Center for Computational Engineering Science Mathematics Division

RWTH AACHEN UNIVERSITY

Diese Arbeit wurde vorgelegt am
Lehrstuhl für Mathematik (MathCCES)

Mehrzieloptimierung mit Ersatzmodellen und adaptiver Abtastung

Multi-objective optimization with surrogate models and adaptive sampling

Bachelorarbeit
Computational Engineering Science

Juni 2018

Vorgelegt von Presented by	Markus Zimmermann Leo-Wenke-Str. 14, 41462 Neuss Matrikelnummer: 342064 markus.zimmermann4@rwth-aachen.de
Erstprüfer First examiner	Prof. Dr. Manuel Torrilhon Lehrstuhl für Mathematik (MathCCES) RWTH Aachen University
Zweitprüfer Second examiner	Prof. Dr. Benjamin Stamm Lehrstuhl für Mathematik (MathCCES) RWTH Aachen University
Koreferent Co-supervisor	Dr. Stefan Bühler Electrical Drives - Bühl/Bühlertal Robert Bosch GmbH

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen und Abbildungen. Diese Arbeit hat in dieser oder einer ähnlichen Form noch nicht im Rahmen einer anderen Prüfung vorgelegen.

Aachen, im Juni 2018

Markus Zimmermann

Contents

List of Figures IV

List of Tables V

1. Introduction 1

2. Multi-objective optimization 1
2.1. Motivating example . . . 2
2.2. Pareto optimality . . . 3
2.3. Optimization methods . . . 6

3. Surrogate models 9
3.1. Polynomial based models . . . 10
3.2. Kriging - Gaussian process regression . . . 12

4. Adaptive sampling strategies 14
4.1. Strategy based on Pareto optimal solutions . . . 15
4.2. Strategy based on maximum confidence intervals . . . 16
4.3. Mixed strategy . . . 17

5. Algorithm 18
5.1. Benchmark problems . . . 20
5.2. Simulation results . . . 25

6. Practical application example 34
6.1. Simulation results . . . 35

7. Conclusion and outlook 40

A. Appendix 41

References 46

List of Figures

1. Multi-objective optimization simple example . . . 3
2. Trade-off curve for a *multi-objective optimization* problem . . . 4
3. Example of a *Pareto front* in two dimensions . . . 5
4. Latin Hypercube sampling (LHS) . . . 10
5. Polynomial based *surrogate models* with different polynomial degree . . . 11
6. Data *over-fitting* using polynomial based *surrogate models* . . . 12
7. Trained and untrained state of an *Kriging surrogate model* . . . 14
8. Adaptive *sampling* focused on *exploitation* . . . 16
9. Adaptive *sampling* focused on *exploration* . . . 18
10. Algorithm flow chart . . . 19
11. Visualization of the convergence and diversity metric . . . 24
12. Final *Pareto front* for all benchmark problems . . . 31
13. Visualization of the convergence metric results . . . 32
14. Visualization of the diversity metric results . . . 33
15. Comparison of the *Pareto fronts* for the application example . . . 37
16. *Pareto fronts* at different iterations of the application example optimization . . . 38
17. *Pareto fronts* at different iterations of the application example optimization (2) . . . 39

List of Tables

1. Overview of all benchmark problems 23
2. Settings of the algorithm for the test simulations 26
3. Number of found *Pareto optimal* solutions 29
4. Settings of the algorithm for the application example 35
A.1. Resulting convergence metric for the NSGA2 strategy 41
A.2. Resulting diversity metric for the NSGA2 strategy 41
A.3. Resulting convergence metric for the CI strategy 42
A.4. Resulting diversity metric for the CI strategy 42
A.5. Resulting convergence metric for the PAR strategy 42
A.6. Resulting diversity metric for the PAR strategy 43
A.7. Resulting convergence metric for the MIX strategy 43
A.8. Resulting diversity metric for the MIX strategy 43
A.9. Surrogate model quality at the *Pareto front* for the CI strategy 44
A.10. Surrogate model quality at the *Pareto front* for the PAR strategy . . . 44
A.11. Surrogate model quality at the *Pareto front* for the MIX strategy . . . 44
A.12. Surrogate overall model quality for the CI strategy 45
A.13. Surrogate overall model quality for the PAR strategy 45
A.14. Surrogate overall model quality for the MIX strategy 45

1. Introduction

Multi-objective optimization is an active field of research because such problems exist in many different domains. Most algorithms proposed to solve such problems either require additional information about the problem or use a high number of model evaluations. Sometimes there is no information on how the solution might look like. Additionally, as the models try to depict reality better, they get more and more complex so that one simulation can take days before it is finished. To address these issues an iterative algorithm to solve multi-objective optimization problems is proposed within this thesis.

First a brief overview over the definition of multi-objective optimization problems and some methods to solve them is given in section 2. Following, in section 3, the concept of surrogate models is introduced. Surrogate models are used to decrease the number of simulations on the complex model within the algorithm. This requires a way to select appropriate values for model parameter to build reasonable surrogate models. Different strategies to adapt the sampling after an iteration are proposed in section 4. Details about the structure of the algorithm are given in section 5. To test the performance of the algorithm, different analytical problems are solved. The results are compared in the same section. Finally, the algorithm is applied to an application example. The simulation results are presented in section 6.

2. Multi-objective optimization

Many engineering, business or logistic applications have to deal with optimization problems. Multiple objectives that are conflicting with each other often need to be satisfied to find an appropriate solution. The following problems are examples with two or more conflicting objectives:

- Designing a fast car with low fuel consumption and high range
- Investing money to generate the highest profit with the lowest risk
- Minimizing the cost of production while increasing the maximum output

A compromise between these objectives is made to find a solution. Such an optimization problem is called *multi-objective optimization problem* (MOP). *Multi-objective optimization* is an area of *multi criteria decision making* (MCDM). It describes the analytical process of solving optimization problems involving more than one objective. Generally, unique solutions for these problems are non-existent. Traditionally, to solve MOPs, all objectives are combined into a single objective function, which then can be solved with classical *single-objective optimization* algorithms. However, solving MOPs that way has several limitations:

- Algorithms for *single-objective optimization* find only one solution
- For some problems the combined single objective function has no solution

- A priori knowledge about importance of the different objectives is required
- The trade-off between objectives cannot be evaluated because of the single solution that is found

The goal of *multi-objective optimization* is to find a set of solutions rather than a single one. As a last step of MCDM a human *decision maker* (DM) can then select a single solution out of the set that satisfies his subjective preferences.

2.1. Motivating example

To introduce the topic of *multi-objective optimization* consider the following simple example:

$$\begin{aligned} &\min f_1, f_2 \\ f_1(x) &= (x-2)^2 - 2 \\ f_2(x) &= (x+1)^2 + 3 \\ x &\in [-5, 5] \end{aligned}$$

The functions f_1 and f_2 should be minimized simultaneously on the interval $[-5, 5]$. The minimum for each objective can easily be seen as $x_1 = 2$ and $x_2 = -1$. If the value of x is decreased beyond -1 the function values for both objective functions increase. The same happens when increasing the value of x beyond 2. As the goal is to minimize both functions simultaneously the intervals $[-5, -1)$ and $(2, 5]$ do not correspond to the set of optimal solutions. For x between -1 and 2 the behaviour of each function is different. While the first objective function f_1 increases with increasing values of x the second objective f_2 decreases. A trade-off between both objectives exists for $x \in [-1, 2]$. Figure 1 visualizes both objective functions in the interval $[-5, 5]$. Their minima are marked with a dot. All values of x in the grey interval are optimal solutions for the MOP. Because the example consists of only two objective functions the optimal solutions can be displayed as a trade-off curve as shown in Figure 2. Decreasing the value of the first objective function increases the value of the second objective. A *multi-objective optimization* algorithm solving this example problem should return equally spaced points on this curve so that a DM can select an appropriate solution. With a higher number of objectives and decision variables, MOPs quickly get more complex.

Generalizing the formulation of MOP from the simple example gives the following definition of MOPs:

$$\begin{aligned} \text{Minimize/maximize } & f_m(x), && m = 1, 2, \ldots, M \\ \text{subject to } & g_j(x) \geq 0, && j = 1, 2, \ldots, J \\ & h_k(x) = 0, && k = 1, 2, \ldots, K \\ \text{with } & x \in \Omega \end{aligned} \tag{1}$$

The vector x contains the decision variables from the decision space Ω. Depending on the problem the objective functions f_m have to be either minimized or maximized. The

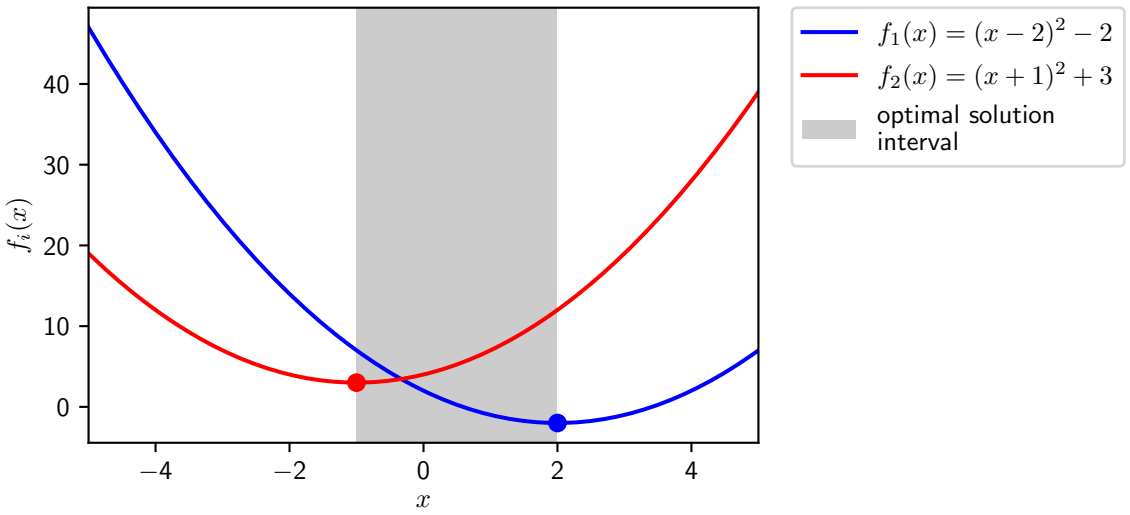


Figure 1: A simple multi-objective optimization example with a two-dimensional objective space is shown. The minima of the quadratic objective functions are marked with a dot. The interval including the optimal solutions of the one-dimensional decision space is coloured in grey.

objective functions can be linear or nonlinear. They form a M-dimensional objective space and map variables from the decision space into it. The functions g_j and h_k are called constraint functions. Constraints divide the decision space Ω into feasible and unfeasible regions. A solution has to satisfy all constraints and thus has to lie in a feasible region. Unlike *single-objective optimization* problems, MOPs in general have an infinite number of solutions. The solutions are found using the theory of *Pareto optimality*.

2.2. Pareto optimality

For *single-objective optimization* problems, the computed solutions can easily be compared. In case of minimization a solution vector u is better than another vector v if its corresponding objective value $f(u)$ is less than the other objective value $f(v)$. For MOPs more than one objective value corresponds to a solution. Thus, the above comparison has to be extended to multiple objective values. The theory of *Pareto optimality* is used to rank different solutions. To understand *Pareto optimality* some concepts need to be defined first. *Dominance* is used to compare multiple objective values, which is defined as follows.

A vector u dominates a vector v if and only if u is partially less than v.

$$u \preceq v \Leftrightarrow u_i \leq v_i \; \forall i \in \{1, \ldots, n\} \wedge \exists j \in \{1, \ldots, n\} : u_j < v_j.$$

When the goal is to maximize the objective value all definitions are used with '$>$' and '$\geq$' instead of '$<$', '$\leq$'. For convenience, minimization is always assumed if not

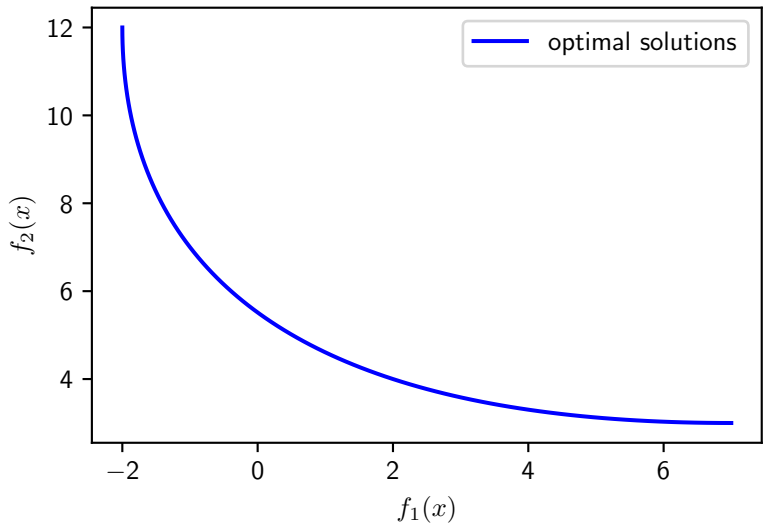


Figure 2: The trade-off between the two objective functions from the simple *multi-objective optimization* problem is shown. Decreasing the first objective results in an increase of the second objective value. Any value on this curve represents an optimal solution of the problem.

stated otherwise.

A vector u strictly dominates a vector v if and only if u is less than v in every dimension.

$$u \prec v \Leftrightarrow \forall i \in \{1, \ldots, n\}, u_i < v_i.$$

Pareto optimality uses *dominance* to compare objective values of solutions obtained in the decision space.

A solution x^* is *Pareto optimal* if and only if there is no other solution x' for which $v = F(x') = (f_1(x'), \ldots, f_M(x'))$ dominates $u = F(x^*) = (f_1(x^*), \ldots, f_M(x^*))$.

In other words x^* is *Pareto optimal* if there is no other vector x' which decreases an objective value without increasing another objective. *Pareto optimal* solutions are also called *non-inferior* or *efficient* solutions. The corresponding vectors from the objective space are called non-dominated.

The *Pareto optimal* set $\mathcal{P}^*$ for a MOP $F(x)$ also referenced as *Pareto archive* contains all *Pareto optimal* solutions.

$$\mathcal{P}^* = \{x \in \Omega \,|\, \nexists x' \in \Omega : F(x') \preceq F(x)\}.$$

The solutions in the *Pareto archive* may have no other relation besides belonging to the same set. They do not have to lie in the same subspace or be a minimum of an objective function. Additionally, they can be linearly independent. This increases the difficulty to find all optimal solutions. All corresponding non-dominated vectors from the objective space form the *Pareto front* $\mathcal{PF}^*$.

$$\mathcal{PF}^* = \{u = F(x) \,|\, x \in \mathcal{P}^*\}$$

Usually a subset of the *Pareto front* is shown to the DM, so he can select the "best" solution out of it. For two objectives, such a plot is called trade-off curve. The previous example showed such a curve in Figure 2 where each point on the curve belongs to the *Pareto front*. For two- and three-dimensional objective spaces, the *Pareto front* can be visualized as either a curve or any two-dimensional subspace. Note that, depending on the problem, this has not to be a continuous region. For higher dimensional objective spaces, two- or three-dimensional slices are used to show the trade-off between the objective functions. Other visualization techniques use shape, colour, size or other properties of two- or three-dimensional objects to visualize higher dimensional data sets. However, within this thesis two-dimensional subsets are chosen to display trade-off curves between the objectives.

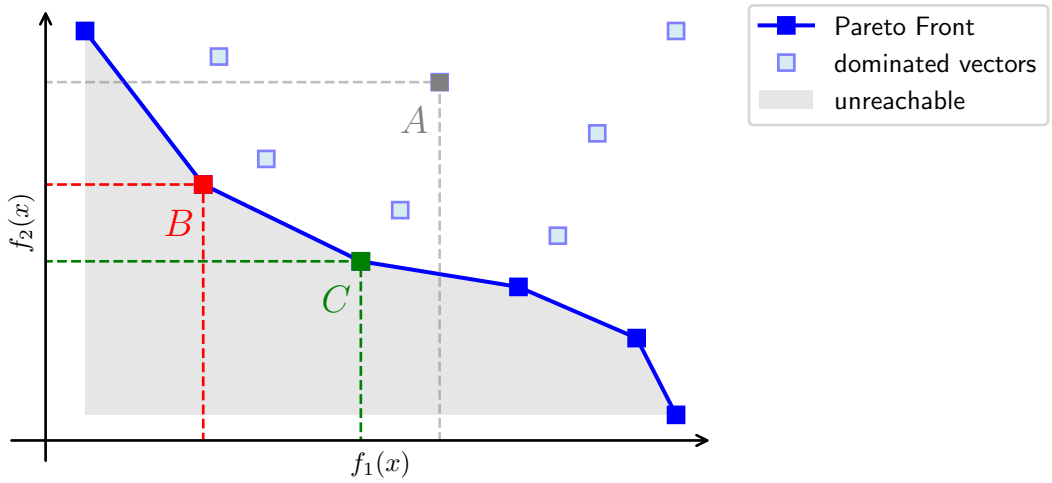


Figure 3: An example of a *Pareto front* in a two-dimensional objective space is shown where both objectives should be minimized. The blue curve represents optimal solutions. The solutions B and C are better than point A. None of the solutions in the grey area can be obtained.

Figure 3 illustrates a two-dimensional *Pareto front* where both objectives should be minimized. The x- and y-axis span the objective space defined by the objective functions f_1, f_2. The points B and C lie on the *Pareto front* displayed as a blue curve. The value of $f_1(B)$ is less than $f_1(C)$ but $f_2(C)$ is less than $f_2(B)$ so neither of them dominates the other. Point A is dominated by both points B and C, thus it does not belong to the *Pareto front*. All points marked in light blue are dominated from at least one point on the *Pareto front*. The grey area underneath the blue curve is unreachable, which means that no decision variable maps to a point in the grey area. To find the *Pareto front* for a given MOP many different methods exist. The next section tries to give a short overview of them.

2.3. Optimization methods

Since research on the *multi-objective optimization* topic started to increase in the 1970s, many methods to solve MOPs were developed [8]. Hwang and Masud [14] (1979) as well as Miettinen [19] (1999) suggested classifying methods into four classes. The role of the DM in the process of generating a solution is used to identify each class. Methods that do not need a DM or his preferences are assigned to the *no-preference* class. These methods have to find a compromise between all objectives based on some assumptions. The solution should be neutral meaning that no objective is preferred over others. The three other classes rely on direct decisions or preferences from the DM. For *a priori* methods preference information has to be defined by the DM first, afterwards the solution process tries to find a solution that satisfies all given preferences as "best" as possible. Preference information could, for example, be importance of different objectives to weight them or preferred values for objective functions. Mostly it is hard to know this information without knowledge about later generated *Pareto fronts*. Often limitations and possibilities of the problem are not known by the DM beforehand, which may cause inappropriate parameter choices. Different parameter settings lead to different *Pareto fronts* that may not be globally optimal. As an alternative *a posteriori* methods can be used to solve MOPs. These methods first generate a representation of the *Pareto front* and then the DM can select a solution based on the given information. It may be difficult for the DM to select a solution if the objective space is high dimensional. Such high dimensional spaces are hard to visualize so they are usually restricted to two dimensions to display the trade-offs between all objectives. Then a large amount of information has to be analysed by the DM to select a solution. In addition, finding the representation of the *Pareto front* can be computationally expensive or the optimal *Pareto front* cannot be found at all. The fourth class of methods is called *interactive* methods. *Interactive* methods form an iterative algorithm that is called repeatedly to find an optimal solution. After each iteration, preference information from the DM is considered for the next iteration. Some information is therefore given to the DM after each iteration, for example the actual *Pareto front*. The DM can adjust his preferences after each iteration when given new information. Methods of each class have different strengths and weaknesses according to the MOP. This classification is not complete as methods can be assigned to more than one or even no class depending on interpretation. Cohon [5] (1985) and Rosenthal [20] (1985), for example, suggest a different classification for *multi-objective optimization* methods. As mentioned in the beginning some methods combine all objectives to create a *single-objective optimization* problem.

For instance *Scalarization* combines all objectives into a single objective function. The MOP is reduced to a single-objective optimization problem as stated below:

$$\begin{aligned} &\min_x \sum_{m=1}^{M} w_m \cdot f_m(x) \\ &\text{with } x \in \Omega \end{aligned} \tag{2}$$

All weights have to be greater than zero $w_m \geq 0, m = 1, \ldots, M$. Typically they are chosen such that $\sum_{m=1}^{M} w_m = 1$ holds true. It can be proven that the solution of

Equation 2 is *Pareto optimal* if $w_m > 0$ for all $m = 1, \ldots, M$ holds true or the solution is unique [19]. *Scalarization* can be considered either as *a priori* method if the weights are specified by the DM or as *a posteriori* method. When used as *a posteriori* method the weights are varied to generate different *Pareto optimal* solutions. Afterwards the DM selects an appropriate solution. One problem is that not all *Pareto optimal* solutions can be found with this method. Censor [4] (1977) presented some conditions under which the whole *Pareto front* can be found. One of them is that the problem has to be convex. This means all objective and constraint functions as well as the decision space have to be convex. For a definition of convex problems see [3]. Another *a priori* method is the *ε-Constraint method*. This method selects one objective function and converts the other ones into constraints. The resulting optimization problem is shown below:

$$\begin{aligned} &\text{minimize } f_i(x) \\ &\text{subject to } f_m(x) \leq \varepsilon_m, \forall m = 1, \ldots, M, m \neq i \\ &\text{with } x \in \Omega \end{aligned} \tag{3}$$

For the unselected objectives the ε_m are the upper boundaries. They have to be given by the DM before solving the optimization problem, thus this method belongs to the *a priori* class. Just as for the *Scalarization* method it can be proven that the solution of the *ε-Constraint method* is *Pareto optimal*. This is only the case if the solution $x^* \in \Omega$ solves Equation 3 for every $i = 1, \ldots, M$ with $\varepsilon_m = f_m(x^*)$ for $m = 1, \ldots, M, m \neq i$ or the solution is unique. Ensuring that a solution is unique is generally not easy. Nevertheless, solving M different problems to confirm *Pareto optimality* is computationally expensive. In contrast to *Scalarization* the *ε-Constraint method* does not need a convex problem to find *Pareto optimal* solutions. Defining upper bounds for objective functions is possibly easier for the DM rather than defining weights. However, if some upper boundaries are too small, or just slightly too small, an optimal solution lying on the infeasible side, even if close to the boundary, is never found. Different *Pareto optimal* solutions can be obtained if the boundaries are chosen in a smart way. Thus, this method can also be classified as *a posteriori* method. An example for *no-preference* methods is the *Method of Global Criterion* (Yu [24] (1973), Zelany [25] (1974)). This method does not require any input from the DM. It minimizes the distance of a feasible solution to some reference point. A good choice as reference point is the so-called *ideal objective vector* z^*. The vector $z^* = (z_1^*, \ldots, z_m^*)^\top$ can be obtained by minimizing each objective individually and combining the results into one single vector $z_m^* = \min f_m(x), x \in \Omega$. The distance can be measured with the L_p- or L_∞-metric. As shown by Miettinen [19] the choice of the distance metric influences the obtained solution. The following equation defines the problem that needs to be solved to get a solution:

$$\begin{aligned} &\text{minimize } \left(\sum_{m=1}^{M} |f_m(x) - z_m^*|^p \right)^{\frac{1}{p}} \\ &\text{with } x \in \Omega \end{aligned} \tag{4}$$

It can be proven that a solution of Equation 4 is a *Pareto optimal* solution. In order

to obtain reasonable results all objectives should have the same order of magnitude. However, if they do not have the same order of magnitude they should be scaled. *Interactive methods* require the DM to update his preferences at each iteration. In general, an *interactive method* consists of the following steps:

(1) Initialize (e.g. compute *ideal objective vector*)

(2) Create a *Pareto optimal* starting point (given by DM or some *no-preference* method)

(3) Update preferences from DM (e.g. number of new solutions to be computed)

(4) Compute new *Pareto optimal* solution(s)

(5) DM may select "best" solution if more than one is given

(6) Abort if DM is satisfied with the obtained solution otherwise return to (3)

The advantage of this approach over others is that the DM does not need any global preference. During the *interactive method* preferences can be updated and only solutions that are of interest are shown to the DM. A downside of this is that the DM has to give input for every iteration. Hwang and Masud [14] as well as Miettinen [19] describe some interactive methods for MOPs. The *Scalarization* and *ε-Constraint method* mentioned above belong to the *a posteriori* class if they are slightly modified. A different approach to solve MOPs are *evolutionary algorithms* (EAs). Based on *Darwin's "Survival-of-the-fittest"* evolutionary theory and named after it, EAs start with an initial population which is then evolved over multiple generations. Between each generation, new members are generated by crossing and mutating members from the existing population. Afterwards the fittest individuals from the population are selected and form the new generation. After a certain number of generations, the algorithm terminates, and the current population is the result. When EAs are applied to MOPs, each individual represents one vector from the decision space. Their fitness corresponds to the objective values associated with the decision vector. *Pareto optimality* is used to compare the fitness of different individuals. The population from the final generation forms the *Pareto archive* with their corresponding fitness values belonging to the *Pareto front*. EAs belong to the *a posteriori* class. They have some advantages over classical optimization methods (Goldberg [13] (1989)). Gradient information is usually not necessary for EAs. A direct search procedure is used to obtain a solution so that EAs can be applied to a wider range of optimization problems. For specific structured problems, their performance should not compete with linear or convex programming methods as they are especially designed for those cases. Unlike most optimization methods, EAs use more than one solution per iteration (*population* approach). This has higher computational costs but allows finding more than one optimal solution, which is useful in case of *multi-objective optimization*. EAs also use stochastic operators allowing them to overcome multiple optima and other complexities better. A common used EA is the *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) developed by Deb [9] (2002).

3. Surrogate models

In the field of engineering as in many other complex models are built to represent processes from the real world. These models can be used for design optimization. In terms of *multi-objective optimization*, they represent objective functions mapping from parameter (decision variables) space to the objective space. Solving MOPs requires a high number of objective function evaluations. Such evaluations could take minutes, several hours or even days depending on the complexity and accuracy of the underlying model. This makes it nearly impossible to run *multi-objective optimization* algorithms like EAs with these models in a reasonable timeframe. *Surrogate models* are an approach to overcome the high computational costs. The idea is to create simple(r) models as surrogate for the complex model, for example based on polynomials or exponential functions, so that evaluations have less computational costs. Other terms for *surrogate models* are *metamodels*, *approximation models* or *emulators*.

Consider a complex model denoted by $F(x)$, which outputs u should be minimized with respect to some of its parameters. The parameters that are not optimized remain constant and can be neglected for the MOP. Remaining parameters are stored in a vector $x \in \Omega$ with Ω as feasible region following the notation from Equation 1. To build a *surrogate model* for $F(x)$ some input and its corresponding output data is needed. Therefore, the original complex model has to be evaluated, which is computationally expensive. It is important to keep the number of evaluations low. So-called *Design of experiment* (DoE) methods are used to extract as much information as possible with the least amount of function evaluations. All design points resulting from a DoE method combined into one set are called *sampling*. An overview of different classical and modern DoEs has been given in [12]. Because the modern *Latin Hypercube sampling* (LHS) method is later used as initial *sampling* for the *surrogate models* it will be shortly introduced.

To create an LHS with 10 design points for instance the decision space is divided into a grid with 10 elements in each dimension. Figure 4 shows an example for an LHS in two dimensions. The 10×10 grid is first filled diagonally. Random coordinates bound by a diagonal adjacent cell are chosen, determining the position of the design point as shown on the left. This ensures that exactly one design point lies in each row and column of the grid. For each dimension, the coordinates of all design points are shuffled so that the resulting *sampling* is nearly equally distributed visualized on the right of Figure 4. Afterwards the *sampling* is used to evaluate the original model. Design points and corresponding outputs from the original model are used as training data for the *surrogate models*. Training data is used to fit so-called *hyperparameters* of the *surrogate model*. More training points lead to more accurate models but they also require higher computational effort. Once the model is trained, evaluations become inexpensive. *Surrogate models* allow the prediction of values in regions where the original model was not evaluated. This means that instead of applying optimization methods to the original model, the *surrogate models* can be used.

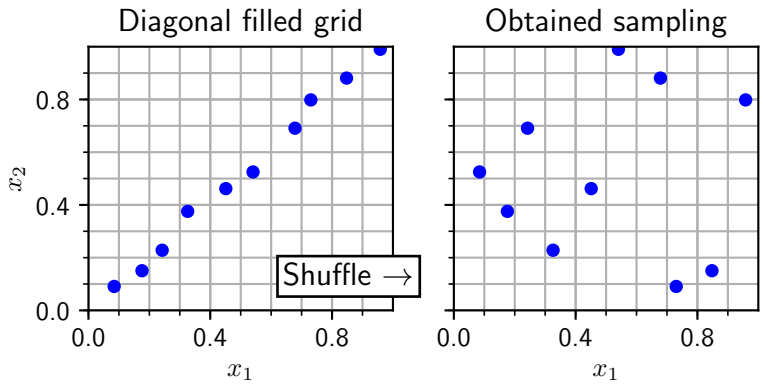


Figure 4: On the right a Latin Hypercube sampling (LHS) for two dimensions with 10 points is shown. To obtain the sampling 10 points are added to the diagonal of a regular grid. Then the coordinates of every dimension except the first one are shuffled.

3.1. Polynomial based models

Polynomials can be used to build simple *surrogate models*. The *surrogate model*, denoted by $\hat{f}$, can be written as follows:

$$\hat{f}(x, m, a) = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m = \sum_{i=0}^{m} a_i x^i \tag{5}$$

The vector $a = (a_0, a_1, \ldots, a_m)^\top$ contains all polynomial coefficients with m as the maximum polynomial degree. Note that Equation 5 is only valid for one-dimensional data. To take more dimensions into account mixed terms (e.g. $x_1x_2, x_1x_2^2, x_1^2x_2, x_1^2x_2^2$) need to be added to the sum. More than one dimension in the output domain is not necessary because a *surrogate model* can be built for each objective individually.

Assume the polynomial degree is fixed and a training dataset $P_{train} = \{(x, y)|x, y \in \mathbb{R}\}$ with n points is given. To fit the polynomial coefficients a to the dataset the following linear system is constructed:

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^m \\ 1 & x_2 & x_2^2 & \ldots & x_2^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^m \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

A short form of the system can be written as follows:

$$\Phi \cdot a = y \tag{6}$$

Generally, this system has no solution. Nevertheless a "best" fitting vector a can be found. Solving Equation 6 for polynomial coefficients a is done by *least-squares*

analysis. This means to solve the following minimization problem or more precisely find a^* that minimizes the residuals.

$$\min_{a \in \mathbb{R}^n} \|\Phi a - y\|$$

Using theory behind *least-squares analysis* the approximation $\hat{a}$ for a can be computed as follows:

$$\hat{a} = (\Phi^\top \Phi)^{-1} \Phi y \tag{7}$$

In practice different formulations of Equation 7 are used to numerically solve for $\hat{a}$ [2]. Therefore more training points than the highest polynomial degree are needed $n > m$. The method to find such polynomial based *surrogate models* is called *least square polynomial regression*. An assumption to obtain a solution $\hat{a}$ was that the polynomial degree m is known a priori. Usually the best polynomial degree is not known. If the model that should be approximated does not contain polynomials and instead, for example, is a combination of sine and cosine functions no optimal polynomial degree exists.

Figure 5 shows a comparison between different polynomial degrees $m = 1, 2, 3, 4$. The blue points are used as training points for the construction of the *surrogate models*. With higher polynomial degrees, the curve fits the training points better. However,

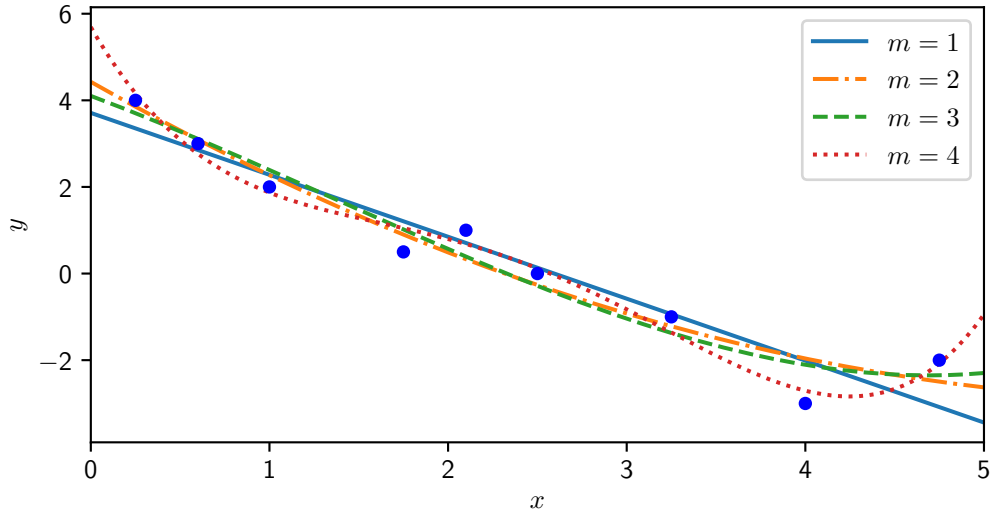


Figure 5: Four polynomial based *surrogate models* with different maximal polynomial degrees are shown. The training points are marked in blue. Higher polynomial degrees better fit the training points but rapidly increase at the boundaries.

increasing the degree to large values results in *over-fitting* the data. Figure 6 illustrates this behaviour. The same training points, marked in blue, fitted with a polynomial degree of $m = 8$ result in values in the interval $[-20, -5]$ while the initial training points lie inside the interval $[-3, 5]$. With a nearly linear trend observed for the training

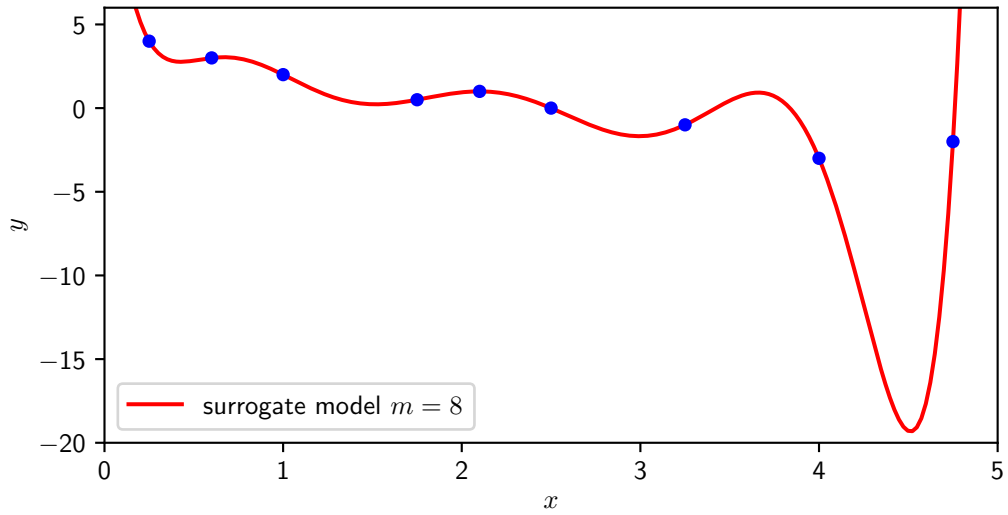


Figure 6: The red curve represents a *surrogate model* with a maximum polynomial degree of $m = 8$. The same training points as before are fitted perfectly. However, on the right side the data is *over-fitted* resulting in wrong predictions for values in this interval.

points, the *surrogate model* obviously gives a wrong representation of the original model. It becomes even worse when the initial training points contained some noise, for example due to measurement errors. To find an optimal polynomial degree, more information is needed. New design points are therefore added that do not originally belong to the training sampling. Usually more design points are a downside because model evaluation is expensive. To overcome this the training data set could be divided into two separate sets - one acting as training data, the other one as test data. The test data is used to compute a *loss function* measuring the distance between the *surrogate model* and the *sampling*. *Over-fitting* the data points is now measured by the *loss function* so that a more accurate *surrogate model* can be found. In most of the cases a polynomial degree around $m = 2$ is sufficient enough to get a reasonable *surrogate model*. However, as seen in the example in Figure 5, the *surrogate model* does not fit the training points perfectly. There is still some offset between the curve and the training points with lower polynomial degrees. As this might be helpful for measurements with certain noise, it distorts the *surrogate model* if the training data is supposed to fit exactly. Normally the simulation models provide exact data points. To build *surrogate models* that fit "best" a different approach than *least square polynomial regression* should be used.

3.2. Kriging - Gaussian process regression

Kriging modelling is named after D. G. Krige who was a South African mining engineer. He developed an interpolation technique to produce underground maps for mining

purpose. Matheron [18] formally developed it and Jones [17] as well as Sacks [22] [21] made it well known in fields of modelling and optimization. *Kriging* uses stochastic processes to approximate the original model. It combines a global model $g(x)$ also called *trend function* and a *Gaussian process* error model $\epsilon(x)$ as follows:

$$\hat{f}(x) = g(x) + \epsilon(x)$$

In "Dakota Version 6.0 Theory Manual" [1] three main steps are suggested when building a *Kriging surrogate model*.

- Choice of a *trend function*
- Choice of a correlation (*kernel*) function
- Estimation of correlation parameters

With different choices of a *trend function Kriging* can be divided into the most common cases. Choosing a constant value as *trend function* is known as *simple Kriging*. Using *least square polynomial regression* to receive a polynomial *trend function* is called *universal Kriging*. Instead of a known constant, an unknown constant can be used as the *trend function* (*ordinary Kriging*). The preferred method depends on the model that is to be approximated. The *Gaussian process* error model $\epsilon(x)$ is a Gaussian random function with zero mean and non-zero covariance. The following equation shows how to calculate the covariance of the error $\epsilon(x)$ between two arbitrary points $x^{(i)}, x^{(j)}$:

$$\mathrm{Cov}(\epsilon(x^{(i)}), \epsilon(x^{(j)})) = \sigma^2 r(x^{(i)}, x^{(j)}) \tag{8}$$

Here σ denotes the standard deviation of the Gaussian random function and $r(x^{(i)}, x^{(j)})$ describes the correlation between both points. As correlation or *kernel* function the *squared exponential* (Equation 9) is often used [15].

$$r(x^{(i)}, x^{(j)}) = \exp\left[-\sum_{k=1}^{n} \theta_k (x_k^{(i)} - x_k^{(j)})^2\right] \tag{9}$$

For each dimension k an unknown correlation factor θ_k has to be calculated. These factors are estimated during the training phase of the *Kriging* model with the maximum likelihood method [16]. Additionally, because stochastic processes are used, *Kriging* models can compute variance and confidence intervals at predicted points. For a given input, the output of the real model lies inside the confidence interval with a certain probability.

Figure 7 illustrates the initial state on the left and the trained model on the right side using *simple Kriging* with the same training points as used in section 3.1. The *trend function* is chosen as constant zero $g(x) = 0$. As this example is one-dimensional, only one correlation factor θ needs to be determined. In the initial state θ equals zero. The slightly thicker black line in Figure 7 represents the *surrogate model* $\hat{f}(x)$. It is calculated as the mean of the random Gaussian functions with the given covariance

from Equation 8. Initially the model has a mean value of zero resulting from $\theta = 0$. The coloured curves represent some sample Gaussian random functions. The grey area shows the 95% confidence interval in both plots. This means that the value from the real model lies in this area with a probability of 95%. Within the training process the correlation factor was found as $\theta = -0.85$. At the right of Figure 7 the trained *surrogate model* is shown with $\theta = -0.85$. The training points are marked with a blue dot. The model fits all training points and results in a smooth curve. In contrast to the polynomial based approach, the training points are not *over-fitted*. Even when looking at the confidence intervals values less than -5 are improbable.

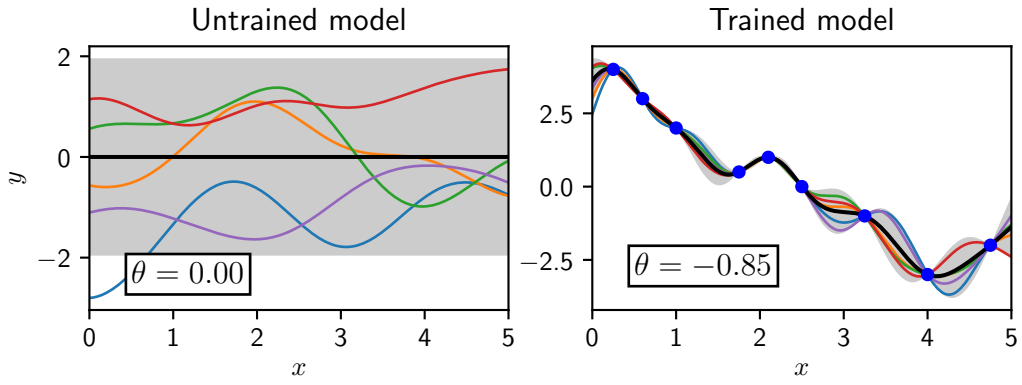


Figure 7: On the left side the initial state of a *Kriging surrogate model* is shown. The *surrogate model* is illustrated as black line. The grey area visualizes the confidence regions and the coloured curves represent some random Gaussian functions. On the right side the trained model using the same training points as before is shown. The correlation factor θ has changed.

Kriging surrogate models have the advantage that they fit all training points perfectly without that much *over-fitting* effects and provide additional information such as variance for predicted points. Every *surrogate model* gets more accurate when increasing the number of design points provided to build the *surrogate model*. However, evaluations of the original model have a high computationally effort. To use a minimum amount of evaluations on the original model and still get the desired global minimum the choice of evaluation points is very important. The next section provides an overview for different *sampling* strategies.

4. Adaptive sampling strategies

To avoid many evaluations and still build an accurate *surrogate model* used for optimization three strategies will be introduced in this section. Instead of using a static predefined set of design points, it is more efficient to adapt the design points. Each strategy starts with an LHS to create an initial *surrogate model*. The number of sample

points for the initial LHS should be small. After the optimization finished on the *surrogate model*, the additionally obtained information is used to decide where to evaluate the original model next. This iterative process is repeated until the solution converged, or some maximum number of iterations is reached. Additional information that can be used to decide where to adapt the *sampling* are current optimal solutions, variances or confidence intervals and the already used design points. With this additional information, the following three strategies can be derived:

- Adapt *sampling* at current *Pareto optimal* solutions
- Adapt *sampling* where confidence intervals are maximal
- Adapt *sampling* at current *Pareto optimal* solutions and where confidence intervals are maximal

4.1. Strategy based on Pareto optimal solutions

A new set of design points can be obtained from the current *Pareto optimal* solutions. After each optimization, the best values obtained for the objectives are stored in the *Pareto front*. The corresponding values for the decision variables are stored in the *Pareto archive*. Several points n_{add} from the *Pareto archive* are selected to create a new *sampling* and improve the *surrogate model*. This selection could be random. However, for multiple objectives the *Pareto archive* for example can be divided into two clearly distinguishable sets. If one set only consists of a small number of points and the other one has a larger number of points a random selection may contain no points from the smaller set and thus be missing information. To avoid this scenario the selection of points is made in a way such that the points are widely spread. First a random point from the whole set is selected. Next, the distance of all other points to the selected point x_i is calculated using the Euclidean norm. The point with the largest distance is added to the set. With more than one point already selected, new points x_{new} are added if the minimal distance to an already selected point x_i is maximal. If the set does not contain n_{add} points new points are selected.

$$x_{new} = \arg\max_{x} \left(\min_{x_i} \|x - x_i\|_2 \right)$$

This results in a set with maximal distant points. It depends on the starting point whether the globally maximal distant subset is found but even if not, the result is still better than a random selection.

Choosing points at the current optimal solution is known under the term *exploitation* [7]. The *surrogate model* gets more accurate in regions where the optimization found some minima. Nevertheless, this approach may get stuck in local minima. An example is given by Figure 8. The unknown function in this one-dimensional example that has to be minimized is visualized as a blue curve. A given *sampling* shown as red points lead to the prediction marked as a red dotted line. The design points to

evaluate the blue function are not equally distributed. This can happen especially for high dimensional MOPs after some iterations with adaptive *sampling* strategies. However the global minimum of the true function is somewhere around $x = 8$. Because the minimum value of the prediction is around $x = 2$ a new sample point based on the *exploitation* strategy is added at $x = 2.2$. Further iterations only give a more accurate prediction of the local minima around $x = 2$. Finding the global minima needs some more globally focused search approaches.

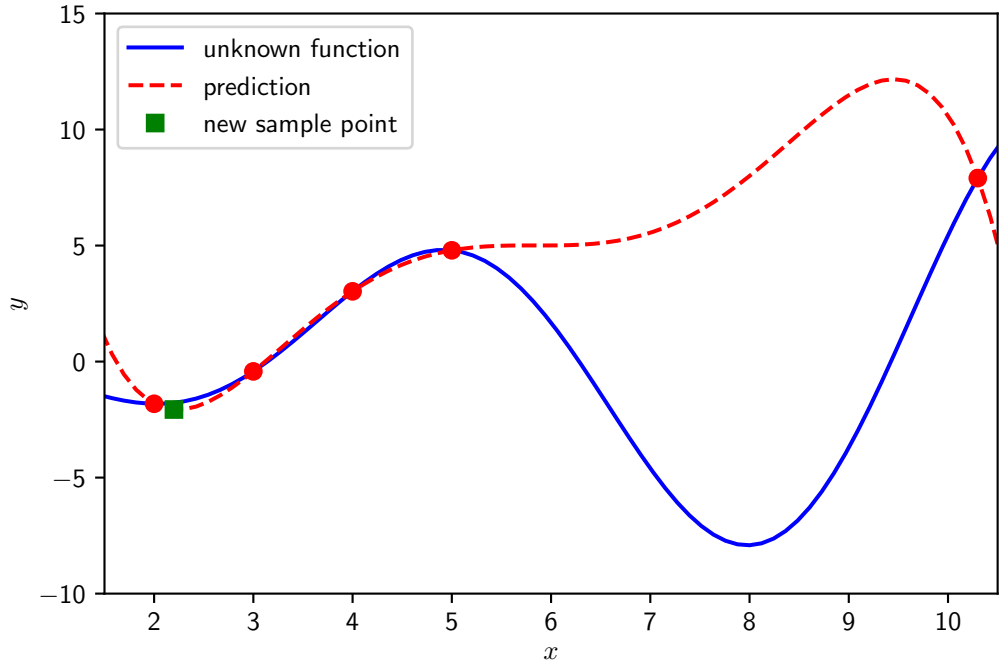


Figure 8: An adaptive *sampling* strategy based on *exploitation* is illustrated. The red dotted line shows the model of the unknown blue function based on the red training points. The green point is added to the existing *sampling* to be evaluated next. However, the global minimum will never be found because no points will be added in this area.

4.2. Strategy based on maximum confidence intervals

Instead of choosing new design points at optimal solutions, the confidence intervals given by the *surrogate model* can be used to decide where new design points are generated. When choosing new points at the current optimal solution, the algorithm might only find new local minima in the adjacency of the current optimal solution. To overcome this, points can be selected based on their uncertainty. Already evaluated points have no uncertainty. Between those points, the *surrogate model* has some uncertainty, which can be measured by the confidence intervals. Larger confidence intervals mean higher uncertainty about the predicted value. To increase the overall accuracy of the

surrogate model additional n_{add} points are selected where the confidence intervals are maximal. The p confidence interval can be computed from the variance σ^2 where p denotes the probability that the value evaluated from the original model lies inside the confidence region. This region has the following boundaries:

$$\begin{aligned} \text{lower bound} &: \mu - Z_p\sigma^2 \\ \text{upper bound} &: \mu + Z_p\sigma^2 \end{aligned}$$

For these boundaries Z_p is calculated from the normal distribution with zero as the mean and a standard deviation of one. The area enclosed by the normal distribution ($\mathcal{N}(0,1)$) for the interval $[-Z_p, Z_p]$ equals p. This is used to calculate Z_p for different probabilities p. For a 95% confidence interval $Z_{0.95}$ nearly equals 1.96.

Choosing points with a maximal confidence interval reduces the uncertainty of the *surrogate model*. The optimization searches in a more global region compared to the first approach. This strategy is known as *exploration* [7]. However, some evaluations are unnecessary in terms of finding the global minima because regions with values much larger than the current optimal value also get new design points. Applying this strategy to the example from the previous section (Figure 8) results in finding the global minima because a new sample point around $x = 8$ is added. Considering a different example shows the disadvantage of this approach. The blue curve in Figure 9 represents the original model and the red dots display the design points used to get the prediction shown as red dotted line. The uncertainty is shown as a grey shade where a larger area corresponds to a larger 95% confidence interval. A new sample point is added around $x = 1$. Although the global minimum is almost found near $x = -8$ the new sample points are added in the region $x \in [-5, 3]$ because the uncertainty is higher than close to the minima. This leads to unnecessary evaluations of the original model while the found minima does not change.

4.3. Mixed strategy

The *mixed* strategy tries to combine both advantages from the previous approaches. Instead of focusing either on *exploration* or on *exploitation*, confidence intervals and the actual *Pareto optimal* solutions are considered to choose new design points. Therefore, half of the new design points that should be added are selected from the *Pareto optimal* solutions. The other half is chosen based on their confidence interval lengths. The selection process for both halves is the same as described above. Although this approach uses a fixed ratio between points selected from the *Pareto optimal* solutions and maximum confidence intervals it should increase the overall convergence with only a minimal number of unnecessary evaluations.

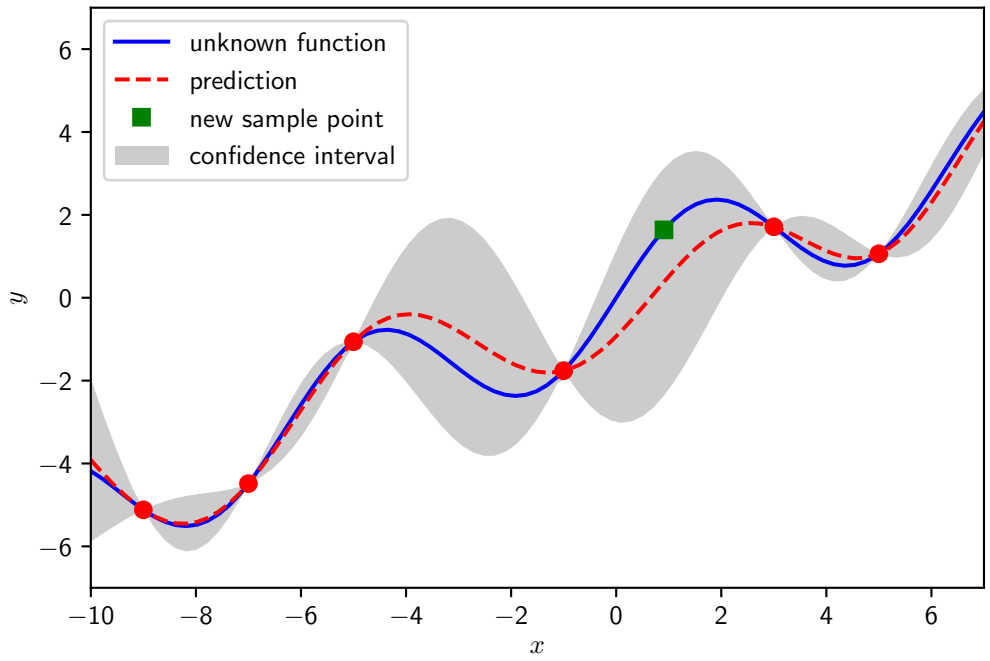


Figure 9: An adaptive *sampling* strategy based on *exploration* is illustrated. The red dotted line shows the model of the unknown blue function based on the red training points. The grey shaded area represents the confidence region. A new design marked as green square is added to the existing *sampling* to be evaluated next. However, the global minimum has already been found on the left so more evaluations are unnecessary.

5. Algorithm

Based on the previous sections an algorithm was developed. Instead of just applying EAs like NSGA-II the algorithm consists of a combined approach using *surrogate models* and adaptive *sampling* strategies as well as EAs to solve MOPs. Figure 10 illustrates the procedure of the algorithm. First the original model is evaluated, these results are used to build *surrogate models*, which then are solved to receive optimal solutions.

During the *Initialize* phase the problem dependent options, for example boundaries of decision variables, are set. Additionally, the initial population for the EA is created. Next is the *Create DoE* phase. During the first iteration of the algorithm an initial DoE is created. For repeating iterations, new design points are added during this phase to the already existing DoE. The design points are used to evaluate the original model. This is done at the *Evaluate original model* phase. Only points that had not already been evaluated are considered. With the results and the *sampling*, *surrogate models* are built for every objective in the *Create surrogate models* phase. If the *surrogate models* already exist from previous iterations, they are updated with the new information from

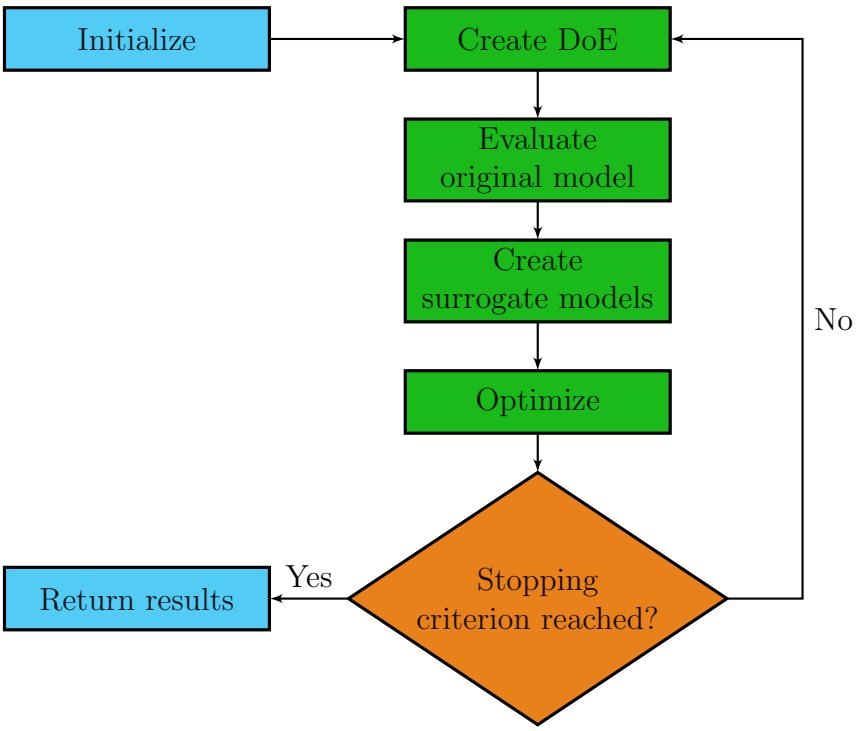


Figure 10: This flowchart illustrates the procedure of the algorithm. The blue phases on the left are only performed once. The main loop consists of the green and orange marked phases on the rights side.

the *Evaluate original model* phase. During the *Optimize* phase an EA solves the MOP using the current *surrogate models*. The high number of model evaluations does not significantly influence the performance because the *surrogate models* are evaluated, instead of the original model. When the stopping criterion is reached, for example a certain number of iterations, the algorithm terminates. In the *Return results* phase the last obtained *Pareto front* and the corresponding *Pareto archive* are returned. If the stopping criterion is not fulfilled the algorithm continues from the *Create DoE* phase.

The algorithm structure allows different approaches for the phases. During the *Initialize* phase and the first iteration of the *Create DoE* phase the required DoE can be built with different methods. For example, the design points could be chosen on a simple grid or an approach like LHS could be used to generate a DoE. For the *Create surrogate models* phase two different approaches to build *surrogate models* were discussed in section 3. Nevertheless, the algorithm is not limited to these cases. For example, *Gaussian processes* can be used with other *kernels* to achieve a better approximation of the original model if for instance the modelled process happens periodically. In the *Optimize* phase any *a-posteriori* method can be used as an optimization method applied on the chosen *surrogate models*. To derive new design points in the *Create DoE* phase out of the optimization results, three strategies were introduced in the previous section 4. Other approaches can work here too.

For this thesis the following settings are used for the algorithm. To create an initial

DoE as well as an initial *population* for the later EA an LHS is utilized. *Gaussian processes* in combination with neural networks called *Deep Gaussian Covariance Network*(DGCN), to reduce the training time, are used as *surrogate models*. The DGCN framework was developed by Cremanns and Roos (2017). More information about the combination of *Gaussian processes* and neural networks can be found in their corresponding paper [6]. The underlying method is similar to the *Kriging* method described earlier. For the optimization an EA is used because it provides both a *Pareto front* and a corresponding *archive* as the result of a single run. The NSGA-II from Deb et al. [9] is chosen as it performs well for a wide range of problems. The algorithm is implemented with Python. Python is an interpreted high-level programming language with a large standard library. Many modules allow to extend the functionality further. The entire framework of the algorithm is implemented as one class. Each phase has its own method, which gets called from a main method following the flow chart from Figure 10. The LHS for the initial DoE is implemented using some methods from NumPy[1], a module for scientific computation allowing the use of matrices and some higher-level mathematical functions. To evaluate the original model the simulations are performed on a high-performance cluster (HPC). This allows the parallel execution of the required simulations, which can save a significant amount of runtime when one evaluation takes longer than one hour. Therefore, each evaluation has to be send separately to the HPC queue. The status of each simulation has to be tracked to ensure a certain number of simulations to run parallel. For this purpose, the implemented algorithm provides a framework that handles the parallel evaluation of given design points. The DGCN framework was provided as a Python module that allows the usage of it within the implementation. To solve a MOP with the built *surrogate models* the implementation of NSGA-II from the inspyred[2] module was used. When the stopping criterion, a maximum number of iterations, is not reached, one of the three strategies described can be used to adapt the *sampling*. Their implementation follows the description from section 4 and one of them can be chosen at the beginning. The results of each iteration can be stored on disk to later comprehend the steps of the algorithm, find some errors during the early development stage or to continue the algorithm from the last saved iteration. To test the performance of the algorithm different benchmark problems are solved, and the results are compared against an optimization using only the NSGA-II algorithm.

5.1. Benchmark problems

Six MOPs are chosen to test the performance of the algorithm. This includes two simple problems with two objectives and no more than three inputs as well as four more complex problems selected from the ZDT family. Each problem is described below including the mathematical function definition, boundaries of the decision variables and the optimal solutions.

[1] `http://www.numpy.org/`, Online access: 29.03.2018
[2] `https://github.com/aarongarrett/inspyred`, Online access: 03.04.2018

The two simple problems are proposed by Schaffer (SCH1) as well as by Fonseca and Fleming (FON). SCH1 [23] has one decision variable and two objective functions. The problem is formulated as follows:

$$\begin{aligned} &\min_x \ f_1, f_2 \\ &f_1(x) = x^2 \\ &f_2(x) = (x-2)^2 \end{aligned}$$

The decision variable x is not restricted by the problem itself. A larger decision space increases the difficulty to find the optimal solution. For the performance tests the decision space is chosen as $\Omega = [-1000, 1000]$. Obviously, the optimal solution is $x \in [0, 2]$ which results in a convex *Pareto front*.

FON proposed by Fonseca and Fleming [10] has two objective functions and two or more decision variables. For this benchmark problem three decision variables $n = 3$ are used as input. The general formulation of this problem is as follows:

$$\begin{aligned} &\min_x \ f_1, f_2 \\ &f_1(x) = 1 - \exp\left[-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right] \\ &f_2(x) = 1 - \exp\left[-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right] \end{aligned}$$

For each dimension x_i can be chosen within the interval $[-4, 4]$. The optimal solution for three dimensions is $x_i \in \left[-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right], i = 1, 2, 3$ with $x_1 = x_2 = x_3$. Opposite to the first benchmark problem FON has a non-convex optimal *Pareto front*.

Zitzler et al. proposed a method to create more complex test functions. Six test problems were constructed, which are known as ZDT1 to ZDT6 [26]. The first four are used for the performance tests. These problems consist of two objective functions and a variable number of decision variables. All four test problems are constructed using the same framework.

$$\begin{aligned} &\min_x \ f_1, f_2 \\ &f_1(x) \text{ arbitrary function} \\ &f_2(x) = g(x) \cdot h(x) \end{aligned}$$

In this framework f_1 controls the difficulty along the *Pareto front* and $g(x)$ controls the difficulty lateral to the *Pareto front*. The function $h(x)$ defines the properties of the *Pareto front* such as convexity or discontinuity.

The ZDT1 problem has a convex *Pareto front*. It is used with $n = 30$ decision variables and the functions f_1, f_2 and g are defined as follows:

$$f_1(x) = x_1$$

$$f_2(x) = g(x) \cdot \left[1 - \sqrt{\frac{x_1}{g(x)}}\right]$$
$$g(x) = 1 + \frac{9 \cdot \sum_{i=2}^{n} x_i}{n-1}$$

For all dimensions x_i can be chosen from the interval $[0, 1]$. An optimal solution requires $g(x) = 1$. Therefore the equation $x_i = 0, i = 2, 3, \ldots, n$ has to hold while x_1 can be any value in the interval $[0, 1]$.

The ZDT2 problem is constructed with the same definition for the functions f_1 and g. It is also constructed with $n = 30$ dimensions for the decision space. All decision variables can be chosen within the interval $[0, 1]$.

$$f_1(x) = x_1$$
$$f_2(x) = g(x) \cdot \left[1 - \left(\frac{x_1}{g(x)}\right)^2\right]$$
$$g(x) = 1 + \frac{9 \cdot \sum_{i=2}^{n} x_i}{n-1}$$

The resulting optimal *Pareto front* has a non-convex shape. Because the function $g(x)$ is defined the same as in ZDT1 the optimal solutions are also $x_1 \in [0, 1]$, $x_i = 0, i = 2, 3, \ldots, n$.

The ZDT3 problem has a more complex *Pareto front* compared to the previous ones. Like in ZDT1 and ZDT2, 30 design variables chosen within the interval $[0, 1]$ are used. However, the resulting *Pareto front* is discontinuous and convex.

$$f_1(x) = x_1$$
$$f_2(x) = g(x) \cdot \left[1 - \sqrt{\frac{x_1}{g(x)}} - \frac{x_1}{g(x)} \sin(10\pi x_1)\right]$$
$$g(x) = 1 + \frac{9 \cdot \sum_{i=2}^{n} x_i}{n-1}$$

For the decision variables the optimal solutions are the same as for the first two ZDT problems $x_1 \in [0, 1]$, $x_i = 0, i = 2, 3, \ldots, n$.

The ZDT4 problem has a continuous convex *Pareto front*. The difficulty is to find the global optimal solution because it has 21^9 local *Pareto fronts* [9] of which only one is the global optimum. Instead of 30 only 10 design variables are used for this problem.

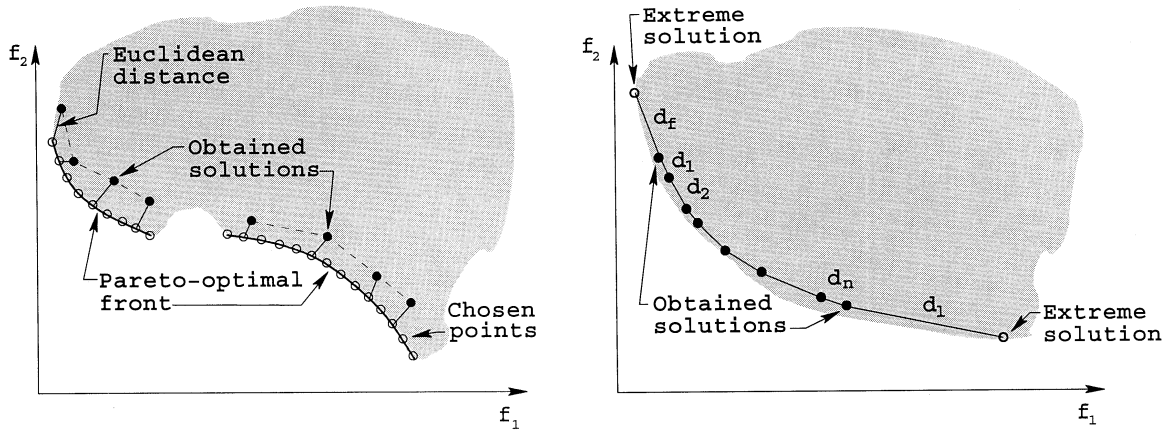
$$f_1(x) = x_1$$
$$f_2(x) = g(x) \cdot \left[1 - \sqrt{\frac{x_1}{g(x)}}\right]$$
$$g(x) = 1 + 10(n-1) + \sum_{i=2}^{n} \left[x_i^2 - 10\cos(4\pi x_i)\right]$$

Problem	No. of design variables	Boundaries	No. of objectives	Pareto front
SCH1	1	$x \in [-1000, 1000]$	2	convex, large decision space
FON	3	$x_i \in [-4, 4], i = 1, 2, 3$	2	non-convex
ZDT1	30	$x_i \in [0, 1], i = 1, 2, \ldots, 30$	2	convex
ZDT2	30	$x_i \in [0, 1], i = 1, 2, \ldots, 30$	2	non-convex
ZDT3	30	$x_i \in [0, 1], i = 1, 2, \ldots, 30$	2	convex, discontinuous
ZDT4	10	$x_1 \in [0, 1]$, $x_i \in [-5, 5], i = 2, 3, \ldots, 10$	2	convex, many local optima

Table 1: An overview of all benchmark problems used to test the different strategies of the algorithm. The last column lists some properties of the optimal *Pareto front* in the two-dimensional objective space as well as some difficulties of the benchmark problem.

Furthermore, the decision space is different compared to decision spaces of ZDT1 - ZDT3. The decision variable in the first dimension x_1 can be chosen from the interval $[0, 1]$. All other decision variables $x_i, i = 2, 3, \ldots, n$ lie in the larger interval $[-5, 5]$. It follows from the function definitions that the optimal solutions are still $x_1 \in [0, 1]$, $x_i = 0, i = 2, 3, \ldots, n$ where $g(x) = 1$ holds true [26].

Table 1 summarizes the properties of all benchmark problems that are used to measure the performance of the algorithm against an optimization using only NSGA-II. Measuring the performance by the runtime of the two approaches has a huge disadvantage. Not only does it depend on the hardware components of the executing system, but also the algorithm will have a much longer runtime for the benchmark problems than a pure optimization using NSGA-II. Because the benchmark problems only consist of two functions, which can be evaluated within milliseconds, the training time of the *surrogate models* results in a much longer runtime of the algorithm. The algorithm can only be effective if the training time is neglectable in comparison to the time needed to evaluate one design on the original model. Therefore, the performance has to be measured differently. All strategies are run until a certain number of original model evaluations is reached. The results obtained at the end of these optimizations are used to calculate some performance metrics. Two metrics introduced by Deb et al. [9] are used. The first metric describes how close the solution is to the optimal *Pareto front* and the second metric measures the spread of the solutions along the *Pareto front*. The first metric is called *convergence metric* and is denoted by γ. To compute the metric the optimal *Pareto front* has to be known. Then the distance of each point on the approximated *Pareto front* to the optimal front can be measured. Figure 11a shows an example of those distances.



(a) Convergence metric γ measuring the distance to the optimal *Pareto front*.

(b) Diversity metric Δ measuring the spread of the obtained solutions.

Figure 11: Visualization of the convergence and diversity metric taken from Deb et al. [9]. These metrics are used to compare the results of the different strategies for the algorithm.

$$\gamma = \frac{1}{n} \sum_{i=0}^{n} c_i \tag{10}$$

The metric is the mean over all the distances with n as number of points lying on the approximated front. The best value for the convergence metric is zero because this means all approximated points lie exactly on the optimal *Pareto front*. If the points are far away from the optimal solution γ increases. The metric has no upper boundary, so a smaller value means better convergence. However, to compute this metric for the benchmark problems the optimal front is discretized with 500 points. Hence the metric will only be zero if all approximated points lie exactly on the discrete points chosen to represent the optimal *Pareto front*. This is very unlikely so values near zero mean the approximated front matches the optimal front.

Since the aim of *multi-objective optimization* is not only to find a certain set of solutions, but also to find evenly distributed solutions, Deb et al. introduced a second metric measuring the spread of the obtained *Pareto front*. This metric is called *diversity metric* and is denoted by Δ. It takes the distance between individual points on the approximated front as well as the distance between the approximated and optimal boundary solutions into account. The *diversity metric* is computed using the following equation:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{n-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (n-1)\bar{d}} \tag{11}$$

Here, n is the number of points on the approximated *Pareto front* and d_f, d_l are the distances between the boundary solutions of the approximated and the optimal *Pareto front*. Figure 11b visualizes all distances used in Equation 11. The mean over the

Euclidean distances $d_i, i = 1, 2, \ldots, n-1$ between consecutive points on the front is denoted by $\bar{d}$. With a large variance in the distances d_i this metric takes a value greater than one. Like the *convergence metric* the best value is zero with no upper bound. Therefore all distances d_i have to be equal, which results in the sum equal zero. Additionally, the approximated boundary solutions have to match the boundary solutions of the optimal *Pareto front*, $d_f = d_l = 0$. The *diversity metric* in this form can only be computed for up to two dimensional objective spaces. In higher dimensional objective spaces, the distances d_i can be calculated, for example, by utilizing a triangularization technique. With these two metrics the next section discusses the results obtained from the algorithm using all three different adaptive *sampling* strategies and an optimization using only NSGA-II.

5.2. Simulation results

All approaches are run until 500 evaluations of the original model are reached. In case of ZDT benchmark problems the maximum number of evaluations is increased to 2500. These limits are chosen because complex simulations, even when parallelized on HPC, can take around two or three days until 2500 are evaluated, so it is reasonable not to increase the number of evaluations far beyond this maximum. For the simpler problems (SCH1, FON) the number of points is reduced to see the difference between the approaches better because otherwise they may all be converged and thus the difference is at the same order as computer accuracy. Therefore, the NSGA-II is run with a population size of 100 individuals for 4 generations with an initial population counting also 100 individuals. When solving a MOP problem from the ZDT-family the maximum number of generations is increased to 24 so that the overall number of evaluations of the original model matches 2500. The algorithm runs for 5 iterations for the SCH1 and FON problems and the number of iterations increases to 25 for the ZDT benchmark problems. One iteration is defined as the execution of all phases on the right side of Figure 10. The first iteration is like the generation of the initial population in the case of the NSGA-II. In each iteration except of the first one, which starts with an LHS of 100 design points, 100 points are added according to the strategy mentioned before. During the *Optimize* phase a NSGA-II is used. Because it only evaluates the *surrogate models*, which has a low computational effort, the NSGA-II runs for a maximum number of 100 generations with a population of 100 individuals in each generation. This results in $10,100$ *surrogate model* evaluations for each iteration, the additional 100 evaluations originate from building the initial population. The algorithm is abbreviated based on the different adaptive *sampling* strategies within figures, tables and throughout the following text. Adding new points based on the found *Pareto archive* is called PAR, adding points at maximal confidence intervals is abbreviated as CI and using the *mixed* strategy is called MIX. The reference optimization without *surrogate models* and adaptive *sampling* is referred to as NSGA2 within figures, tables and throughout the text. Table 2 summarizes the settings for each approach and benchmark problem. Because the LHS and the *surrogate models* as well as the NSGA-II involve random processes more than one optimization is performed. For each

Method	SCH1	FON	ZDT1	ZDT2	ZDT3	ZDT4
NSGA2	4 gen ×100 pop +100 init	4 gen ×100 pop +100 init	24 gen ×100 pop +100 init	24 gen ×100 pop +100 init	24 gen ×100 pop +100 init	24 gen ×100 pop +100 init
PAR / CI / MIX	5 iter ×100 pts	5 iter ×100 pts	25 iter ×100 pts	25 iter ×100 pts	25 iter ×100 pts	25 iter ×100 pts
Total model evaluations	500	500	2500	2500	2500	2500

Table 2: Settings of the algorithm for each strategy and specific benchmark problem. Abbreviations used: gen - generations, pop - population, init - initial population, iter - iterations, pts - design points

optimization both metrics are computed and stored to later compute, for example, the mean values of convergence. The NSGA2 optimization was performed 100 times for each benchmark problem. Due to its longer runtime for simple model functions the algorithm was applied 25 times to each problem. The whole sample should still be large enough to compare the results.

Box plot diagrams are used to visualize the resulting metrics for all runs. The box is bounded by the first and third quartile such that the box represents 50% of the values. The median is displayed as orange band inside the box. The whiskers extend to the minimum and maximum values. For every benchmark problem the box plots of the different strategies are combined into one figure to compare the results. Figure 13 shows the convergence metrics of all strategies for each benchmark problem. The diversity metric is shown in the same way in Figure 14. For every benchmark problem the resulting *Pareto fronts* of all strategies are displayed in Figure 12. The results were taken from an arbitrary run. If the resulting *Pareto front* includes more than 100 entries a subset of 100 solutions is chosen to better visualize the result. The optimal *Pareto front* as described previously for each problem is visualized as blue curve. In addition tables including the mean, standard deviation, maximum and minimum values for each strategy can be found in Appendix A.

Figure 13a shows the convergence metric of all approaches for the SCH1 problem. The results of NSGA2 are distributed between $\gamma = 0.0003$ and $\gamma = 22.41$. Around 35% of the runs did not find the correct solution while the convergence metric for the other runs is less than 0.1. This spread in the metric is a result of the large decision space where the optimal solutions only lie in a small subspace. Because the model uses quadratic functions, the further values are away from the optimal solutions in the decision space, the larger the value of the objective function becomes. Thus, the convergence metric for some runs is greater than 1 even if the found solutions may be relatively close to the optima with respect to the whole decision space. Nevertheless, the performance of the PAR and MIX strategy is much better than the performance

of NSGA2. With a minimum value around $\gamma \approx 0.0031$ and a maximum value, or worst convergence, around $\gamma \approx 0.27$ both approaches converged to the optimal solution in around 90% of the cases as shown in the zoomed in plot on the right. The difference between PAR and MIX can be neglected because it is a result of the discretization of the optimal *Pareto front* to compute the convergence metric. As described previously, CI uses a more globally focused approach. With a small number of samples compared to the interval for the decision variable the result is worse than all others. The diversity metric does not show much difference between the strategies. All strategies use the NSGA-II as the optimization method. This method finds good distributed solutions. The difference between the approaches is caused by the location of the boundary solutions of the approximated *Pareto front*. Figure 12a shows that the solutions found by PAR, MIX and CI do not extend to the boundaries of the optimal *Pareto front*. Therefore, the distance between the approximated and optimal boundary solutions increases, which causes the diversity metric to increase as well. The number of points on the *Pareto front* also has an impact on the diversity metric. With more points on the *Pareto front* it is more unlikely that all points are equally distributed. Table 3 shows the arithmetic mean of points lying on the *Pareto front* for the final iteration or generation. Because PAR, MIX and CI use *surrogate models* to find solution candidates these approaches have more candidates overall. Thus, their *Pareto front* consists of more points than the front found by NSGA2. The result looks smoother and discontinued parts are clearly visible.

For the FON benchmark problem the results are similar. Most metric values are smaller than for the SCH1 problem because the decision variables can only take values in the interval $[-4, 4]$. This leads to smaller objective values compared to SCH1 and thus the metric measuring the Euclidean distance to the true *Pareto front* is smaller even if the optimal solution is not found. Both strategies CI and NSGA2 were close to the optimal solution with their metrics lying around $\bar{\gamma} \approx 0.06$. The convergence values of PAR have an arithmetic mean around $\bar{\gamma} = 0.007$ while the convergence values of the MIX strategy are slightly worse with their mean around $\bar{\gamma} = 0.016$. PAR and MIX have the best convergence values followed by NSGA2 and CI, which performances are nearly equal as seen in Figure 13b. The visualization of the *Pareto front* for an arbitrary run in Figure 12b shows that the MIX and PAR strategy found the optimal solutions as opposed to CI and NSGA2, which did not fully converged. The mean of the diversity metric for all approaches is around $\bar{\Delta} \approx 0.65$. Compared with all other benchmark problems this is the best value. Figure 14b shows that the PAR and MIX strategy have a slightly better diversity than the CI strategy. The found solutions are distributed along the whole optimal *Pareto front*, which decreases the distance between the approximated and optimal boundary solutions and thus decreases the value of the diversity metric.

The results of the ZDT1 benchmark problem support the previous observations. The PAR and MIX approaches convergence is better than of CI and NSGA2 as shown in Figure 13c. With the mean value of the convergence metric being around $\bar{\gamma} = 0.67$ the NSGA2 strategy did not find the optimal *Pareto front*. The performance of the CI strategy is slightly better. The mean of the convergence metric over all runs is

$\bar{\gamma} \approx 0.24$ and seems to be closer to the optimal solution. The worse convergence metric is caused by the solutions found at the left boundary of the *Pareto front*. The prediction of all strategies using *surrogate models* is generally worse in this area. Considering the definition of the function $f_1(x) = x_1$ the left boundary corresponds to decision variables $x_1 = 0$. Because this is also the boundary of the decision space the predicted values might be extrapolated. This can cause the wrong predictions seen at the left of Figure 12c. If more points are added along the boundary the prediction gets better so that the *over-fitting* for the PAR and MIX strategy is not that high. Therefore, the mean convergence value of PAR and MIX is better ($\bar{\gamma} = 0.09$). The results of an arbitrary run shows that the found solutions in general are well distributed along the optimal *Pareto front*. However, the wrong predictions on the left side increase the diversity metric to $\bar{\Delta} = 0.84$ for PAR, $\bar{\Delta} = 0.87$ for CI and $\bar{\Delta} = 0.8$ for the MIX strategy. The mean of the diversity metric for NSGA2 is $\bar{\Delta} = 0.85$. Without the *over-fitting* caused by the *surrogate models*, the distance between the approximated and optimal boundary solutions is slightly better but the solutions are not equally distributed along the front.

For the ZDT2 problem Figure 13d shows that PAR and MIX have the best convergence values over all runs compared to the other strategies. The convergence values of the NSGA2 approach lie approximately in the interval $[0.9, 1.6]$ for all runs. The mean of the convergence metric for the CI strategy is $\bar{\gamma} = 1.09$. For MIX the convergence values of all runs are in the interval $[0.30, 2.03]$. Nevertheless, none of the approaches was able to find the optimal solutions in any run. One problem is the *over-fitting* due to extrapolation on the boundary of the decision space similar to the *over-fitting* in the ZDT1 problem. The diversity shown in Figure 14d can be neglected because the approximated *Pareto fronts* are neither similar in shape, nor in the range of the objective values. Thus, the mean of the diversity metrics of all strategies is around $\bar{\Delta} \approx 1$.

The disconnected *Pareto front* of the ZDT3 problem was almost found by the PAR approach as seen in Figure 12e. The PAR strategy leads to better results than NSGA2. The mean of the convergence metric is 0.12 for PAR as opposed to 0.59 for NSGA2. The CI approach did not model the oscillations of the optimal *Pareto front* correctly. Because new sample points are chosen based on the global model quality the local resolution is not high enough for the *surrogate model* to identify the oscillations. Nevertheless, the mean of the convergence values over all runs is 0.33 and Figure 12e shows that the solution found by CI is near the optimal *Pareto front*. The MIX strategy models the oscillations better due to half of the new *sampling* points being chosen at the current *Pareto front*. Therefore, the mean of the convergence over all runs is lower compared to CI and NSGA2 with a value of $\bar{\gamma} = 0.25$. NSGA2 did not find the optimal solutions but the approximated front is disconnected as opposed to the MIX and CI strategies. Thus, NSGA2 gives a better impression of the actual shape than CI or MIX even if the values are too large. The diversity is near one for all approaches, which is caused either by *over-fitting* or the discontinuity that is not considered for the metric calculation.

For the ZDT4 problem none of the approaches found a solution near the optimal

Pareto front, which should look the same as in ZDT1. Figure 12f shows that the approximated solutions have much higher values than the optimal front. The strategies were stuck on one of the many local optima and tend to only minimize the first objective function, which leads to these results. The convergence metric shown in Figure 13f as well as the diversity metric (Figure 14f) have the highest mean value compared to all other problems. For the PAR strategy, for example, the mean value is around 70. Because the objective function values of the approximated solutions are greater than 100 the convergence metric is greater than one.

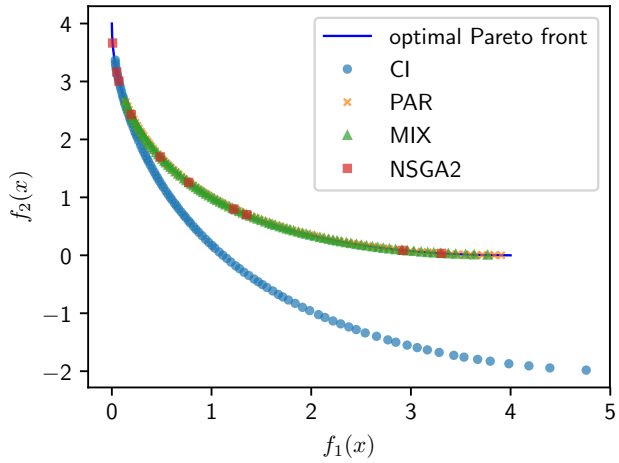
In Appendix A six tables show the *surrogate model* quality, which is computed as mean distance between predicted points on the *surrogate models* and the corresponding points from the original model. For the overall quality the objective values of 5000 points covering the whole decision space are predicted and compared with the values from the original model. Values near zero represent a better model quality. Because SCH1 includes quadratic objective functions and has the highest boundary values for the decision space compared to the other benchmark problems the model quality results in much higher values. Neglecting the SCH1 problem because the standard deviations have high values CI has the best overall model quality followed by the MIX and PAR strategy. However, considering only the model quality computed for the approximated *Pareto front* PAR and MIX have a better quality than the CI strategy. This is caused by the higher density of design points at the local optimal region. Adding design points "far" away from the global optimal solutions increases the overall *surrogate model* quality but does not lead to a better solution as these areas are of no importance to the optimization.

Method	SCH 1	FON	ZDT1	ZDT2	ZDT3	ZDT4
NSGA2	9.40	16.11	31.19	8.80	38.36	8.56
PAR	8563.52	1070.00	254.68	68.12	220.48	95.00
CI	7528.92	1107.96	314.24	98.20	288.08	120.96
MIX	5782.04	1095.08	328.00	64.68	191.24	103.48

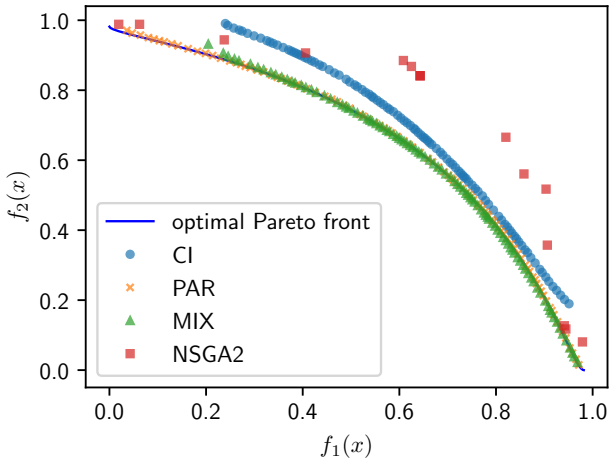
Table 3: Mean value of number of points on the approximated *Pareto front* over all runs. The algorithm using *surrogate models* lead to a higher number of solutions and thus a smoother representation of the *Pareto front*.

Considering the simple benchmark functions the PAR and MIX strategy outperformed the other strategies. The global focused approach CI is worse than NSGA2 for this number of function evaluations. The test problems ZDT2 and ZDT4 could not be solved by any of the approaches. Generally, the population of EAs tend to split on non-convex trade-off surfaces into individuals particularly strong in each of the objectives [11]. Therefore, the optimal front of the ZDT2 problem is more difficult to find because it is non-convex as opposed to the ZDT1 problem [26]. This coincides with the observations from the *Pareto fronts* of arbitrary runs where for ZDT2 most individuals tend to minimize only the first objective. A higher number of generations

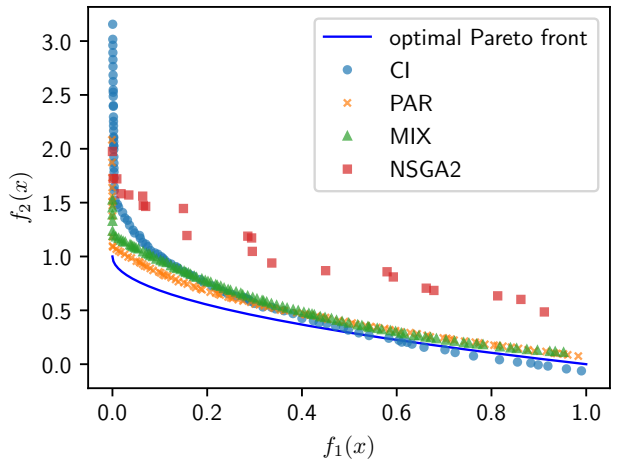
or iterations should increase the convergence to the optimal solution especially for ZDT2 and ZDT3. Deb et al. [9] tested the NSGA-II on the same problems but with 250 generations instead of the 25 used in this test case. Even with the higher number of generations, the mean value of the convergence metric for ZDT2 is only 8 times better than when using the MIX strategy. To overcome the local optima of ZDT4 the parameters of the NSGA-II have to be changed as proposed by Deb et al. [9]. Overall the PAR and MIX strategies showed the best performance. For the ZDT3 benchmark problem the PAR strategy leads to the best results. The results of PAR and MIX for the other benchmark problems were almost the same.



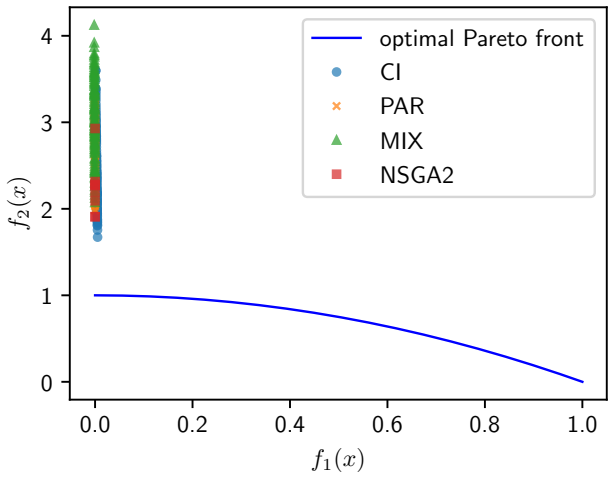
(a) Results of the SCH1 problem.



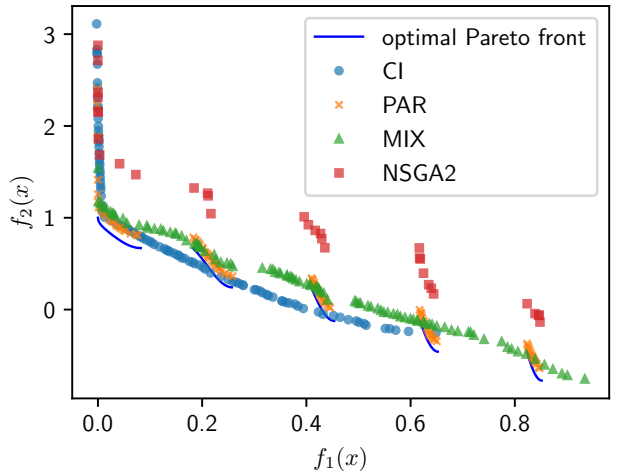
(b) Results of the FON problem.



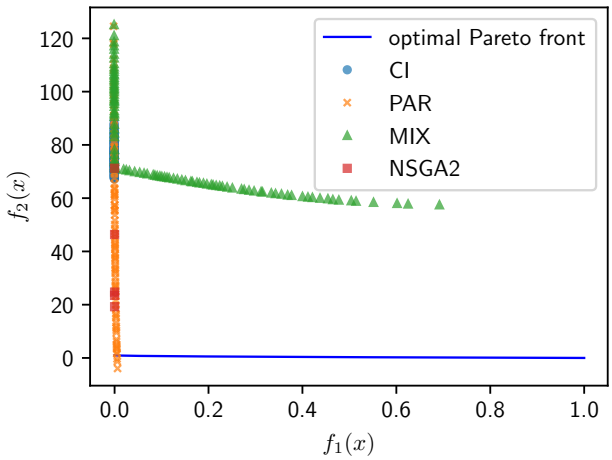
(c) Results of the ZDT1 problem.



(d) Results of the ZDT2 problem.

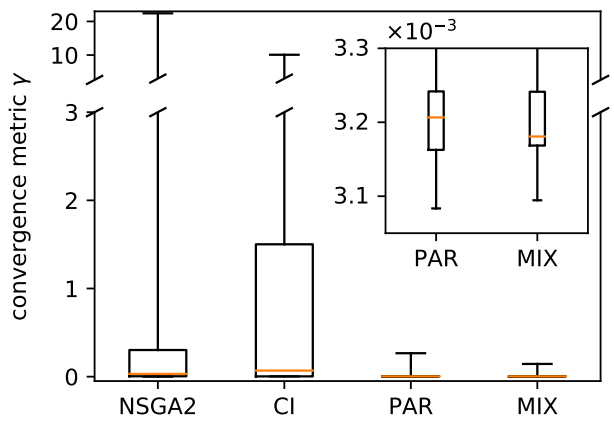


(e) Results of the ZDT3 problem.

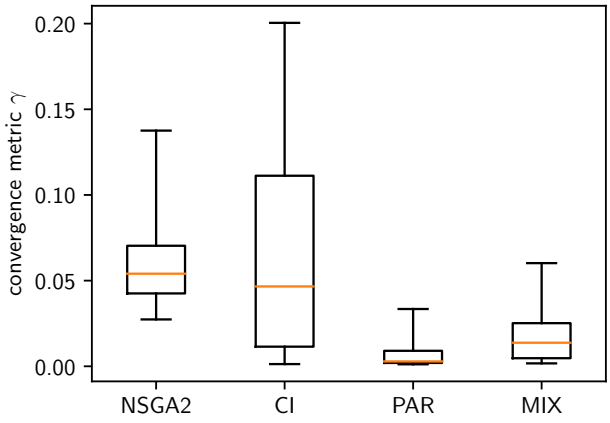


(f) Results of the ZDT4 problem.

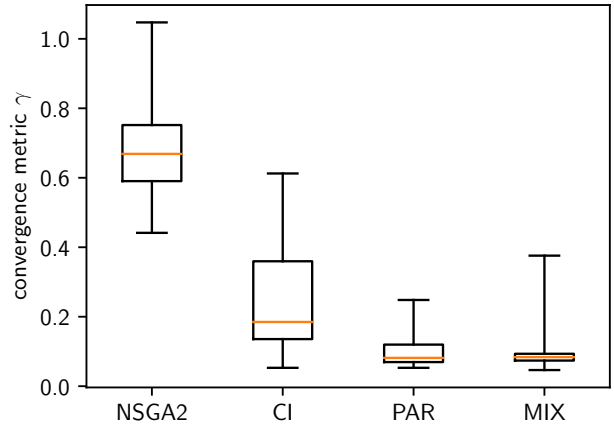
Figure 12: Resulting *Pareto fronts* obtained with all strategies on the different benchmark problems. The results are taken from an arbitrary simulation run.



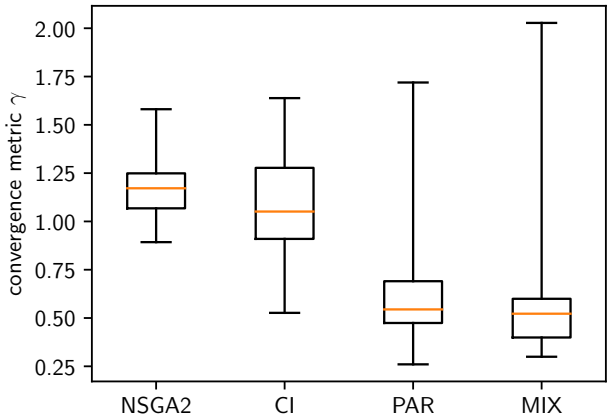
(a) Convergence metric for the SCH1 problem.



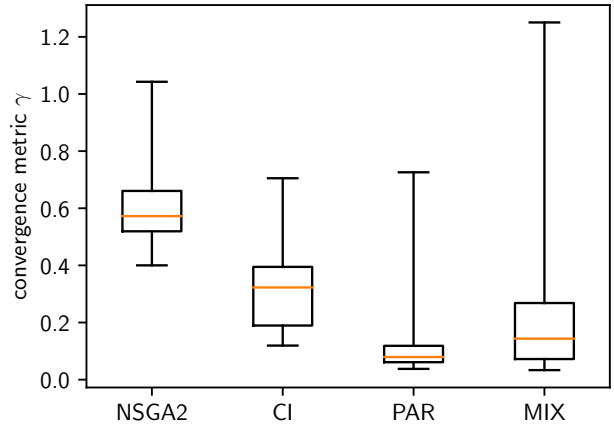
(b) Convergence metric for the FON problem.



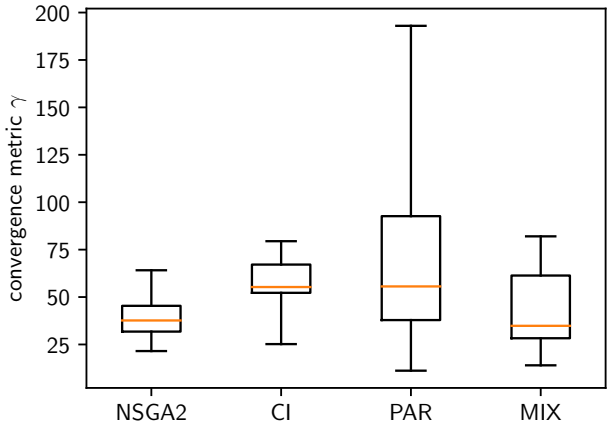
(c) Convergence metric for the ZDT1 problem.



(d) Convergence metric for the ZDT2 problem.

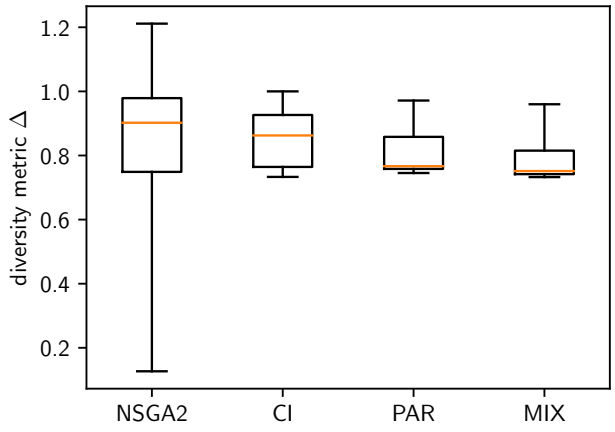


(e) Convergence metric for the ZDT3 problem.

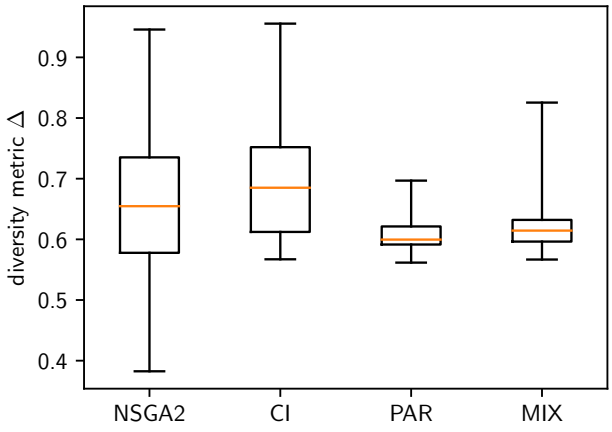


(f) Convergence metric for the ZDT4 problem.

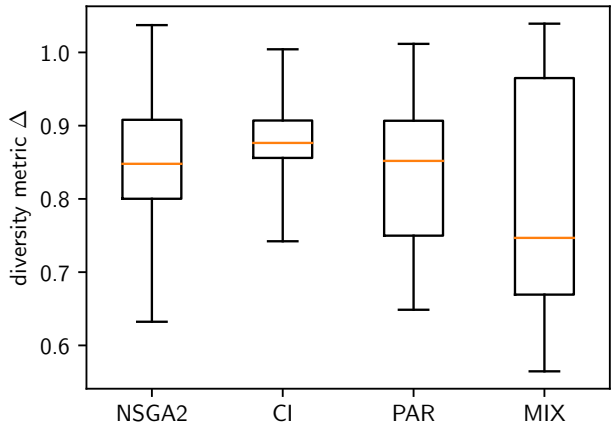
Figure 13: The resulting convergence metric shown as box plot for each strategy applied to the different benchmark problems. The box includes 50% of the values, the median is marked as orange band. The best value for the convergence metric is zero.



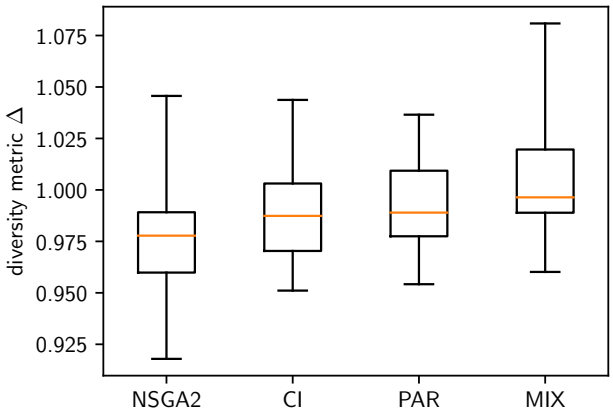
(a) Diversity metric for the SCH1 problem.



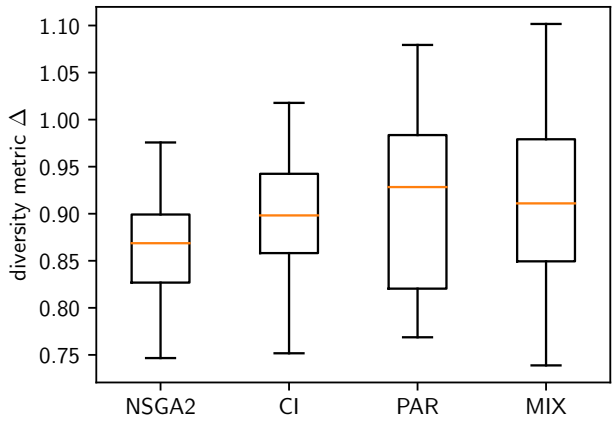
(b) Diversity metric for the FON problem.



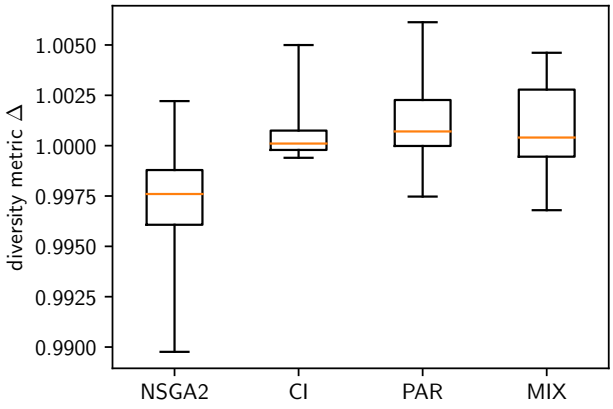
(c) Diversity metric for the ZDT1 problem.



(d) Diversity metric for the ZDT2 problem.



(e) Diversity metric for the ZDT3 problem.



(f) Diversity metric for the ZDT4 problem.

Figure 14: The resulting diversity metric shown as box plot for each strategy applied to the different benchmark problems. The box includes 50% of the values, the median is marked as orange band. The best value for the diversity metric is zero.

6. Practical application example

In this section the algorithm is applied to a practical application example. An electromagnetic motor is modelled, and its length, mass and noise emission should be minimized while the motor is running at a certain speed. Electromagnetic motors transform electrical energy into mechanical energy. The motor consists of a stator and a rotor. While the rotor can rotate freely, the stator is fixed. Usually a set of stationary magnets is built into the stator. The rotor consists of armatures with one or more windings of wire wrapped around an iron core. When current is running through the wire an electromagnetic field is produced. The opposite polarities of the magnetic field from the stationary magnets in the stator and the rotating electromagnetic field create a force. This causes the rotation of the rotor part. Alternating the direction of the current creates a rotating electromagnetic field. To obtain a constant rotation the direction of the current is reversed every time the rotor rotates by 180°. The number of windings, the coil size and the current sent through the coil are examples for parameters that influence the strength of the electromagnetic field and as such determine the rotation speed of the motor. Additionally, geometric parameters of the stator and rotor, as well as given voltage and current, are considered. In total 42 parameters are used as inputs for the model. 5 objectives including length, mass and an indicator for noise, vibration and harshness (NVH) have to be minimized. Because a constant speed is required it is set as a constraint. NVH is influenced by many parameters, therefore it should be optimized in a robust way. This means instead of only evaluating one design point to get the corresponding NVH values, multiple points around the initial design point are evaluated. The results are combined into single NVH values using the mean and standard deviation of all evaluated NVH values. For a subset of 28 parameters a stochastic distribution is given representing, for example, measurement errors or manufacturing tolerances. For these parameters a random value determined by the given distribution is added to the design point before it is evaluated. Additionally, some of the parameters are dependent on others and have to be calculated with a given formula for each design point.

The algorithm uses two DoEs, one for the optimization of mass and length and another one for the robust optimization of the NVH performance. The first DoE covers the 14 parameters used for the optimization of mass and length. To evaluate the model all 42 parameters are needed as input. Therefore, the first DoE is expanded to include the 28 robust parameters. For each robust parameter a given reference value is used such that these parameter values do not vary between different points of the first DoE. The second DoE covers the 28 robust parameters that influence the NVH performance. It is expanded with the other parameters in the same way as the first DoE to evaluate the model. When the surrogate models for NVH are evaluated 50 points around each design point are added and the resulting NVH indicator is a combination of the mean and standard deviation from the 50 additional evaluation points. If the model evaluation results in a violation of the constraint a penalty term is added for each objective corresponding to this design point. For the speed, used as constraint, a surrogate model is also built. This is used during the *Optimize* phase of

Applied strategies	initial DoE	additional samples at each iteration	NSGA-II settings	maximum iterations
PAR / CI / MIX	512 pts robust 128 pts	192 pts robust 32 pts	100 gen ×200 pop	9
Total model evaluations	640	224		2432

Table 4: Settings of the algorithm for the application example. The NVH objectives are evaluated in a robust way resulting in more design points. Abbreviations used: gen - generations, pop - population, pts - design points

the algorithm to ensure that solutions are found within feasible parameter regions.

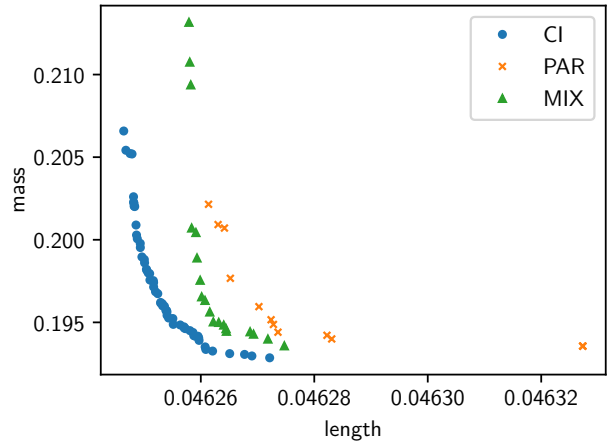
6.1. Simulation results

Three optimizations with the different strategies described in section 4 are performed. For each optimization 640 points are used for the initial DoEs. The DoE used for robust optimization has 512 initial points generated by an LHS. For the other DoE 128 initial points are used. At each iteration 192 new points are added to the DoE covering the robust parameters and 32 points are added to the other DoE. The algorithm runs for a maximum of 9 iterations. In total the original model is evaluated 2432 times. NSGA-II is used for the optimization of the surrogate models with a population of 200 individuals, which evolved over 100 generations. As explained in section 5.2 the strategies are abbreviated with PAR, CI and MIX.

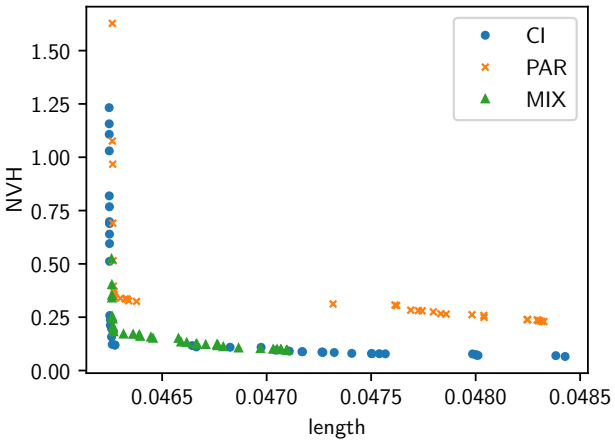
Figure 15 shows the trade-off curves for the objectives length, mass and NVH. The results of all strategies at the final iteration are combined into one figure to directly compare their results. The shape of the *Pareto fronts* is similar for all strategies. For all objectives the CI strategy found the lowest values. However, these results are predicted using the *surrogate models* so evaluating the solutions on the original model may lead to different results. The results of the benchmark problems showed that for the CI strategy the *surrogate model* is not as accurate as the PAR or MIX strategy at the *Pareto front*. Therefore, it is very likely that the solutions of the PAR and MIX strategies are better when evaluated on the original model. Figure 15d visualizes the trade-off between mass and NVH. The solutions found by the MIX strategy have the highest spread. The mass varies around 10% along the *Pareto front* whereas for the CI and PAR strategies the values only differ for approximately 5%. For the NVH indicator the solutions of the MIX strategy also have the greatest diversity compared to the other strategies. The length is nearly constant along the *Pareto front* for all strategies as shown in Figure 15a and Figure 15b. Compared to the other *Pareto fronts* the length and NVH curve in Figure 15b has a sharp bend. The NVH indicator highly differs for low length but stays nearly the same for all strategies with increasing length.

Figure 16 and Figure 17 illustrate the *Pareto fronts* after different iterations for all approaches. The trade-off curve for length and mass is nearly identical for all strategies. In addition, the change of the *Pareto front* between different iterations is really small (Figure 16a, 16b, 16c). The figures 16d, 16e and 16f display the *Pareto fronts* for the length and NVH objective. For the first iterations the CI and PAR strategies only found different solutions for the NVH objective while the length is constant. Only at the final iteration the approximated solutions also spread in the length dimension resulting in the sharp bend on the lower left. The MIX strategy approximated the sharp bend in the first iteration. For the next iterations the solutions do not change except in the fifth iteration where greater values for the length where approximated. The trade-off curve between the mass and NVH objective shows the largest difference between iterations for the approximated solutions. PAR and CI found high values for the NVH indicator at the first iteration. The next approximated *Pareto fronts* cover a wider range for the mass objective with values for the NVH objective smaller than one. The MIX strategy did not find a wide range at the first iteration. For this strategy the *Pareto front* covers a larger interval for each iteration.

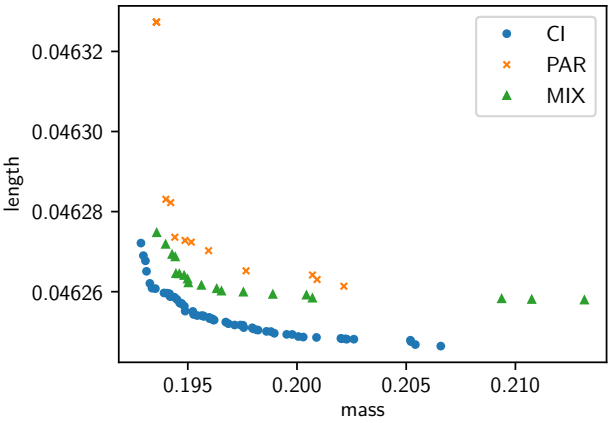
Considering the results of the benchmark functions PAR and MIX should lead to the same results. For mass and length their results are nearly identical. Both strategies have the largest difference for the mass and NVH objectives where the solutions of the MIX strategy cover a wider range for the mass. Generally, all strategies tend to find solutions in the same region with similar *Pareto fronts*. Additionally, the *Pareto fronts* did not rapidly change between the iterations. Therefore the approximated solutions might be globally or at least locally optimal. Although Figure 15 may lead to the conclusion that the CI strategy gives the best results, the benchmark tests showed that the convergence of PAR and MIX is better. The results are approximated using the *surrogate models* and thus depend on their quality in the specific region.



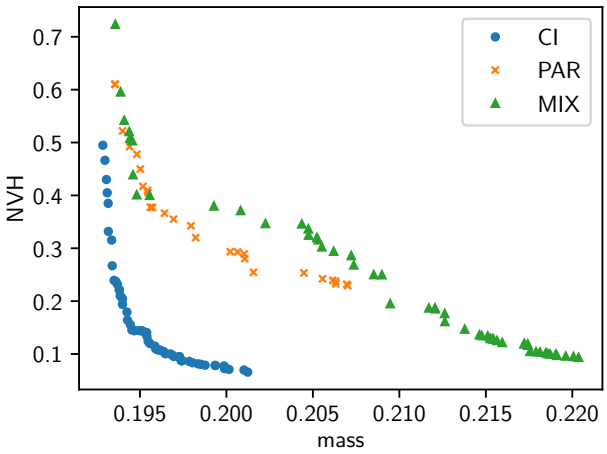
(a) Trade-off length - mass.



(b) Trade-off length - NVH.

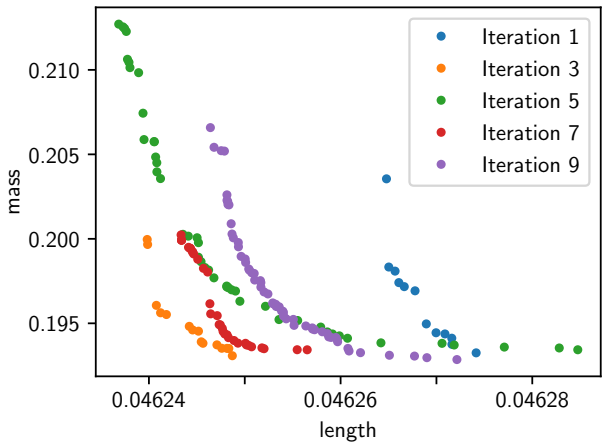


(c) Trade-off mass - length.

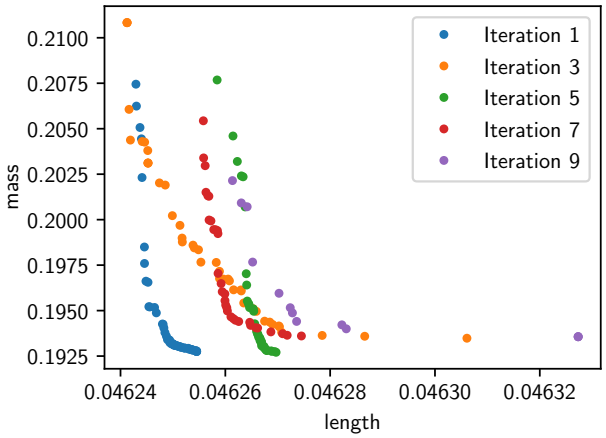


(d) Trade-off mass - NVH.

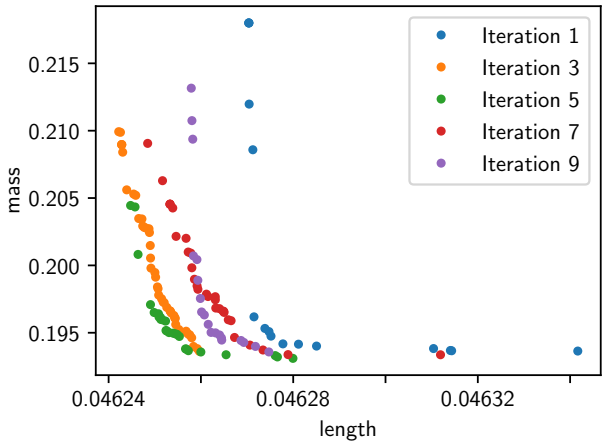
Figure 15: The resulting *Pareto fronts* of the different strategies are shown. The results depend on the quality of the *surrogate models* in the certain region. Therefore, the results obtained with the CI strategy may not be the best results.



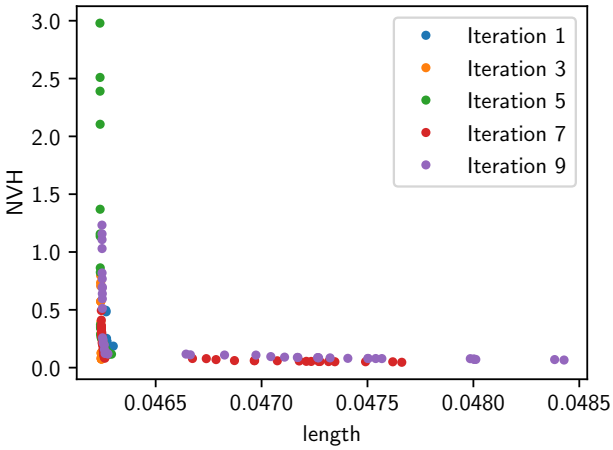
(a) Trade-off length - mass for CI strategy.



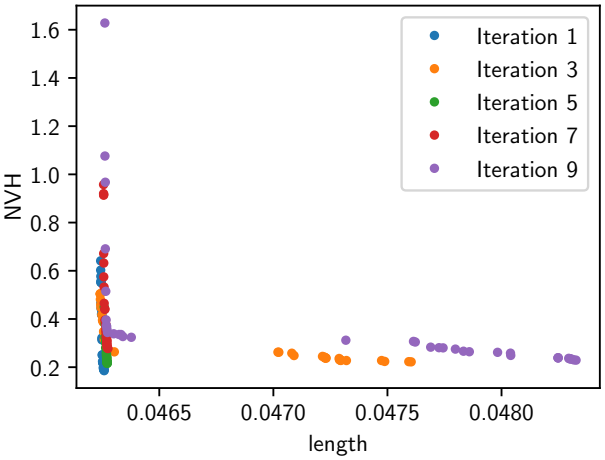
(b) Trade-off length - mass for PAR strategy.



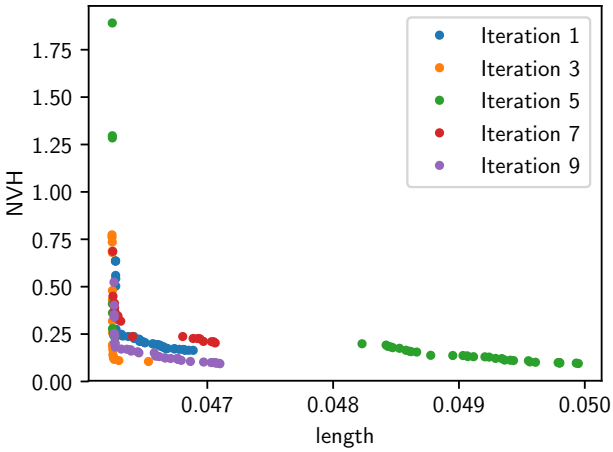
(c) Trade-off length - mass for MIX strategy.



(d) Trade-off length - NVH for CI strategy.

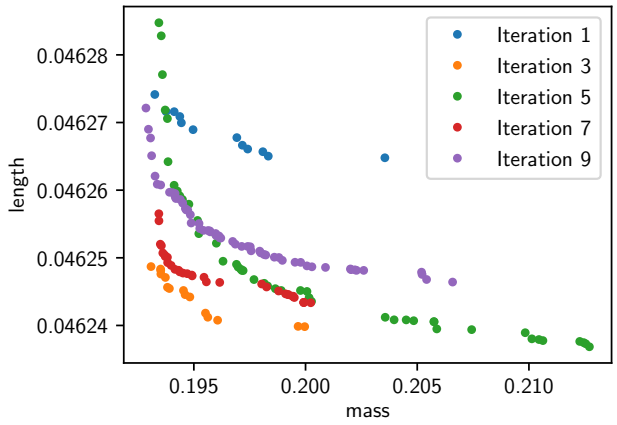


(e) Trade-off length - NVH for PAR strategy.

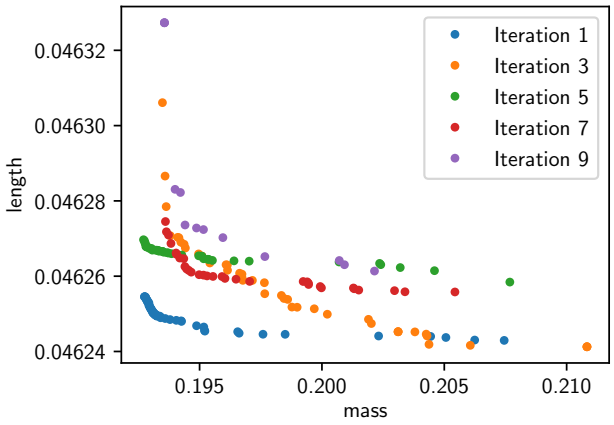


(f) Trade-off length - NVH for MIX strategy.

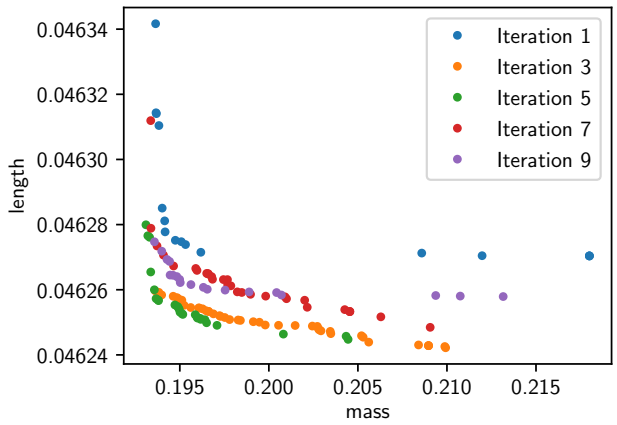
Figure 16: *Pareto fronts* obtained on the iterations $1, 3, 5, 7$ and 9 are shown for the different strategies.



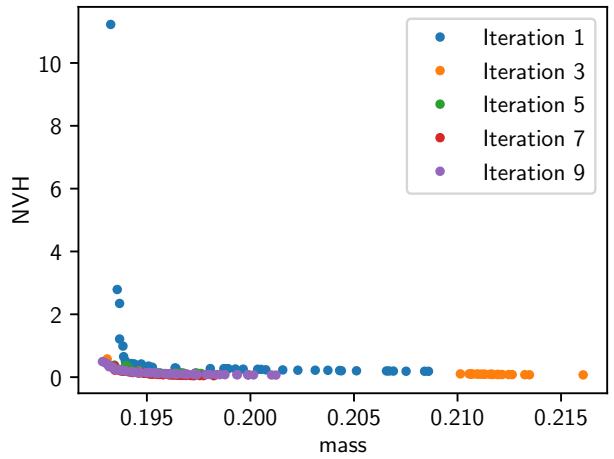
(a) Trade-off mass - length for CI strategy.



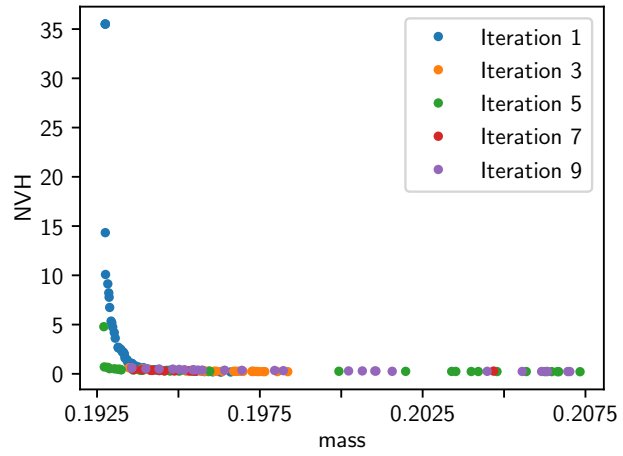
(b) Trade-off mass - length for PAR strategy.



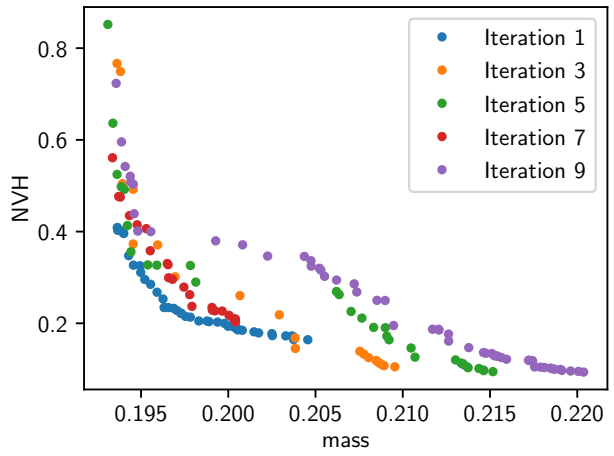
(c) Trade-off mass - length for MIX strategy.



(d) Trade-off mass - NVH for CI strategy.



(e) Trade-off mass - NVH for PAR strategy.



(f) Trade-off mass - NVH for MIX strategy.

Figure 17: *Pareto fronts* obtained on the iterations $1, 3, 5, 7$ and 9 are shown for the different strategies.

7. Conclusion and outlook

The algorithm proposed in this thesis was applied to various test problems and an application example. With the PAR strategy the algorithm found the correct solution for all benchmark problems except for ZDT2 and ZDT3 where none of the approaches were able to find the correct solution. The main problem seems to be solving MOPs with high dimensional decision spaces and only a small number of samples. Furthermore, some points on the surrogate models are extrapolated, which impairs the results as seen in some of the benchmark problems. A higher number of model evaluations should decrease the amount of extrapolated points and thus increase the overall solution accuracy. Another downside is that the algorithm heavily depends on the quality of the surrogate models, at least in the region around the global optimum. While *Gaussian process* models generally fit well, better surrogate models may exist for certain problems. However, this requires more insight into the specific problem. Nevertheless, the structure of the algorithm allows to continue an optimization from previous iterations, for example to increase the number of model evaluations.

For most of the benchmark problems the PAR and MIX approach performed better than an optimization using only an EA like NSGA-II. The CI strategy should not be used because its performance cannot compete with the other ones. By utilizing *surrogate models*, the found *Pareto front* consists of more points, which allows to see the actual shape as well as discontinuities in the trade-off curve better. With a fixed number of model evaluations, the results obtained with the algorithm are better than with an optimization using only NSGA-II. Compared to NSGA2 the algorithm converges faster to the optimal solution. Because the time needed for the optimization is neglectable compared to the time that is needed for one complex model evaluation, the algorithm does not take more time than an optimization using an EA. In addition, the modular set-up allows to adapt the algorithm individually for any MOP.

For the application example the MIX and PAR strategy lead to the best results when considering the worse *surrogate model* quality at the *Pareto front* for the CI strategy as observed for the benchmark problems. The approximated *Pareto fronts* have a similar shape and cover almost the same regions for all strategies. Additionally, they are not rapidly changing between iterations, which leads to the conclusion that an optimal solution was found for the application example.

The most promising strategy seems to be the PAR strategy. To further increase the performance of the algorithm the newer NSGA-III optimization method could be implemented. In addition an automatic evaluation of the optimal solutions with the original model would help to check the reliability of the approximated solutions. Therefore, an additional step could be added to the end of the algorithm. Further work would involve the investigation of the influence of the initial and adaptive sample size on the algorithm. The algorithm, from a combination of *surrogate models*, *adaptive sampling* and EAs, to solve MOPs provides better results than solving them with an EA only. However, it still needs fine-tuning dependent on the problem to provide reliable results for any given problems.

A. Appendix

This appendix includes the underlying data from the box plots shown in section 5.2. For the NSGA2 strategy 100 optimizations were performed to generate the box plot. The other strategies were performed 25 times because the runtime is longer than for NSGA2. The following tables include the mean, standard deviation, maximum and minimum value of the convergence and diversity metric for each strategy over all performed optimizations on the benchmark problems.

The last six tables show the *surrogate model* quality, which is computed as mean distance between predicted points on the *surrogate models* and the corresponding points from the original model. For the overall quality the objective values of 5000 points covering the whole decision space are predicted and compared with the values from the original model. The *Pareto front* quality is computed as mean distance between the approximated *Pareto front* and the original model values for the corresponding *Pareto archive*. Values near zero represent a better model quality.

Problem	mean	standard deviation	maximum	minimum
SCH1	1.0989	3.5889	22.4133	0.0003
FON	0.0595	0.0221	0.1376	0.0274
ZDT1	0.6726	0.1110	1.0473	0.4415
ZDT2	1.1672	0.1399	1.5809	0.8928
ZDT3	0.5950	0.1135	1.0427	0.4001
ZDT4	38.5581	9.3837	64.1226	21.5245

Table A.1: Mean, standard deviation, maximum and minimum value of the convergence metric over 100 runs for the NSGA2 strategy.

Problem	mean	standard deviation	maximum	minimum
SCH1	0.8363	0.2285	1.2114	0.1267
FON	0.6576	0.1085	0.9459	0.3825
ZDT1	0.8504	0.0717	1.0373	0.6323
ZDT2	0.9753	0.0225	1.0457	0.9179
ZDT3	0.8637	0.0531	0.9757	0.7466
ZDT4	0.9972	0.0025	1.0022	0.9898

Table A.2: Mean, standard deviation, maximum and minimum value of the diversity metric over 100 runs for the NSGA2 strategy.

Problem	mean	standard deviation	maximum	minimum
SCH1	1.4931	2.6012	10.0853	0.0031
FON	0.0659	0.0622	0.2004	0.0013
ZDT1	0.2383	0.1530	0.6124	0.0529
ZDT2	1.0927	0.2800	1.6383	0.5267
ZDT3	0.3334	0.1641	0.7050	0.1193
ZDT4	57.3886	13.8405	79.4477	25.1939

Table A.3: Mean, standard deviation, maximum and minimum value of the convergence metric over 25 runs for the CI strategy.

Problem	mean	standard deviation	maximum	minimum
SCH1	0.8608	0.0859	1.0000	0.7332
FON	0.7056	0.1106	0.9556	0.5671
ZDT1	0.8728	0.0619	1.0043	0.7422
ZDT2	0.9911	0.0245	1.0437	0.9511
ZDT3	0.8953	0.0616	1.0178	0.7517
ZDT4	1.0004	0.0011	1.0050	0.9994

Table A.4: Mean, standard deviation, maximum and minimum value of the diversity metric over 25 runs for the CI strategy.

Problem	mean	standard deviation	maximum	minimum
SCH1	0.0137	0.0515	0.2659	0.0031
FON	0.0070	0.0075	0.0334	0.0012
ZDT1	0.0997	0.0435	0.2482	0.0529
ZDT2	0.6449	0.3049	1.7192	0.2601
ZDT3	0.1262	0.1391	0.7259	0.0377
ZDT4	70.0718	44.4524	192.9944	11.1778

Table A.5: Mean, standard deviation, maximum and minimum value of the convergence metric over 25 runs for the PAR strategy.

Problem	mean	standard deviation	maximum	minimum
SCH1	0.8121	0.0737	0.9714	0.7453
FON	0.6073	0.0284	0.6968	0.5616
ZDT1	0.8398	0.1006	1.0117	0.6487
ZDT2	0.9924	0.0216	1.0365	0.9542
ZDT3	0.9087	0.0905	1.0793	0.7687
ZDT4	1.0012	0.0020	1.0061	0.9975

Table A.6: Mean, standard deviation, maximum and minimum value of the diversity metric over 25 runs for the PAR strategy.

Problem	mean	standard deviation	maximum	minimum
SCH1	0.0092	0.0276	0.1443	0.0031
FON	0.0168	0.0152	0.0602	0.0017
ZDT1	0.0923	0.0606	0.3758	0.0464
ZDT2	0.5715	0.3243	2.0276	0.2995
ZDT3	0.2489	0.2873	1.2504	0.0334
ZDT4	42.3694	21.5826	82.0046	14.0071

Table A.7: Mean, standard deviation, maximum and minimum value of the convergence metric over 25 runs for the MIX strategy.

Problem	mean	standard deviation	maximum	minimum
SCH1	0.7861	0.0666	0.9598	0.7326
FON	0.6294	0.0636	0.8254	0.5667
ZDT1	0.8048	0.1530	1.0392	0.5646
ZDT2	1.0051	0.0311	1.0809	0.9602
ZDT3	0.9117	0.0940	1.1016	0.7388
ZDT4	1.0008	0.0022	1.0046	0.9968

Table A.8: Mean, standard deviation, maximum and minimum value of the diversity metric over 25 runs for the MIX strategy.

Problem	mean	standard deviation	maximum	minimum
SCH1	1.2112	0.4171	1.9187	0.4044
FON	0.1076	0.0663	0.2997	0.0068
ZDT1	0.2199	0.1130	0.4904	0.0534
ZDT2	0.1467	0.0309	0.2107	0.0850
ZDT3	0.4082	0.2219	0.8929	0.0719
ZDT4	46.4004	16.3670	81.9113	13.0087

Table A.9: Mean, standard deviation, maximum and minimum of the quality of the *surrogate models* at the *Pareto front* over 25 runs for CI.

Problem	mean	standard deviation	maximum	minimum
SCH1	0.0033	0.0084	0.0392	0.0000
FON	0.0247	0.0292	0.1235	0.0003
ZDT1	0.0476	0.0280	0.1853	0.0083
ZDT2	0.1922	0.0504	0.2757	0.1015
ZDT3	0.3642	0.1893	1.0196	0.0809
ZDT4	165.6757	47.9357	253.5300	79.7393

Table A.10: Mean, standard deviation, maximum and minimum of the quality of the *surrogate models* at the *Pareto front* over 25 runs for PAR.

Problem	mean	standard deviation	maximum	minimum
SCH1	0.0149	0.0329	0.1675	0.0000
FON	0.0291	0.0208	0.0979	0.0002
ZDT1	0.0366	0.0232	0.1312	0.0027
ZDT2	0.0742	0.0297	0.1368	0.0277
ZDT3	0.4532	0.3281	1.3482	0.0769
ZDT4	106.8670	30.6544	169.6712	41.1139

Table A.11: Mean, standard deviation, maximum and minimum of the quality of the *surrogate models* at the *Pareto front* over 25 runs for MIX.

Problem	mean	standard deviation	maximum	minimum
SCH1	79.0229	559.8355	11879.4361	0.0005
FON	0.0439	0.0566	0.6128	0.0002
ZDT1	0.0615	0.0516	0.4543	0.0002
ZDT2	0.0153	0.0110	0.0849	0.0001
ZDT3	0.0933	0.0735	0.4648	0.0002
ZDT4	19.7691	14.7692	89.8983	0.0056

Table A.12: Mean, standard deviation, maximum and minimum of the overall quality of the *surrogate models* over 25 runs for CI.

Problem	mean	standard deviation	maximum	minimum
SCH1	244.6748	1681.5905	30958.8642	0.0000
FON	0.0704	0.0651	0.5110	0.0004
ZDT1	0.1085	0.0896	0.7407	0.0003
ZDT2	0.0885	0.0625	0.3749	0.0002
ZDT3	0.3619	0.2845	1.7527	0.0006
ZDT4	27.8404	22.0965	182.0857	0.0112

Table A.13: Mean, standard deviation, maximum and minimum of the overall quality of the *surrogate models* over 25 runs for PAR.

Problem	mean	standard deviation	maximum	minimum
SCH1	36.2730	229.4371	7004.5644	0.0001
FON	0.0317	0.0476	0.5075	0.0001
ZDT1	0.0768	0.0604	0.5000	0.0002
ZDT2	0.0552	0.0385	0.2588	0.0002
ZDT3	0.3392	0.2737	1.5675	0.0005
ZDT4	22.7608	17.4891	119.7605	0.0058

Table A.14: Mean, standard deviation, maximum and minimum of the overall quality of the *surrogate models* over 25 runs for MIX.

References

[1] Brian M Adams, Lara E Bauman, William J Bohnhoff, Keith R Dalbey, John P Eddy, Mohamed S Ebeida, Michael S Eldred, Patricia D Hough, Kenneth T Hu, John D Jakeman, et al. Version 6.0 theory manual. *Sandia National Laboratories, Tech. Rep. SAND2014-4253*, pages 47–57, 2014.

[2] Ake Bjorck. *Numerical methods for least squares problems*. Siam, 1996.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] Yair Censor. Pareto optimality in multiobjective problems. *Applied Mathematics and Optimization*, 4(1):41–59, 1977.

[5] Jared L Cohon. *Multiobjective programming and planning*, volume 140. Courier Corporation, 2004.

[6] Kevin Cremanns and Dirk Roos. Deep gaussian covariance network. *arXiv preprint arXiv:1710.06202*, 2017.

[7] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, 45(3): 35:1–35:33, 2013.

[8] Olivier L De Weck. Multiobjective optimization: History and promise. In *Invited Keynote Paper, GL2-2, The Third China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems, Kanazawa, Japan*, volume 2, page 34, 2004.

[9] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[10] Carlos M Fonseca and Peter J Fleming. Multiobjective genetic algorithms made easy: selection sharing and mating restriction. *IET Conference Proceedings*, 1995.

[11] Carlos M Fonseca and Peter J Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation*, 3(1):1–16, 1995.

[12] Anthony Giunta, Steven Wojtkiewicz, and Michael Eldred. Overview of modern design of experiments methods for computational simulations. In *41st Aerospace Sciences Meeting and Exhibit*, page 649, 2003.

[13] DE Goldberg. Genetic algorithms in search, optimization, and machine learning, addison-wesley, reading, ma, 1989. *Google Scholar*, 2014.

[14] C-L Hwang and Abu Syed Md Masud. *Multiple objective decision making—methods and applications: a state-of-the-art survey*, volume 164. Springer Science & Business Media, 2012.

[15] Yaochu Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing*, 9(1):3–12, 2005.

[16] Donald R Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383, 2001.

[17] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4): 455–492, 1998.

[18] Georges Matheron. Principles of geostatistics. *Economic geology*, 58(8):1246–1266, 1963.

[19] Kaisa Miettinen. A posteriori methods. In *Nonlinear multiobjective optimization*, pages 77–113. Springer, 1998.

[20] Richard E Rosenthal. Concepts, theory, and techniques principles of multiobjective optimization. *Decision Sciences*, 16(2):133–152, 1985.

[21] Jerome Sacks, Susannah B Schiller, and William J Welch. Designs for computer experiments. *Technometrics*, 31(1):41–47, 1989.

[22] Jerome Sacks, William J Welch, Toby J Mitchell, and Henry P Wynn. Design and analysis of computer experiments. *Statistical science*, pages 409–423, 1989.

[23] James David Schaffer. Some experiments in machine learning using vector evaluated genetic algorithms. Technical report, Vanderbilt Univ., Nashville, TN (USA), 1985.

[24] Po-Lung Yu. A class of solutions for group decision problems. *Management Science*, 19(8):936–946, 1973.

[25] Milan Zelany. A concept of compromise solutions and the method of the displaced ideal. *Computers & Operations Research*, 1(3-4):479–496, 1974.

[26] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.