Operational Research 2001;130:449–67.
[16] Mladenović N, Hansen P. Variable neighborhood search. Computers and Operations Research 1997;24:1097–100.
[17] Audet C, Béchard V, Le Digabel S. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. Journal of Global Optimization 2008;41:299–318.
[18] Audet C, Dennis Jr J. Pattern search algorithms for mixed variable programming. SIAM Journal on Optimization 2000;11:573–94.
[19] Liuzzi G, Lucidi S, Rinaldi F. Derivative-free methods for bound constrained mixed-integer optimization, Computational Optimization and Applications, 2011, http://dx.doi.org/10.1007/s10589-011-9405-3.
[20] Booker A, Dennis Jr J, Frank P, Serafini D, Torczon V, Trosset M. A rigorous framework for optimization of expensive functions by surrogates. Structural Multidisciplinary Optimization 1999;17:1–13.

[21] Forrester A, Sóbester A, Keane A. Engineering design via surrogate modelling—a practical guide. Wiley; 2008.
[22] Gutmann H. A radial basis function method for global optimization. Journal of Global Optimization 2001;19:201–27.
[23] Jones D. A taxonomy of global optimization methods based on response surfaces. Journal of Global Optimization 2001;21:345–83.
[24] Jones D, Schonlau M, Welch W. Efficient global optimization of expensive black-box functions. Journal of Global Optimization 1998;13:455–92.
[25] Müller J, Piché R. Mixture surrogate models based on Dempster–Shafer theory for global optimization problems. Journal of Global Optimization 2011;51:79–104.
[26] Regis R. Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. Computers and Operations Research 2011;38:837–53.
[27] Regis R, Shoemaker C. A stochastic radial basis function method for the global optimization of expensive functions. INFORMS Journal on Computing 2007;19:497–509.
[28] Wild S, Regis R, Shoemaker C. ORBIT: optimization by radial basis function interpolation in trust-regions. SIAM Journal on Scientific Computing 2007;30:3197–219.
[29] Glaz B, Friedmann P, Liu L. Surrogate based optimization of helicopter rotor blades for vibration reduction in forward flight. Structural and Multidisciplinary Optimization 2008;35:341–63.
[30] Jouhaud J, Sagaut P, Montagnac M, Laurenceau J. A surrogate-model based multidisciplinary shape optimization method with application to a 2D subsonic airfoil. Computers and Fluids 2007;36:520–9.
[31] Lam X, Kim Y, Hoang A, Park C. Coupled aerostructural design optimization using the kriging model and integrated multiobjective optimization algorithm. Journal of Optimization Theory and Applications 2009;142:533–56.
[32] Li C, Wang F, Chang Y, Liu Y. A modified global optimization method based on surrogate model and its application in packing profile optimization of injection molding process. The International Journal of Advanced Manufacturing Technology 2010;48:505–11.
[33] Davis E, Ierapetritou M. Kriging based method for the solution of mixed-integer nonlinear programs containing black-box functions. Journal of Global Optimization 2009;43:191–205.
[34] Hemker T. Derivative free surrogate optimization for mixed-integer nonlinear black box problems in engineering, PhD thesis, TU Darmstadt; 2008.
[35] Holmström K, Quttineh N, Edvall M. An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer constrained global optimization. Journal of Global Optimization 2008;41:447–64.
[36] Rashid K, Ambani S, Cetinkaya E. An adaptive multiquadric radial basis function method for expensive black-box mixed-integer nonlinear constrained optimization. Engineering Optimization http://dx.doi.org/10.1080/0305215X.2012.665450, 2012.
[37] Friedman J. Multivariate adaptive regression splines. The Annals of Statistics 1991;19:1–141.
[38] Myers R, Montgomery D. Response surface methodology, process and product optimization using designed experiments. Wiley-Interscience Publication; 1995.
[39] Currin C, Mitchell T, Morris M, Ylvisaker D. A Bayesian approach to the design and analysis of computer experiments, Technical Report, Oak Ridge National Laboratory, Oak Ridge, TN; 1988.
[40] Duchon J. Constructive theory of functions of several variables. Berlin: Springer-Verlag; 1977.
[41] Regis R, Shoemaker C. Constrained global optimization of expensive black-box functions using radial basis functions. Journal of Global Optimization 2005;31:153–71.
[42] Regis R, Shoemaker C. Improved strategies for radial basis function methods for global optimization. Journal of Global Optimization 2007;37:113–35.
[43] Powell M. 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; 1992, p. 105–210.
[44] Neumeier A. Complete search in constrained global optimization. Acta Numerica 2004;13:271–369.
[45] Kuipers E. 〈http://www.mathworks.com/matlabcentral/fileexchange/95〉, Last accessed May 31, 2011. (2000).
[46] Herrera F, Lozano M, Verdegay J. Tackling real-coded genetic algorithms: operators and tools for behavioural analysis. Artificial Intelligence Review 1998;12:265–319.
[47] Srinivas M, Patnaik L. Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics 1994;24:565–667.
[48] Zhang J, Chung H, Lo W. Clustering-based adaptive crossover and mutation probabilities for genetic algorithms. IEEE Transactions on Evolutionary Computation 2007;11:326–35.
[49] Koziel S, Michalewicz Z. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. Evolutionary Computation 1999;7:19–44.
[50] Bussieck M, Stolbjerg Drud A, Meeraus A. MINLPLib—a collection of test models for mixed-integer nonlinear programming. INFORMS Journal on Computing 2003;15:114–9.
[51] Adjiman C, Androulakis I, Floudas C. Global optimization of mixed-integer nonlinear problems. AIChE Journal 2000;46:1769–97.
[52] Berman O, Ashrafi N. Optimization models for reliability of modular software systems. IEEE Transactions on Software Engineering 1993;19:1119–23.
[53] Floudas C, Pardalos O, Adjiman C, Esposito W, Gumus Z, Harding S, et al. Handbook of test problems in local and global optimization. Dordrecht: Kluwer Academic Publishers; 1999.
[54] Yuan X, Zhang S, Piboleau L, Domenech S. Une méthode d'optimisation non linéaire en variables mixtes pour la conception de procédés. R.A.I.R.O. Operation Research 1988;22:331–46.
[55] Dede T, Bekiroğlu S, Ayvaz Y. Weight minimization of trusses with genetic algorithm. Applied Soft Computing 2011;11:2565–75.
[56] Jensen H, Sepulveda J. Structural optimization of uncertain dynamical systems considering mixed-design variables. Probabilistic Engineering Mechanics 2011;26:269–80.
[57] Rajan S, Nguyen D. Design optimization of discrete structural systems using MPI-enabled genetic algorithm. Structural and Multidisciplinary Optimization 2004;27:1–9.
[58] Schittkowski K, Zillober C, Zotemantel R. Numerical comparison of nonlinear programming algorithms for structural optimization. Structural Optimization 1994;7:1–19.
[59] Stocki R, Kolanek K, Jendo S, Kleiber M. Study on discrete optimization techniques in reliability-based optimization of truss structures. Computers and Structures 2001;79:2235–47.
[60] Kuo W, Prasad V, Tillman F, Hwang C-L. Optimal reliability design: fundamentals and applications. Cambridge University Press; 2001.
[61] Gopal K, Aggarwal K, Gupta J. A method for solving reliability optimization problem. IEEE Transactions on Reliability 1980;R-29:36–8.
[62] Tillman F, Hwang C, Kuo W. Determining component reliability and redundancy for optimum system reliability. IEEE Transactions on Reliability 1977;R-26:162–5.
[63] Kuo W, Lin H, Xu Z, Zhang W. Reliability optimization with Lagrange multiplier and branch-and-bound technique. IEEE Transactions on Reliability 1987;R-36:624–30.
[64] Coit D, Smith A. Reliability optimization of series-parallel systems using a genetic algorithm. IEEE Transactions on Reliability 1996;45:254–66.
[65] Hsieh Y, Chen T, Bricker D. Genetic Algorithms for reliability design problems, Technical Report, Department of Industrial Engineering, University of Iowa; 1997.
[66] Chen T-C. IAs based approach for reliability redundancy allocation problems. Applied Mathematics and Computation 2006;182:1556–67.
[67] Dhingra A. Optimal apportionment of reliability and redundancy in series systems under multiple objectives. IEEE Transactions on Reliability 1992;41:576–82.
[68] Hikita M, Nakagawa Y, Nakashima K, Narihisa H. Reliability optimization of systems by a surrogate-constraints algorithm. IEEE Transactions on Reliability 1992;R-41:473–80.
[69] Yokota T, Gen M, Li Y-X. Genetic algorithm for nonlinear mixed-integer programming problems and its application. Computers and Industrial Engineering 1996;30:905–17.
[70] Floudas C. Nonlinear and mixed-integer optimization: fundamentals and applications Oxford: Oxford University Press; 1995.
[71] Törn A, Zilinskas A. Global optimization. Lecture notes in computer science, vol. 350. Berlin: Springer-Verlag; 1989.