Robust Evolutionary Optimization Based on Coevolution

Steffen Limmer and Tobias Rodemann

Honda Research Institute Europe GmbH
63073 Offenbach am Main, Germany
{steffen.limmer,tobias.rodemann}@honda-ri.de

Abstract. A way to deal with uncertainties in the fitness function of an optimization problem is robust optimization, which optimizes the expected value of the fitness. In the context of evolutionary optimization, it is a common practice to compute the expected value of the fitness approximately with the help of Monte-Carlo simulation. This approach requires a lot of evaluations of the fitness function in order to evaluate an individual and thus it can be very compute-intensive.
In the present paper, we propose a coevolution-based approach for the robust optimization of problems with a fitness function basically depending on discrete random variables, which conditionally depend on the decision variables. Experiments on three benchmark functions show that the approach yields a good trade-off between the number of required fitness function evaluations and the quality of the results.

Keywords: robust optimization, coevolution, particle swarm optimization, evolutionary optimization

1 Introduction

For many real-world optimization tasks, the exact fitness of a vector $\boldsymbol{x}$ of decision variables is not known at the time of optimization, because the fitness function $f(\boldsymbol{x}, \boldsymbol{\alpha})$ does not only depend on the decision variables, but additionally on certain parameters $\boldsymbol{\alpha}$, whose values are not exactly known.

This problem arises, for example, often in the domain of design optimization, where design parameters of a certain product are optimized with respect to a given objective function f. Due to a limited precision of the manufacturing process, often the design parameters can be realized only to a certain degree of accuracy. In this case, the fitness of given design parameters $\boldsymbol{x}$ is actually not $f(\boldsymbol{x})$, but $f(\boldsymbol{x}+\boldsymbol{\alpha})$ with random disturbances $\boldsymbol{\alpha}$ underlying a certain probability distribution.

Another example of an optimization, where the exact fitness is often not known, is the optimization of a scheduling or control plan, like the plan for the operation of certain components of a microgrid. Here, the quality of a given plan $\boldsymbol{x}$ often depends on future environmental or operational conditions $\boldsymbol{\alpha}$, like

The final publication is available at Springer via
http://dx.doi.org/10.1007/978-3-319-77538-8_54

the future energy demand in a microgrid, which cannot be predicted with 100% accuracy.

A common way to deal with this problem is *robust optimization* [1, 2], which searches for a solution $\boldsymbol{x}*$ that is as robust as possible regarding the uncertainties in the parameters $\boldsymbol{\alpha}$ by optimizing the expected fitness

$$\mathbb{E}(f(\boldsymbol{x}, \boldsymbol{\alpha})) = \int_{-\infty}^{\infty} f(\boldsymbol{x}, \boldsymbol{\alpha}) \cdot p(\boldsymbol{\alpha})\, d\boldsymbol{\alpha}, \tag{1}$$

where $p(\boldsymbol{\alpha})$ is the probability distribution of $\boldsymbol{\alpha}$. A popular approach for robust optimization with evolutionary algorithms (EAs) is to approximate the expected fitness with the help of Monte-Carlo simulation: In order to evaluate a vector $\boldsymbol{x}$ of decision variables, κ samples $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\kappa$ are drawn from $p(\boldsymbol{\alpha})$ and the mean of the corresponding fitness values is used as fitness value for $\boldsymbol{x}$:

$$F(\boldsymbol{x}) := \frac{1}{\kappa} \sum_{i=1}^{\kappa} f(\boldsymbol{x}, \boldsymbol{\alpha}_i). \tag{2}$$

There are many examples of robust evolutionary optimization using this approach [3–8]. Its advantages are that it is easy to use, that it can be applied for arbitrary probability distributions of the parameters $\boldsymbol{\alpha}$ and that it does not require any information about properties of the function f, like its derivatives. But a disadvantage is that it requires a lot of evaluations of f, what makes it very compute-intensive.

Different approaches for accelerating the Monte-Carlo simulation can be found in the literature. For example, Loughlin and Ranjithan [9] propose to use Latin hypercube sampling (LHS) [10] instead of random sampling in order to reduce the number of required samples. Branke [11] suggests reducing the number of samples by using more samples for the evaluation of seemingly good individuals than for the evaluation of seemingly bad ones. Additionally, he suggests estimating the expected fitness of an individual by using information about the fitness of neighboring individuals. Aizawa and Wa [12] propose to use only a small number of samples for evaluations in the initial generations and to gradually increase the number of samples during the optimization. Like for conventional evolutionary optimization, the use of surrogate models is another option to speed up robust evolutionary optimization. Lee and Park [13] construct a kriging based surrogate model for f and use this model instead of f for the approximation of the expected fitness according to Equation (2). Paenke *et al.* [14] describe the construction of a surrogate model for F. Thus, this model can be used as a surrogate for the complete Monte-Carlo simulation.

In the present paper, we propose an approach based on coevolution for the robust optimization of problems with a fitness function that is basically a function in discrete random variables, which conditionally depend on the decision variables. The proposed approach might not be as accurate as Monte-Carlo simulation according Equation (2), but experiments show that it yields a good trade-off between the number of required fitness function evaluations and the

quality of the optimization results. Furthermore, the approach can be used for the fast computation of a good initial population for robust optimization with full Monte-Carlo simulation.

The rest of the paper is organized as follows: In the next section, the problem is described more in detail. Section 3 outlines the proposed approach for solving the problem as well as the conventional approach and a further approach that is based on an approximation of the expected fitness, which is only very rough. These three approaches are evaluated in numerical experiments, which are described and discussed in Section 4. Finally, Section 5 summarizes the work and its findings and provides an outlook on future work.

2 Problem Description

Before the assumed problem is described in general terms, a real-world example of a problem of the assumed type is given: The robust optimization of prices for products or services offered to N customers $c_1, \ldots, c_N$. Here, a decision variable x_i is a vector $(p_i^1, \ldots, p_i^K)$ of K prices for K products offered to customer c_i. Each customer can select exactly one of the offered products/prices or can alternatively decline all offers. The choice of a customer c_i is reflected by a random variable Y_i. Although the choices of the customers can be controlled to a certain degree by the selection of the prices, the exact choices are usually not known before the customers make their decisions. The costs associated with the delivery of the chosen products or services are computed over a function $g(\boldsymbol{Y})$. This might require the optimal scheduling of resources used for the provisioning of the products/services (e.g. with the help of linear programming), which is a compute-intensive task. A function $h(\boldsymbol{Y}, \boldsymbol{x})$ reflects the amount of money, the customers pay for the chosen products/services given the prices $\boldsymbol{x}$. The goal is the optimization of the prices with respect to the maximization of the profit $f(\boldsymbol{x}, \boldsymbol{Y}) = h(\boldsymbol{Y}, \boldsymbol{x}) - g(\boldsymbol{Y})$ robust to the uncertainties in the choices of the customers.

In the following, we assume the general problem of optimizing a vector $\boldsymbol{x} = (x_1, \ldots, x_N)$ of N decision variables. Each decision variable x_i is associated with a discrete random variable Y_i, which can take a small number K of values $y_i^1, \ldots, y_i^K$. Further, it is assumed that Y_i conditionally depends on x_i and that the probability distribution $p(Y_i|x_i)$ is known for each $i = 1, \ldots, N$. The fitness function is of the form

$$f(\boldsymbol{x}, \boldsymbol{Y}) = m(g(\boldsymbol{Y}), h(\boldsymbol{Y}, \boldsymbol{x})) \tag{3}$$

with $\boldsymbol{Y} = (Y_1, \ldots, Y_N)$. It is assumed that the function g is non-separable and compute-intensive, that m is a linear function and that h and m are computationally inexpensive. Thus, the fitness is a linear combination of a function that only depends on the random variables and a function that can also depend on the decision variables. Note that a special case of (3) is a fitness function that only depends on g and thus only on the random variables:

$$f(\boldsymbol{x}, \boldsymbol{Y}) = g(\boldsymbol{Y}). \tag{4}$$

The goal is to find a solution $\boldsymbol{x}*$ that minimizes/maximizes f robust to the uncertainties in the random variables $\boldsymbol{Y}$. Since g is assumed to be compute-intensive, only a small budget of evaluations of g is allowed.

3 Approaches for Solving the Problem

As outlined in Section 1, a problem like described in the previous section is commonly solved by minimizing[1] the expected fitness

$$F(\boldsymbol{x}) = \mathbb{E}(f(\boldsymbol{x}, \boldsymbol{Y})|\boldsymbol{x}). \tag{5}$$

According to Equation (3) and under consideration that m is linear, the expected fitness can be computed as

$$F(\boldsymbol{x}) = m(\mathbb{E}(g(\boldsymbol{Y})|\boldsymbol{x}), \mathbb{E}(h(\boldsymbol{Y}, \boldsymbol{x})|\boldsymbol{x})). \tag{6}$$

The expected value of g, given the decision variables $\boldsymbol{x}$, can be computed as

$$\mathbb{E}(g(\boldsymbol{Y})|\boldsymbol{x}) = \sum_{(i_1,\dots,i_N)\in\{1,\dots,K\}^N} g(y_1^{i_1}, \dots, y_N^{i_N}) \cdot P(\boldsymbol{Y} = (y_1^{i_1}, \dots, y_N^{i_N})|\boldsymbol{x}), \tag{7}$$

where $P(\boldsymbol{Y} = (y_1^{i_1}, \dots, y_N^{i_N})|\boldsymbol{x})$ is the probability for $\boldsymbol{Y} = (y_1^{i_1}, \dots, y_N^{i_N})$, given the decision variables $\boldsymbol{x}$. The computation of the expected value of g for a given $\boldsymbol{x}$ according (7) requires K^N evaluations of g. Since g is assumed to be compute-intensive, this is very time consuming and the runtime grows exponentially in the number N of decision variables.

In the case that g is a separable function, the runtime can be significantly reduced. Like described in Section 2, we assume g to be non-separable. But for the sake of completeness, the following subsection addresses the special case of a separable g.

3.1 The Special Case of a Separable Function g

If g is an additively separable function, i.e. $g(Y_1, \dots, Y_N)$ can be computed as $g_1(Y_1) + \cdots + g_N(Y_N)$ with certain functions $g_1, \dots, g_N$, the expected value of g for a given $\boldsymbol{x}$ can be computed as the sum of the expected values of the functions $g_1, \dots, g_N$:

$$\mathbb{E}(g(\boldsymbol{Y})|\boldsymbol{x}) = \sum_{i_1=1}^{K} g_1(y_1^{i_1}) P(Y_1 = y_1^{i_1}|\boldsymbol{x}) + \cdots + \sum_{i_N=1}^{K} g_N(y_N^{i_N}) P(Y_N = y_N^{i_N}|\boldsymbol{x}). \tag{8}$$

Thus, the computation of the expected value requires only $N \cdot K$ evaluations of g. For multiplicatively separable functions g, $\mathbb{E}(g(\boldsymbol{Y})|\boldsymbol{x})$ can be computed

[1] In the rest of the paper, we assume that the optimization problem at hand is a minimization problem.

analogously as a product of N sums. Since g does not depend on the decision variables, the $N \cdot K$ function values of g in Equation (8) can be computed offline before the actual optimization and can be reused in each evaluation of a vector $\boldsymbol{x}$ of decision variables. Thus, only $N \cdot K$ evaluations of g are required for the total optimization.

In the context of optimization, there exists the following definition of separability [15]: A function $f\colon S^N \to \mathbb{R}$ is separable, if and only if for all $k \in \{1, \ldots, N\}$ the following implication holds:

$$\begin{aligned} & f(x_1, \ldots, x_k, \ldots, x_N) < f(x_1, \ldots, x'_k, \ldots, x_N) \\ \rightarrow\ & f(z_1, \ldots, x_k, \ldots, z_N) < f(z_1, \ldots, x'_k, \ldots, z_N), \forall z_i \in S, 1 \leq i \leq N, i \neq k. \end{aligned} \tag{9}$$

Note that additively separable functions represent a special case of this type of separable functions, but a multiplicatively separable function does not necessarily fulfill implication (9). A function that is separable according to the previous definition, can be optimized by optimizing each decision variable individually. If the fitness function depends only on the function g (see Equation (4)) and g is separable, it is not necessary to compute the expected value of g in order to minimize the expected fitness. Instead, from (9) it follows that each decision variable x_i, $i = 1, \ldots, N$, can be optimized separately while assuming that the other decision variables x_j, $j \neq i$ equal an arbitrary constant, like 1, resulting in the following fitness function for the optimization of an individual decision variable x_i:

$$\begin{aligned} F'(x_i) &= \mathbb{E}(g(1, \ldots, 1, Y_i, 1, \ldots, 1) | x_i) \\ &= \sum_{j=1}^{K} g(1, \ldots, 1, y_i^j, 1, \ldots, 1) \cdot P(Y_i = y_i^j | x_i). \end{aligned} \tag{10}$$

Again, the K function values of g required in (10) can be computed offline and thus, for the optimization of the N decision variables, only $N \cdot K$ evaluations of g are required.

3.2 The Conventional Approach

In general, the analytical computation of the expected value of g requires K^N evaluations of g according Equation (7). Typically, the computation is accelerated by computing an approximation of the expected value over Monte-Carlo simulation as shown in Algorithm 1. In the following, this approach is denoted as the *conventional approach*.

With a sufficiently large number κ of samples, this approach will yield a good approximation of the true expected value. Using n times more samples, reduces the variance of the results of the Monte-Carlo simulation by a factor of $\sqrt{n}$. But a fitness evaluation of an individual requires a lot of evaluations of g if κ is chosen large.

Algorithm 1 Computation of the fitness of decision variables $\boldsymbol{x}$ over the conventional approach.

$fit \leftarrow 0$
for $s = 1, \ldots, \kappa$ **do**
  $\boldsymbol{Y_s} \leftarrow$ sample from distribution $p(\boldsymbol{Y}|\boldsymbol{x})$
  $fit \leftarrow fit + m(g(\boldsymbol{Y_s}), h(\boldsymbol{Y_s}, \boldsymbol{x}))$
end for
return $\frac{fit}{\kappa}$

3.3 The Lazy Approach

An approach that requires only one evaluation of g for the computation of the fitness of an individual is to approximate the expected value $\mathbb{E}(g(\boldsymbol{Y})|\boldsymbol{x})$ of the function in the random variables as the function value $g(\mathbb{E}(\boldsymbol{Y}|\boldsymbol{x}))$ in the expected value of the random variables.[2] In this case, the Monte-Carlo sampling is done only on the random variables and not on their function values, as shown in Algorithm 2. We denote this approach in the following as *lazy approach*. The expected value $\mathbb{E}(h(\boldsymbol{Y}, \boldsymbol{x})|\boldsymbol{x})$ of the function h might be computed analogously to the conventional approach, since h is assumed to be not compute-intensive.

Algorithm 2 Approximation of $\mathbb{E}(g(\boldsymbol{Y})|\boldsymbol{x})$ in the lazy approach.

$\boldsymbol{Y_{exp}} \leftarrow \boldsymbol{0}$
for $s = 1, \ldots, \kappa$ **do**
  $\boldsymbol{Y_s} \leftarrow$ sample from distribution $p(\boldsymbol{Y}|\boldsymbol{x})$
  $\boldsymbol{Y_{exp}} \leftarrow \boldsymbol{Y_{exp}} + \boldsymbol{Y_s}$
end for
return $g(\frac{\boldsymbol{Y_{exp}}}{\kappa})$

For most functions g, $\mathbb{E}(g(\boldsymbol{Y})|\boldsymbol{x}) \neq g(\mathbb{E}(\boldsymbol{Y}|\boldsymbol{x}))$ holds. Thus, generally the lazy approach can provide only a very rough approximation of the expected value of g. This may guide the optimization in the wrong direction, leading to a suboptimal solution of low robustness.

3.4 The Proposed Approach Based on Coevolution

As discussed in the previous two subsections, the conventional approach usually requires a high number of evaluations of g in order to find a good solution, while the lazy approach is likely to converge to a solution of low robustness. We propose an approach that combines the conventional and the lazy approach with the objective of achieving a reasonable trade-off between compute-intensity and quality

[2] Since $\boldsymbol{Y}$ is assumed to be discrete, g might be undefined for $\mathbb{E}(\boldsymbol{Y}|\boldsymbol{x})$. In this case, the most probable value of $\boldsymbol{Y}$ given $\boldsymbol{x}$ can be used instead of the expected value.

of the optimization results. Unlike the conventional and the lazy approach, the proposed approach does not only affect the way, individuals are evaluated, but also the control flow of the optimization. The approach employs coevolution analogous to the cooperative coevolutionary genetic algorithm proposed by Potter and De Jong [16]. Hence, we call it in the following *coevolution-based approach*.

The approach decomposes the N decision variables $\boldsymbol{x} = (x_1, \ldots, x_N)$ in $\frac{N}{G}$ groups $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{\frac{N}{G}}$ of G decision variables, each. This yields $\frac{N}{G}$ corresponding groups $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_{\frac{N}{G}}$ of random variables. Each decision variable group is optimized one after another, what is repeated for C cycles like illustrated in Figure 1a.

In the optimization of a group $\boldsymbol{x}_i$, the currently best values $\boldsymbol{x}_j^*$, $j \neq i$ for the other groups are used for the evaluation of a candidate solution. After the optimization, the currently best value $\boldsymbol{x}_i^*$ for the i-th group is replaced by the optimization result.

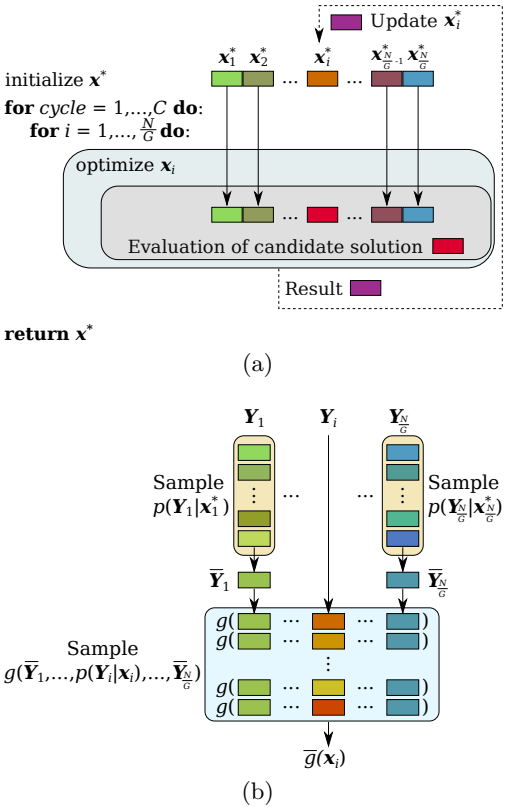

Fig. 1. Working principle of the coevolution-based approach: (a) Coevolution of multiple groups of decision variables. (b) Approximation of the expected value $\mathbb{E}(g(\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_{\frac{N}{G}})|\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_{\frac{N}{G}}^*)$ for the evaluation of a group $\boldsymbol{x}_i$ of decision variables.

In the evaluation of a group $\boldsymbol{x}_i$ of decision variables, the expected fitness according Equation (6) is approximated for $\boldsymbol{x} = (\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_{\frac{N}{G}}^*)$. The approximation of $\mathbb{E}(h(\boldsymbol{Y}, \boldsymbol{x})|\boldsymbol{x})$ is done over the conventional approach. The expected value of g given $\boldsymbol{x}$ is approximated like illustrated in Figure 1b. The random variables $\boldsymbol{Y}_j$ associated with all groups $\boldsymbol{x}_j$, $j \neq i$ are sampled from $p(\boldsymbol{Y}_j, \boldsymbol{x}_j^*)$ in order to approximate the expected values $\mathbb{E}(\boldsymbol{Y}_j|\boldsymbol{x}_j^*)$ of the random variables and these approximations are used together with samples of $\boldsymbol{Y}_i$ given $\boldsymbol{x}_i$ for the approximation of the expected value of g. Thus, group $\boldsymbol{x}_i$ is evaluated with the following fitness function:

$$F_i(\boldsymbol{x_i}) = m(\mathbb{E}(g(\mathbb{E}(\boldsymbol{Y}_1|\boldsymbol{x}_1^*), \ldots, \boldsymbol{Y}_i, \ldots, \mathbb{E}(\boldsymbol{Y}_{\frac{N}{G}}|\boldsymbol{x}_{\frac{N}{G}}^*))|\boldsymbol{x}_i), \mathbb{E}(h(\boldsymbol{Y}, \boldsymbol{x})|\boldsymbol{x})), \quad (11)$$

where the expected values are approximated over Monte Carlo simulation.

Since the currently best variables $\boldsymbol{x}_j^*$, $j \neq i$ are fixed during an optimization of $\boldsymbol{x}_i$, the approximations $\overline{\boldsymbol{Y}}_j$ of the expected values $\mathbb{E}(\boldsymbol{Y}_j|\boldsymbol{x}_j^*)$ can be precomputed before the optimization and can be used for the evaluations of different candidate solutions. This allows to precompute $g(\overline{\boldsymbol{Y}}_1, \ldots, \boldsymbol{Y}_i, \ldots, \overline{\boldsymbol{Y}}_{\frac{N}{G}})$ for all possible values of $\boldsymbol{Y}_i$ according Algorithm 3 and to use the precomputed values in the Monte-Carlo simulation for the fitness evaluation of a candidate solution. This requires K^G evaluations of g per optimization of a group, independent of the number of actual fitness evaluations during the optimization. The total coevolutionary optimization according to Figure 1a requires $C \cdot \frac{N}{G} \cdot K^G$ evaluations of g. Thus, with a group size G of only 1 and only one cycle in the optimization, $N \cdot K$ evaluations of g are required.

Although the fitness (11) is only a rough approximation of the expected fitness (6), the approximation should be in general better than that used in the lazy approach and compared to the conventional approach, less evaluations of g are required as long as C and G are not chosen too high.

Algorithm 3 Precomputation of $\overline{\boldsymbol{Y}}_j = \mathbb{E}(\boldsymbol{Y}_j|\boldsymbol{x}_j^*)$ and of $g(\overline{\boldsymbol{Y}}_1, \ldots, \boldsymbol{Y}_i, \ldots, \overline{\boldsymbol{Y}}_{\frac{N}{G}})$ for the optimization of a group $\boldsymbol{x}_i$ of decision variables with associated random variables $\boldsymbol{Y}_i = (Y_{i_1}, \ldots, Y_{i_G})$.

$g\_vals = \emptyset$
$\overline{\boldsymbol{Y}}_j \leftarrow 0$ for $j = 1, \ldots, \frac{N}{G}$, $j \neq i$
for $j = 1, \ldots, \frac{N}{G}$, $j \neq i$ **do**
  for $s = 1, \ldots, \kappa$ **do**
    $\overline{\boldsymbol{Y}}_j \leftarrow \overline{\boldsymbol{Y}}_j +$ sample from distribution $p(\boldsymbol{Y}_j|\boldsymbol{x}_j^*)$
  end for
  $\overline{\boldsymbol{Y}}_j \leftarrow \frac{\overline{\boldsymbol{Y}}_j}{\kappa}$
end for
for all $(k_1, \ldots, k_G) \in \{1, \ldots, K\}^G$ **do**
  $g_{k_1,\ldots,k_G} \leftarrow g(\overline{\boldsymbol{Y}}_1, \ldots, (y_{i_1}^{k_1}, \ldots, y_{i_G}^{k_G}), \ldots, \overline{\boldsymbol{Y}}_{\frac{N}{G}})$
  $g\_vals = g\_vals \cup \{g_{k_1,\ldots,k_G}\}$
end for
return g_vals

In experiments we compared the coevolution-based approach with the conventional and the lazy approach. This is discussed in the next section.

4 Numerical Experiments

4.1 Experimental Setup

The different approaches described in Section 3 are compared on the following benchmark problem: The K possible values $y_i^1, \ldots, y_i^K$ of the random variable Y_i associated with decision variable x_i, $i = 1, \ldots, N$, are chosen normally distributed with a mean of 0 and a standard deviation of 15. Additionally, for each random variable Y_i, helper variables $U_1^i, \ldots, U_K^i$ are chosen randomly as follows:

$$U_j^i = \mathcal{N}(20, \sigma_U^2) - (j-1), \text{ for } j = 1, \ldots, K, \tag{12}$$

and Y_i is set to y_i^k, if and only if U_k^i is the helper variable with the smallest distance to the decision variable x_i:

$$Y_i = y_i^k \leftrightarrow |x_i - U_k^i| \leq |x_i - U_j^i|, \text{ for all } j = 1, \ldots, K. \tag{13}$$

The exact values of the helper variables and thus, the values of the random variables for a given vector $\boldsymbol{x}$ of decision variables are not known to the optimization. Only the distribution of the helper variables as well as all possible values of the random variables are known to the optimization.

In the experiments we restrict ourselves to the case that the fitness function only depends on the random variables (Eq. 4). We evaluated the different approaches for robust optimization on the three different fitness functions given in Table 1. All of them are non-separable (i.e., implication (9) does not hold).

Table 1. Benchmark functions used in the experiments.

Function Name	Function
Scaled Schwefel 1.2	$g_1(Y_1, \ldots, Y_N) = \left(\sum_{i=1}^{N} (\sum_{j=1}^{i} Y_j)^2\right)/100$
Cubed Maximum	$g_2(Y_1, \ldots, Y_N) = \max(\lvert Y_1\rvert, \ldots, \lvert Y_N\rvert)^3$
Scaled Rosenbrock	$g_3(Y_1, \ldots, Y_N) = \sum_{i=1}^{N-1} \left[100(Y_{i+1} - Y_i^2)^2 + (1 - Y_i)^2\right]/100N$

For the actual optimization, Particle Swarm Optimization (PSO) [17] is used. We also tested CMA-ES (Covariance Matrix Adaptation Evolution Strategy) [18], but did not observe significant differences in the results compared to PSO. The update of the velocity $\boldsymbol{v}$ of a particle in order to update its position $\boldsymbol{x}$ is done as follows:

$$\boldsymbol{v}_{n+1} = w \cdot \boldsymbol{v}_n + c_1 \cdot u_1 \cdot (\boldsymbol{x}_{best} - \boldsymbol{x}_n) + c_2 \cdot u_2 \cdot (\boldsymbol{x}_{g\_best} - \boldsymbol{x}_n), \tag{14}$$

with random u_1 and u_2 uniformly distributed chosen from $[0, 1]$, $c_1 = c_2 = 2.05$ and $w = \frac{2}{|2-c-\sqrt{c^2-4c}|}$ with $c = c_1 + c_2$. $\boldsymbol{x}_{best}$ is the so far best position of the particle and $\boldsymbol{x}_{g\_best}$ is the so far best position of the complete population.

For the lazy and the coevolution-based approach, the population size is set to 20. For the conventional approach a population size of 10 is chosen, which yielded the best results in initial experiments. The number of generations per cycle of the coevolution-based approach is set to 500 and the number of Monte-Carlo samples per fitness evaluation (the lower part in Figure 1b) is also set to 500. The number κ of samples for the computation of the expected values of the random variables in the lazy and the coevolution-based approach (see algorithms 2 and 3) is set to 1000. The grouping of the decision variables in the coevolution-based approach is done randomly and is changed in each cycle.

4.2 Experimental Results

The three approaches are evaluated on the three benchmark functions in Table 1 with 10, 20, 50 and 100 decision variables. The optimizations are executed with different values for the number g_{max} of allowed evaluations of the fitness function. These values correspond to different settings of the group size G and the number C of cycles in the coevolution-based approach. In the experiments, the standard deviation σ_U of the helper variables is set to 0.5 and the number K of possible values per random variable is set to 5. The conventional approach is executed with different values (5, 10, 50 and 100) for the number κ of Monte-Carlo samples. For each setting, 100 optimization trials with different seeds for the random initialization of the helper variables and of the possible values of the random variables are executed. At the end of each trial, the helper variables initialized at the beginning of a trial are used to evaluate the decision variables yielded by the optimization.

The tables 2, 3, 4 and 5 show the medians of the results of the 100 trials with the different approaches on the Schwefel No. 1.2 function g_1 for 10, 20, 50 and 100 decision variables, respectively. In the tables, $conv_\kappa$ stands for the conventional approach with κ Monte-Carlo samples, *coevo* stands for the coevolution-based approach and *lazy* for the lazy approach.

The superscripts indicate whether the results of an approach are significantly better than the results of another approach, according to the two-sided Wilcoxon rank sum test with a significance level of 0.05. For example, a 1 means significant better results than with the first approach in the table (conventional approach with $\kappa = 5$). A "-" in the table indicates that the number g_{max} of allowed evaluations of the fitness function is too low to execute at least one generation of the optimization.

With 10 and 20 decision variables, the proposed coevolution-based approach yielded the best results for 10 of the 12 settings. With 50 and 100 variables, it is even more superior compared to the other evaluated approaches. With 10 decision variables, the conventional approach is significantly better than the lazy approach for $g_{max} \geq 6250$. This is not the case for higher numbers of decision variables.

The experimental results on the Cubed Maximum function g_2 are shown in the tables 6, 7, 8 and 9. Again, in most cases the coevolution-based approach yielded the best results. But it can be seen that the results do not improve with

Table 2. Medians of 100 optimization trials with the different approaches on benchmark function g_1 (Schwefel No. 1.2) with 10 decision variables. The superscripts indicate whether an approach performed significantly better compared to another approach (according to two-sided Wilcoxon rank-sum test with a significance level of 0.05).

G	C	g_{max}	$conv_5$	$conv_{10}$	$conv_{50}$	$conv_{100}$	$coevo$	$lazy$
1	1	50	35.73	-	-	-	**16.60**^{1}	25.05^{1}
1	2	100	30.54	35.73	-	-	**13.84**^{1,2,6}	25.24^{2}
1	4	200	25.91	31.69	-	-	**13.07**^{1,2,6}	19.56^{1,2}
1	8	400	21.58	22.68	-	-	**12.78**^{1,2}	18.48
2	1	125	31.71	34.66	-	-	**17.07**^{1,2}	21.13^{1,2}
2	2	250	24.44	24.09	-	-	**14.04**^{1,2}	20.28
2	4	500	20.75^{3}	22.96^{3}	35.39	-	**11.94**^{1,2,3,6}	18.62^{3}
2	8	1000	16.28^{3,4}	17.90^{3,4}	24.49	34.03	**10.77**^{1,2,3,4,6}	20.69^{4}
5	1	6250	11.55^{3,4,6}	12.37^{4}	15.21	16.88	**10.82**^{3,4,6}	16.30
5	2	12500	**10.07**^{6}	12.02^{6}	13.40	13.10	10.56^{3,4,6}	18.72
5	4	25000	**9.73**	11.79^{6}	10.81	12.15	10.22^{6}	16.52
5	8	50000	10.54	12.11^{6}	10.36^{6}	10.85^{6}	**10.00**^{6}	18.77

Table 3. Results on g_1 analogous to Table 2 with 20 decision variables.

G	C	g_{max}	$conv_5$	$conv_{10}$	$conv_{50}$	$conv_{100}$	$coevo$	$lazy$
1	1	100	134.71	145.52	-	-	**72.18**^{1,2}	77.15^{1,2}
1	2	200	120.15	139.53	-	-	**57.11**^{1,2,6}	78.34^{1,2}
1	4	400	99.87	96.75	-	-	**52.32**^{1,2,6}	67.00^{1,2}
1	8	800	79.74^{3}	89.08^{3}	134.33	-	**51.31**^{1,2,3,6}	86.19^{3}
2	1	250	120.63	102.17	-	-	**61.57**^{1,2}	74.63^{1,2}
2	2	500	89.33^{3}	97.64^{3}	145.52	-	**42.46**^{1,2,3,6}	73.87^{3}
2	4	1000	82.73^{3,4}	79.84^{3,4}	118.37^{4}	136.61	**40.69**^{1,2,3,4,6}	72.40^{3,4}
2	8	2000	59.04^{3,4}	75.54^{4}	80.95	99.65	**31.99**^{1,2,3,4,6}	71.03^{3,4}
5	1	12500	49.69	**45.03**^{4}	50.72	61.07	45.70	56.83
5	2	25000	50.18	**37.97**^{1,4}	45.42	48.27	38.42	50.16
5	4	50000	42.47	40.48	35.32	42.67	**34.60**^{1,4}	48.13
5	8	100000	44.77	39.09	33.19	42.50	**27.68**^{1,4,6}	43.08

Table 4. Results on g_1 analogous to Table 2 with 50 decision variables.

G	C	g_{max}	$conv_5$	$conv_{10}$	$conv_{50}$	$conv_{100}$	$coevo$	$lazy$
1	1	250	705.93	754.01	-	-	**326.68**^{1,2,6}	675.99
1	2	500	675.42^{2,3}	825.89	782.23	-	**329.83**^{1,2,3,6}	539.26^{2,3}
1	4	1000	600.76^{4}	571.68^{4}	661.59^{4}	923.60	**294.05**^{1,2,3,4,6}	441.36^{1,3,4}
1	8	2000	427.66^{3,4}	508.02^{4}	567.20^{4}	747.29	**262.38**^{1,2,3,4,6}	356.26^{1,2,3,4}
2	1	625	652.42^{3}	680.87	780.59	-	**407.45**^{1,2,3,6}	552.25^{3}
2	2	1250	572.30^{4}	552.51^{4}	647.36	898.28	**275.49**^{1,2,3,4,6}	418.28^{1,2,3,4}
2	4	2500	390.34^{4}	592.61^{4}	554.56^{4}	721.24	**253.68**^{1,2,3,4,6}	310.17^{2,3,4}
2	8	5000	422.12^{4}	515.12	528.57	557.56	**194.30**^{1,2,3,4,6}	342.74^{2,3,4}
5	1	31250	332.09	305.71	355.56	354.41	**244.02**^{3,4}	271.80^{4}
5	2	62500	255.99	288.86	333.69	317.69	**220.16**^{3,4}	234.56
5	4	125000	222.08	255.69	276.04	238.07	**220.95**	304.01
5	8	250000	237.04	247.01	231.36	217.12	**164.68**^{1,2,3,4,6}	245.84

Table 5. Results on g_1 analogous to Table 2 with 100 decision variables.

G	C	g_{max}	$conv_5$	$conv_{10}$	$conv_{50}$	$conv_{100}$	$coevo$	$lazy$
1	1	500	2507.85^{3}	3014.68^{3}	3841.61	-	1681.57^{1,2,3}	**1643.98**^{1,2,3}
1	2	1000	2432.49^{3,4}	2192.03^{3,4}	3617.54	4322.18	**1302.16**^{1,2,3,4}	1663.43^{1,2,3,4}
1	4	2000	2336.02^{4}	2444.14^{4}	2971.89	3326.27	**1077.93**^{1,2,3,4,6}	1574.03^{1,2,3,4}
1	8	4000	2057.88^{4}	2179.68	2451.37	2795.51	**963.33**^{1,2,3,4,6}	1281.49^{1,2,3,4}
2	1	1250	2320.20^{3,4}	2192.03^{3,4}	3355.35	3662.65	**1216.28**^{1,2,3,4}	1663.43^{1,2,3,4}
2	2	2500	2387.29^{4}	2477.34	2439.90	3219.95	**900.02**^{1,2,3,4,6}	1557.61^{1,2,3,4}
2	4	5000	1912.52^{4}	1693.14^{4}	2702.69	2704.55	**714.14**^{1,2,3,4,6}	1201.97^{1,2,3,4}
2	8	10000	2119.62	1720.91^{4}	2037.06	2197.62	**584.07**^{1,2,3,4,6}	1265.38^{1,2,3,4}
5	1	62500	1440.20	1309.60	1219.28	1347.65	**1130.05**	1232.24
5	2	125000	1404.63	1217.75	1171.65	1029.05	**907.52**^{1,2,3,4}	1170.57
5	4	250000	1169.78	1080.15	1032.91	1067.33	**666.39**^{1,2,3,4}	1064.13
5	8	500000	973.41	935.23	1121.89	869.31	**532.67**^{1,2,3,4}	851.56^{3}

Table 6. Medians of 100 optimization trials with the different approaches on benchmark function g_2 (Cubed Maximum) with 10 decision variables. The superscripts indicate whether an approach performed significantly better compared to another approach (according to two-sided Wilcoxon rank-sum test with a significance level of 0.05).

G	C	g_{max}	$conv_5$	$conv_{10}$	$conv_{50}$	$conv_{100}$	$coevo$	$lazy$
1	1	50	13817	-	-	-	**5425**^{1}	7044^{1}
1	2	100	8173^{2}	13044	-	-	**3210**^{1,2,6}	7283^{2}
1	4	200	6289^{2}	8324	-	-	**2389**^{1,2,6}	6046^{2}
1	8	400	5687	6500	-	-	**2389**^{1,2,6}	6362
2	1	125	7608^{2}	11052	-	-	**4857**^{1,2}	6937^{2}
2	2	250	6289	7421	-	-	**3163**^{1,2,6}	6723
2	4	500	5426^{3}	6500^{3}	13044	-	**2440**^{1,2,3,6}	6636^{3}
2	8	1000	4169^{3,4}	5376^{3,4}	9168^{4}	13044	**2484**^{1,2,3,4,6}	5626^{3,4}
5	1	6250	**2890**^{3,4,5,6}	3263^{3,4,6}	5280	6411	4648^{4}	5746
5	2	12500	3329^{4,6}	2997^{4,6}	3939^{4,6}	5241	**2991**^{3,4,6}	7154
5	4	25000	3390^{6}	2997^{6}	3198^{6}	3844^{6}	**2799**^{4,6}	6977
5	8	50000	3329^{6}	2860^{6}	2908^{6}	3313^{6}	**2799**^{6}	6504

Table 7. Results on g_2 analogous to Table 6 with 20 decision variables.

G	C	g_{max}	$conv_5$	$conv_{10}$	$conv_{50}$	$conv_{100}$	$coevo$	$lazy$
1	1	100	17682	20521	-	-	**9871**^{1,2,6}	14257^{1,2}
1	2	200	15167	18059	-	-	**6303**^{1,2,6}	14199^{2}
1	4	400	10644^{2}	15092	-	-	**4148**^{1,2,6}	13574
1	8	800	8983^{3}	11192^{3}	18960	-	**4227**^{1,2,3,6}	10146^{3}
2	1	250	13817	17105	-	-	**9716**^{1,2,6}	14109^{2}
2	2	500	9808^{2,3}	13109^{3}	20390	-	**6303**^{1,2,3,6}	12075^{3}
2	4	1000	8636^{2,3,4}	10949^{3,4}	16918^{4}	19809	**4227**^{1,2,3,4,6}	11531^{3,4}
2	8	2000	7530^{3,4}	7587^{3,4}	13527^{4}	16490	**4227**^{1,2,3,4,6}	8868^{3,4}
5	1	12500	6763^{4,5}	**6693**^{4,5,6}	7544^{4,5}	9269	9845	10862
5	2	25000	6411^{6}	6384^{6}	6411^{6}	7680	**5839**^{4,6}	10073
5	4	50000	6411^{6}	6539^{6}	5821^{6}	6220^{6}	**4178**^{1,2,3,4,6}	11235
5	8	100000	6929^{6}	6290^{6}	5807^{1,6}	5733^{1,6}	**4178**^{1,2,3,4,6}	12946

Table 8. Results on g_2 analogous to Table 6 with 50 decision variables.

G	C	g_{max}	$conv_5$	$conv_{10}$	$conv_{50}$	$conv_{100}$	$coevo$	$lazy$
1	1	250	27436	30258	-	-	**20482**^{1,2,6}	26887
1	2	500	23945^{3}	23593^{3}	35884	-	**12896**^{1,2,3,6}	25945^{3}
1	4	1000	19622^{3,4,6}	20854^{3,4,6}	30020^{4}	35884	**7340**^{1,2,3,4,6}	23794^{3,4}
1	8	2000	17646^{3,4,6}	17291^{3,4,6}	23579^{4}	30866	**7199**^{1,2,3,4,6}	22659^{4}
2	1	625	22778^{3}	23224^{3}	35884	-	**20187**^{2,3}	23307^{3}
2	2	1250	19262^{3,4,6}	19524^{3,4,6}	28100^{4}	35884	**13031**^{1,2,3,4,6}	23682^{3,4}
2	4	2500	16042^{3,4,6}	16868^{3,4,6}	22565^{4}	26548	**7304**^{1,2,3,4,6}	21822^{4}
2	8	5000	15382^{3,4,6}	14335^{3,4,6}	19642^{4}	23811	**7260**^{1,2,3,4,6}	21245^{4}
5	1	31250	12343^{4,5,6}	**11575**^{4,5,6}	12335^{4,5,6}	15640^{5}	19152	18980
5	2	62500	11950	**11605**^{6}	12153^{6}	13238^{6}	11634^{4,6}	18300
5	4	125000	11950^{6}	11575^{6}	11421^{6}	11917^{6}	**7428**^{1,2,3,4,6}	18247
5	8	250000	11961	11495^{6}	11237^{6}	11498^{6}	**7160**^{1,2,3,4,6}	17738

Table 9. Results on g_2 analogous to Table 6 with 100 decision variables.

G	C	g_{max}	$conv_5$	$conv_{10}$	$conv_{50}$	$conv_{100}$	$coevo$	$lazy$
1	1	500	32419^{2,3,6}	36690^{3}	54329	-	**31159**^{2,3,6}	39621^{3}
1	2	1000	31353^{3,4,6}	32777^{3,4}	45533^{4}	54006	**18258**^{1,2,3,4,6}	37695^{3,4}
1	4	2000	25868^{2,3,4,6}	27504^{3,4,6}	39369^{4}	45940	**11158**^{1,2,3,4,6}	33938^{4}
1	8	4000	23801^{3,4,6}	24153^{3,4,6}	32101^{4}	37976	**10811**^{1,2,3,4,6}	33003^{4}
2	1	1250	**27794**^{2,3,4,6}	32661^{3,4}	43599^{4}	50762	31159^{3,4,6}	37624^{3,4}
2	2	2500	24673^{3,4,6}	25772^{3,4,6}	37185^{4}	40577	**18683**^{1,2,3,4,6}	33513^{4}
2	4	5000	22889^{3,4,6}	23684^{3,4,6}	31152^{4}	35774	**11294**^{1,2,3,4,6}	32508
2	8	10000	20412^{3,4,6}	21163^{3,4,6}	27660^{4,6}	31840	**11278**^{1,2,3,4,6}	33098
5	1	62500	18372^{4,5,6}	**18163**^{4,5,6}	19750^{5,6}	21410^{5,6}	31520	27921^{5}
5	2	125000	18198^{6}	**17828**^{4,6}	19530^{6}	19673^{6}	18843^{6}	24101
5	4	250000	18403^{6}	17720^{6}	19298^{6}	18499^{6}	**11385**^{1,2,3,4,6}	23739
5	8	500000	18339^{6}	17962^{6}	19152^{6}	18383^{6}	**10811**^{1,2,3,4,6}	23739

Table 10. Medians of 100 optimization trials with the different approaches on benchmark function g_3 (Rosenbrock) with 10 decision variables. The superscripts indicate whether an approach performed significantly better compared to another approach (according to two-sided Wilcoxon rank-sum test with a significance level of 0.05).

G	C	g_{max}	$conv_5$	$conv_{10}$	$conv_{50}$	$conv_{100}$	$coevo$	$lazy$
1	1	50	57256	-	-	-	$\mathbf{4818}^{1,6}$	30652^{1}
1	2	100	32068^{2}	46138	-	-	$\mathbf{4684}^{1,2,6}$	$20554^{1,2}$
1	4	200	21377	26245	-	-	$\mathbf{4684}^{1,2,6}$	18364^{2}
1	8	400	18300	26135	-	-	$\mathbf{4684}^{1,2,6}$	18100
2	1	125	30100	39979	-	-	$\mathbf{5137}^{1,2,6}$	$18492^{1,2}$
2	2	250	20488^{2}	27624	-	-	$\mathbf{4818}^{1,2,6}$	18212^{2}
2	4	500	15647^{3}	21396^{3}	47510	-	$\mathbf{4818}^{1,2,3,6}$	15999^{3}
2	8	1000	$11630^{3,4}$	$13549^{3,4}$	31299^{4}	46590	$\mathbf{4846}^{1,2,3,4,6}$	$14000^{3,4}$
5	1	6250	$8019^{3,4}$	$6736^{3,4,6}$	12410^{4}	17922	$\mathbf{5234}^{1,3,4,6}$	12605
5	2	12500	$7060^{4,6}$	$6832^{3,4,6}$	10510^{4}	14437	$\mathbf{4818}^{3,4,6}$	13174
5	4	25000	6913^{6}	6889^{6}	8016^{6}	9759^{6}	$\mathbf{5060}^{3,4,6}$	13513
5	8	50000	8392^{6}	6168^{6}	$5697^{1,6}$	6702^{6}	$\mathbf{5067}^{1,6}$	18181

Table 11. Results on g_3 analogous to Table 10 with 20 decision variables.

G	C	g_{max}	$conv_5$	$conv_{10}$	$conv_{50}$	$conv_{100}$	$coevo$	$lazy$
1	1	100	57784^{2}	75051	-	-	$\mathbf{6908}^{1,2,6}$	$30752^{1,2}$
1	2	200	43596^{2}	55369	-	-	$\mathbf{6982}^{1,2,6}$	$22469^{1,2}$
1	4	400	32469^{2}	40201	-	-	$\mathbf{7026}^{1,2,6}$	$21306^{1,2}$
1	8	800	24902^{3}	29510^{3}	60345	-	$\mathbf{7026}^{1,2,3,6}$	$19921^{1,2,3}$
2	1	250	40389^{2}	52267	-	-	$\mathbf{6943}^{1,2,6}$	$20310^{1,2}$
2	2	500	28443^{3}	34056^{3}	69687	-	$\mathbf{6982}^{1,2,3,6}$	$20231^{1,2,3}$
2	4	1000	$23606^{3,4}$	$25749^{3,4}$	53045^{4}	68999	$\mathbf{7026}^{1,2,3,4,6}$	$18019^{1,2,3,4}$
2	8	2000	$15279^{3,4}$	$20317^{3,4}$	40388^{4}	51314	$\mathbf{7024}^{1,2,3,4,6}$	$19157^{3,4}$
5	1	12500	13401^{4}	12336^{4}	16939^{4}	19943	$\mathbf{7131}^{1,2,3,4,6}$	19907
5	2	25000	13881	12014^{6}	13936^{6}	15450	$\mathbf{7088}^{1,2,3,4,6}$	21131
5	4	50000	12601^{6}	10956^{6}	11245^{6}	11784^{6}	$\mathbf{7082}^{1,2,3,4,6}$	22700
5	8	100000	13488^{6}	11331^{6}	10316^{6}	$10446^{1,6}$	$\mathbf{7111}^{1,2,3,4,6}$	22610

Table 12. Results on g_3 analogous to Table 10 with 50 decision variables.

G	C	g_{max}	$conv_5$	$conv_{10}$	$conv_{50}$	$conv_{100}$	$coevo$	$lazy$
1	1	250	64888	75595	-	-	$\mathbf{8409}^{1,2,6}$	$48225^{1,2}$
1	2	500	51861^{3}	61087^{3}	100546	-	$\mathbf{8389}^{1,2,3,6}$	$40756^{1,2,3}$
1	4	1000	$43120^{2,3,4}$	$48826^{3,4}$	80348^{4}	103644	$\mathbf{8389}^{1,2,3,4,6}$	$34539^{1,2,3,4}$
1	8	2000	$34328^{2,3,4}$	$38838^{3,4}$	64652^{4}	76346	$\mathbf{8389}^{1,2,3,4,6}$	$30280^{1,2,3,4}$
2	1	625	$46592^{2,3}$	57531^{3}	92147	-	$\mathbf{8505}^{1,2,3,6}$	$36261^{1,2,3}$
2	2	1250	$39092^{2,3,4}$	$45377^{3,4}$	74919^{4}	95448	$\mathbf{8412}^{1,2,3,4,6}$	$34546^{2,3,4}$
2	4	2500	$32853^{3,4}$	$36845^{3,4}$	60446^{4}	71424	$\mathbf{8387}^{1,2,3,4,6}$	$30469^{2,3,4}$
2	8	5000	$27736^{3,4}$	$29926^{3,4}$	49395^{4}	59516	$\mathbf{8387}^{1,2,3,4,6}$	$23774^{2,3,4}$
5	1	31250	$20502^{3,4}$	$21629^{3,4}$	26046^{4}	32472	$\mathbf{8588}^{1,2,3,4,6}$	$20865^{3,4}$
5	2	62500	19175^{4}	$19734^{3,4}$	22280	25708	$\mathbf{8741}^{1,2,3,4,6}$	22259
5	4	125000	19048	18543	19955	21010	$\mathbf{8915}^{1,2,3,4,6}$	21349
5	8	250000	18399	18993	17705^{6}	18626	$\mathbf{8915}^{1,2,3,4,6}$	21029

Table 13. Results on g_3 analogous to Table 10 with 100 decision variables.

G	C	g_{max}	$conv_5$	$conv_{10}$	$conv_{50}$	$conv_{100}$	$coevo$	$lazy$
1	1	500	$72617^{2,3}$	84144^{3}	113734	-	$\mathbf{9332}^{1,2,3,6}$	$52837^{1,2,3}$
1	2	1000	$61254^{2,3,4}$	$71151^{3,4}$	102737^{4}	114470	$\mathbf{9222}^{1,2,3,4,6}$	$47996^{1,2,3,4}$
1	4	2000	$51946^{2,3,4}$	$62341^{3,4}$	86136^{4}	102519	$\mathbf{9222}^{1,2,3,4,6}$	$42478^{1,2,3,4}$
1	8	4000	$44142^{2,3,4}$	$51229^{3,4}$	72186^{4}	84957	$\mathbf{9222}^{1,2,3,4,6}$	$39415^{2,3,4}$
2	1	1250	$60493^{2,3,4}$	$67511^{3,4}$	98498^{4}	109598	$\mathbf{9336}^{1,2,3,4,6}$	$46690^{1,2,3,4}$
2	2	2500	$51326^{2,3,4}$	$55208^{3,4}$	82215^{4}	98302	$\mathbf{9221}^{1,2,3,4,6}$	$42038^{1,2,3,4}$
2	4	5000	$42615^{2,3,4}$	$48619^{3,4}$	67437^{4}	79279	$\mathbf{9221}^{1,2,3,4,6}$	$38793^{2,3,4}$
2	8	10000	$35346^{3,4}$	$39242^{3,4}$	58246^{4}	66962	$\mathbf{9229}^{1,2,3,4,6}$	$35067^{3,4}$
5	1	62500	$29190^{3,4}$	$27686^{3,4}$	36627^{4}	40393	$\mathbf{9271}^{1,2,3,4,6}$	$28983^{3,4}$
5	2	125000	28773^{4}	$26938^{3,4}$	31241^{4}	36239	$\mathbf{9139}^{1,2,3,4,6}$	29264^{4}
5	4	250000	27716	25926^{4}	29269	30209	$\mathbf{9128}^{1,2,3,4,6}$	28556
5	8	500000	28626	27393	26740	27725	$\mathbf{9128}^{1,2,3,4,6}$	28001

an increasing group size. With a group size of 1 and 8 cycles, the coevolution-based approach yielded better results than the other approaches with a g_{max} corresponding to a group size of 5 and 8 cycles for all considered numbers of decision variables. With an increasing number of variables, the conventional approach becomes more and more superior to the lazy approach.

Tables 10, 11, 12 and 13 show the results on the Rosenbrock function. It can be seen that the coevolution-based approach yields the best results for all settings. With 20, 50 and 100 decision variables, it is significantly better than all other approaches for all considered values of g_{max}. Interestingly, the results do not improve with an increasing group size or an increasing number of cycles. This indicates that the Rosenbrock function is well suited for the separate optimization of the individual decision variables, although it is non-separable.

In a further experiment, we investigated the impact of the standard deviation σ_U of the distribution of the helper variables on the optimization results and executed the optimizations with different values for σ_U. With an increasing σ_U, the uncertainty increases. Table 14 shows the median results of 100 optimization trials on benchmark function g_1 with 20 decision variables executed with a group size of $G = 2$ and with $C = 8$ cycles and a corresponding g_{max} of 2000 for different values of σ_U. The number K of possible values per random

Table 14. Medians of 100 optimization trials with the different approaches on benchmark function g_1 (Schwefel No. 1.2) with 20 decision variables, a group size of $G = 2$ and $C = 8$ cycles ($g_{max} = 2000$) with different values for the standard deviation σ_U of the distribution of the helper variables. Superscripts indicate statistical significance according to two-sided Wilcoxon rank-sum test with a significance level of 0.05.

σ_U	$conv_5$	$conv_{10}$	$conv_{50}$	$conv_{100}$	$coevo$	$lazy$
0.2	$20.25^{2,3,4}$	$27.26^{3,4}$	48.38^{4}	63.64	$\mathbf{10.16}^{1,2,3,4,6}$	$19.21^{2,3,4}$
0.5	$59.04^{3,4}$	75.54^{4}	80.95	99.65	$\mathbf{31.99}^{1,2,3,4,6}$	$71.03^{3,4}$
1.0	164.40	148.94	133.12	185.66	$\mathbf{95.38}^{1,2,3,4,6}$	122.88^{4}
1.5	203.40	209.99	176.33	214.98	$\mathbf{104.66}^{1,2,3,4,6}$	$151.50^{1,2,4}$
2.0	268.21	235.65	307.91	231.79	$\mathbf{135.79}^{1,2,3,4,6}$	$189.08^{1,3,4}$
2.5	266.46	288.74	226.92	259.04	$\mathbf{147.34}^{1,2,3,4}$	$196.42^{2,4}$
3.0	221.03	204.39	236.11	250.26	$\mathbf{159.37}^{3,4}$	186.70^{4}
3.5	229.43	239.08	259.02	341.83	$\mathbf{189.84}^{1,3,4}$	222.13^{4}
4.0	378.14	287.21	276.47	331.84	$\mathbf{181.13}^{1,2,3,4}$	$204.46^{1,2,3,4}$
4.5	293.08	314.05	283.23	325.91	$\mathbf{172.27}^{1,2,3,4}$	$209.03^{1,2,3,4}$
5.0	293.93	$213.36^{1,3,4}$	296.96	305.57	$\mathbf{207.02}^{1,3,4}$	$226.13^{1,3,4}$

variable is again set to 5. For all considered values of σ_U, the coevolution-based approach yields the best results. But with an increasing σ_U, the results with the different approaches become more and more similar because the quality of the optimization results becomes more and more random due to an increasing uncertainty. The results on the benchmark functions g_2 and g_3 are not shown because they are similar to those on g_1.

In a final experiment, we evaluated the different approaches for different numbers K of possible values per random variable. Again, the optimizations were executed with 20 decision variables, a group size of $G = 2$ with $C = 8$

cycles and corresponding values for g_{max}. The results on benchmark function g_1 are shown in Table 15. For $K > 2$, the results with the coevolution-based

Table 15. Medians of 100 optimization trials with the different approaches on benchmark function g_1 (Schwefel No. 1.2) with 20 decision variables, a group size of $G = 2$ and $C = 8$ cycles with different numbers K of possible values per random variable. Superscripts indicate statistical significance according to two-sided Wilcoxon rank-sum test with a significance level of 0.05.

K	g_{max}	$conv_5$	$conv_{10}$	$conv_{50}$	$conv_{100}$	$coevo$	$lazy$
2	320	72.70^{2}	97.43	-	-	$\mathbf{52.54}^{1,2}$	$56.92^{1,2}$
3	720	70.35^{3}	71.77^{3}	127.19	-	$\mathbf{38.59}^{1,2,3,6}$	59.92^{3}
4	1280	$8.50^{3,4}$	$78.67^{3,4}$	111.87^{4}	165.43	$\mathbf{37.38}^{1,2,3,4,6}$	$50.52^{1,2,3,4}$
5	2000	$59.04^{3,4}$	75.54^{4}	80.95	99.65	$\mathbf{31.99}^{1,2,3,4,6}$	$71.03^{3,4}$
6	2880	63.72^{4}	$64.93^{3,4}$	84.82	106.23	$\mathbf{26.35}^{1,2,3,4,6}$	$66.47^{3,4}$
7	3920	$55.56^{3,4}$	$51.15^{3,4}$	79.24	79.67	$\mathbf{29.65}^{1,2,3,4,6}$	$61.98^{3,4}$
8	5120	$52.27^{3,4}$	$51.33^{3,4}$	77.08	76.96	$\mathbf{28.43}^{1,2,3,4,6}$	74.75
9	6480	$52.07^{3,4}$	49.83^{4}	71.03^{4}	95.62	$\mathbf{35.53}^{1,2,3,4,6}$	$46.42^{3,4}$
10	8000	54.45^{4}	47.94^{4}	53.67^{4}	70.94	$\mathbf{26.26}^{1,2,3,4,6}$	51.47^{4}
11	9680	$40.38^{3,4}$	$52.05^{3,4}$	64.63	68.21	$\mathbf{26.55}^{1,2,3,4,6}$	68.56
12	11520	$45.63^{3,4,6}$	56.30^{4}	62.46	76.54	$\mathbf{25.78}^{1,2,3,4,6}$	75.23
13	13520	50.08^{4}	$40.76^{3,4}$	55.83^{4}	79.62	$\mathbf{23.65}^{1,2,3,4,6}$	59.87^{4}
14	15680	$40.49^{3,4}$	43.42^{4}	56.55	61.28	$\mathbf{18.73}^{1,2,3,4,6}$	50.64
15	18000	53.33^{4}	48.11^{4}	47.30^{4}	65.21	$\mathbf{24.71}^{1,2,3,4,6}$	60.03

approach are significantly better than the results with the other approaches. Again, the results on g_2 and g_3 are not shown, because they are similar to those on g_1.

5 Summary and Outlook

In the present work, we proposed a coevolution-based approach for the robust optimization of problems with a fitness function basically depending on discrete random variables, which conditionally depend on the decision variables. Numerical experiments have shown that it is able to outperform the conventional approach based on full Monte-Carlo simulation and the "lazy" approach with a very rough approximation of the expected fitness on common benchmark functions. The proposed approach is particularly beneficial if the number of allowed evaluations of the function in the random variables is small. With an increasing amount of uncertainty or an increasing number of possible values of the discrete random variables, the proposed approach is still highly competitive with the other evaluated approaches.

In a next step, we plan to evaluate the approach on a real-world problem, more precisely, the optimization of dynamic prices for the charging of electric vehicles at public charging stations. Furthermore, we plan to investigate the effect of more advances strategies for the grouping of the decision variables, like delta grouping [19], and acceleration techniques, like described in Section 1, on the performance of the proposed approach.

References

1. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments – a survey. IEEE Trans. on Evol. Comp. 9(3), 303–317 (2005)
2. Beyer, H.G., Sendhoff, B.: Robust optimization – a comprehensive survey. Computer Methods in Applied Mechanics and Engineering 196(33), 3190–3218 (2007)
3. Leon, V.J., Wu, S.D., Storer, R.H.: Robustness measures and robust scheduling for job shops. IIE Transactions 26(5), 32–43 (1994)
4. Wiesmann, D., Hammel, U., Bäck, T.: Robust design of multilayer optical coatings by means of evolutionary algorithms. IEEE Trans. on Evol. Comp. 2(4), 162–167 (1998)
5. Hacker, S., Lewis, K.: Robust design through the use of a hybrid genetic algorithm. In: Proc. 28th Design Automation Conference. pp. 703–712. The American Society of Mechanical Engineers (2002)
6. Singh, A., Minsker, B.: Uncertainty based multi-objective optimization of groundwater remediation at the umatilla chemical depot. In: Proc. World Water and Environmetal Resources Congress. pp. 3589–3598 (2004)
7. Wang, H., Kim, N., Kim, Y.J.: Safety envelope for load tolerance and its application to fatigue reliability design. Journal of Mechanical Design 128(4), 919–927 (2006)
8. Kavakeb, S., Nguyen, T.T., Yang, Z., Jenkinson, I.: Identifying the robust number of intelligent autonomous vehicles in container terminals. In: EvoApplications 2014. pp. 829–840. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
9. Loughlin, D.H., Ranjithan, S.R.: Chance-constrained genetic algorithms. In: Proc. GECCO 1999. pp. 369–376. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)
10. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21(2), 239–245 (1979)
11. Branke, J.: Creating robust solutions by means of evolutionary algorithms. In: PPSN V. pp. 119–128. Springer Berlin Heidelberg, Berlin, Heidelberg (1998)
12. Aizawa, A.N., Wah, B.W.: Dynamic control of genetic algorithms in a noisy environment. In: Proc. 5th International Conference on Genetic Algorithms. pp. 48–55. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
13. Lee, K.H., Park, G.J.: A global robust optimization using kriging based approximation model. JSME International Journal Series Mechanical Systems, Machine Elements and Manufacturing 49(3), 779–788 (2006)
14. Paenke, I., Branke, J., Jin, Y.: Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation. IEEE Trans. on Evol. Comp. 10(4), 405–420 (2006)
15. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Information Sciences 178(15), 2985–2999 (2008)
16. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: PPSN III. pp. 249–257. Springer Berlin Heidelberg, Berlin, Heidelberg (1994)
17. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. IEEE Conference on Neural Networks. vol. 4, pp. 1942–1948 (1995)
18. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
19. Omidvar, M.N., Li, X., Yao, X.: Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In: Proc. IEEE CEC. pp. 1762–1769 (2010)