Universiteit Leiden

Master Computer Science

Efficiently solving Expensive Multi-Objective Black-Box Optimization Problems using Optimally Weighted Ensembles of Surrogate Models.

Name:	Iddo Johanan Gideon Hanse
Student ID:	S1630784
Date:	02/11/2020
Specialisation:	Data Science
1st supervisor:	Thomas Bäck
2nd supervisor:	Bas van Stein

Master's Thesis in Computer Science (DRAFT)

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

The process of industrial design engineering is often involved with the simultaneous optimization of multiple highly costly objectives. Such Multi-objective Optimization Problems (MOPs) have been widely solved using surrogate models. The S-Metric Selection Efficient Global Optimization (SMS-EGO) algorithm, an efficient surrogate-assisted approach to multi-objective optimization, has become particularly popular in optimization literature. In this thesis, the SMS-EGO algorithm is extended with optimally weighted, linearly combined ensembles of regression models to improve its efficiency. The main idea is to predict multiple objective functions individually by combining the output of multiple surrogate models into optimally weighted ensembles. In addition, uncertainty about predictions is quantified and incorporated in the ensembles to balance between exploration and exploitation, forcing the algorithm to find well-spread data points with minimal target values on all objective functions. The performance of the proposed algorithm is evaluated on a diverse set of benchmark problems with a budget of merely 25 evaluations of the real objective functions. The experimental results show that the proposed Ensemble-based S-Metric Selection Efficient Global Optimization (E-SMS-EGO) algorithm outperforms three state-of-the-art multi-objective optimization algorithms in terms of efficiency, robustness and spread across the objective space.

Acknowledgements

First, I would like to thank Bas van Stein and Thomas Bäck for helping me find this interesting subject on multi-objective optimization, helping me get started with writing and referring me to the right people.
Furthermore, my special gratitude goes to Roy de Winter, who was, at any time, willing and available to provide me with technical knowledge, as well as practical details on writing this thesis.
Finally, I want to thank Capgemini for offering me the opportunity to write this thesis as part of a very enjoyable internship, with special thanks to Marijn Markus and Elise Lakerveld for their enthusiasm in showing me around and for helping me find my way in such a big company.

Contents

–A fresh start is a journey. A journey that requires a plan.

Vivian Jokotade

1

Introduction

In a world of technology, we find ourselves surrounded by all sorts of intelligent and well-built industrial applications. The cars we drive are becoming faster, the chips in our electronic devices are becoming smaller, and airplanes take us all across the world at an ever-increasing rate. In order to make this possible, industrial engineers are constantly optimizing the design of their products to achieve better performance and efficiency while at the same time trying to minimize the costs.

Industrial design processes are often considered to be black-box optimization problems, as it is generally uncertain how a change in design configuration affects the intended objective [34]. Therefore, gaining insight into the objective landscape of a certain product requires industrial engineers to evaluate proposed design configurations in an iterative manner [39]. Nowadays, as more and more computational resources become available, the evaluation of such configurations can be performed very accurately by running advanced computerized simulations. As a result, computational optimization techniques have been increasingly deployed to aid the optimization of industrial products over the last couple of decades [28, 69].

Even though the available computation power grows, such computerized simulations are often so costly that it still takes a lot of time and resources to evaluate objectives, especially when trying to optimize multiple, possibly conflicting objectives [34]. To avoid spending an excessive amount of time and resources at design evaluation, a widely used technique is to fit a regression model through the objective values of previously evaluated designs, to

approximate the costly objective function. In particular, ensembles of multiple, diverse surrogate models have been successfully applied to approximate costly objective functions and they were shown to yield great performance in optimization tasks [39, 64]. Especially when no prior knowledge about the behavior of the objective function is available, ensembles of surrogate models have been proven to perform really well [54].

In addition to optimizing a single objective, industrial engineering applications often require multiple objectives to be optimized simultaneously. Often, these different objectives behave in very different ways and can even be conflicting. For example, when optimizing the design of a car, one objective could be the maximum speed which should be maximized, whereas the fuel usage should be minimized as a second objective. In this case, the objectives are conflicting, which makes it impossible to designate the optimal configuration in an obvious manner. A lot of research has been dedicated to solve such Multi-Objective Optimization Problems (MOPs) [70]. Also, with the interest of reducing computational costs, multiple proposed frameworks make use of surrogate models to predict the costly evaluation functions based on a few evaluated points. Besides being used for single objective optimization problems, the use of surrogate models has also been proven to be effective for multiple-objective optimization tasks [37]. However, existing techniques unfortunately often still depend on domain-specific prior knowledge about the given optimization tasks. To eliminate this need for prior understanding of MOP landscapes, this thesis studies how diverse surrogate models can be combined into adaptive ensembles to solve computationally very expensive multi-objective optimization problems in an efficient manner.

1.1 Research Questions

Intending to investigate the usefulness of adaptive ensembles of surrogate models in a multi-objective optimization setting, this thesis provides an answer to the following research question:

How can convex linear combinations of surrogate models be used to solve computationally very expensive multi-objective optimization problems in an efficient manner?

An answer to this research question will be formulated by answering the following sub-questions:

1. **How can convex linear combinations of surrogate models be used to accurately predict costly objective functions?**
   Friese and Bartz-Beielstein [32] proposed to create ensembles of surrogate base models by linearly combining surrogate model weights. This method was shown to be beneficial in the creation of well-performing

surrogate ensembles, which were able to accurately predict costly objective functions. In order to include a large amount of base models in their ensembles, an evolutionary search on the model weights was shown to perform well. In this thesis, an adjusted version of this convex linear ensemble technique is implemented to investigate its efficiency and robustness in the context of multi-objective optimization.

2. **How does the inclusion of uncertainty quantification measures contribute to the performance of linearly combined ensembles of surrogate models?**
When it comes to global optimization with the use of surrogate models, new points are predicted by finding the most promising regions on the surrogate landscape. Normally, the predicted value is considered to be the most important decision factor in the proposal of new points [58]. However, using the predicted value as the only decision factor can easily lead to getting stuck in local minima [49]. In order to overcome this problem, an uncertainty quantification measure can be used to gain insight in the model's confidence about a predicted data point. Unfortunately, a considerable number of surrogate techniques does not automatically take the uncertainty of points into account [73]. Especially when creating ensembles of surrogates, uncertainty quantification measures are not trivial to calculate and use. With the intention to enhance the linear combination ensemble method, this thesis therefore studies how including uncertainty quantification measures affects the performance of the model.

3. **How can ensemble surrogates for different objectives be combined in an optimal way to obtain the Pareto-front in multi-objective optimization tasks?**
When trying to optimize multiple objective functions, the outcomes of the individual objective functions have to be combined in a certain manner. For this purpose, several methods have been proposed to combine the information from individual objective landscapes in such a way that multiple objectives can be optimized at the same time, while taking the mutual dependencies of all objectives into account. The underlying techniques for these methods differ drastically, varying from scalarization-based approaches [53] to set-based approaches [41]. To answer the third sub-question, this thesis studies how to combine the individual combinations of surrogate models in an optimal way in order to obtain an adequate Pareto-front. A very promising method for finding the Pareto-optimal points is to use the S-metric selection criterion [57]. Therefore, it is investigated how the fundamentals of the S-metric selection method can be used in a sequential manner to optimize multiple ensembles of surrogates.

4. **In terms of efficiency and function evaluations, how does the proposed multi-objective approach perform in comparison to competitive state-of-the-art multi-objective optimization algorithms?**
   Over the past decades, surrogate-assisted optimization algorithms have been used and studied extensively with regard to optimization problems with multiple objectives. Several famous and widely used algorithms like NSGA-II[21], MOEA/D[85], C-TAEA[48] have, among other algorithms, been shown to perform well on multi-objective optimization problems. Existing algorithms mainly differ in the way in which potential candidates are chosen and in how the multiple objectives are weighed up against each other, resulting in different performances in terms of efficiency and accuracy. Whereas some algorithms focus on minimizing the number of objective function evaluations [34, 42, 57], other algorithms try to obtain the best possible result, sometimes at the expense of more objective function evaluations [20, 50]. Since the proposed method in this thesis project aims at optimizing accuracy as well as minimizing the required objective function evaluations, it is studied how the algorithm performs compared to similar state-of-the-art algorithms.

1.2 Thesis Outline

The remainder of this thesis is structured as follows:
To provide the reader with the basic knowledge that is considered to be required to read this thesis, some preliminary concepts will first be discussed in Chapter 2.
Subsequently, related literature will be discussed in more detail in Chapter 3 to provide an overview on what has already been written about the topic.
A formal description of the multi-objective optimization problem in industrial engineering is given in Chapter 4, after which Chapter 5 states the proposed solution to this problem.
Chapter 6 comprehensively describes the experiments that were conducted to evaluate the proposed algorithm, as well as the results.
This work is concluded by giving an overview on the contributions and limitations of the proposed methods in Chapter 7, along with recommendations for possible future research.

–Without knowledge, action is useless.

Abu Bakr

2

Preliminaries

In this chapter, a brief overview of basic concepts is provided to equip the reader with the requisite knowledge to read this thesis. First, an introduction on general surrogate-assisted optimization is given, after which the concept of multi-objective optimization is described. The reader is considered to be familiar with basic mathematical and computer science concepts like (multi-variate) regression, optimization and linear algebra. Please note that the mathematical optimization processes in this thesis are applied with the intention to minimize objectives. However, a minimization problem can be easily transformed into a maximization problem without loss of generality (multiply by -1).

2.1 Surrogate-assisted Optimization

In mathematical optimization, the goal is to find the best solution out of a set of feasible solutions, where the quality of the solutions is provided by an objective function. As we consider minimization in this thesis, the best solution would be the solution with the lowest value on the objective function on a given interval of one or more independent variables, formally stated:

$$\min_x f(x) \tag{2.1.1}$$

In case of a known mathematical objective function, this optimum can often be found easily by means of numerical differential methods [59]. However, many real-world optimization problems lack such a formal mathematical definition of the objective function. For example, when minimizing the amount of

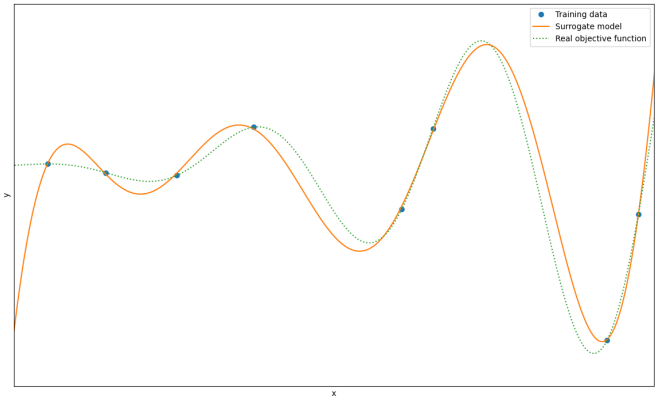


Figure 2.1.1: An example of a surrogate model approximating an objective function.

drag on an aircraft wing, there is no pre-defined formula available to calculate the effect of adjustments to the wing design [77]. Therefore, every variable configuration has to be evaluated, possibly by using simulations, in order to obtain the corresponding objective value. In that case, obtaining insight in the whole objective landscape would be infeasible, as it takes too long to evaluate every individual configuration. As a result, finding the optimum of the objective function becomes impossible in reasonable time, since the shape of the function is unknown. Fortunately, the real, costly objective function can be approximated by fitting a regression model through a selection of known, already evaluated configurations.

Such models, which mimic the behavior of a function, are referred to as surrogate models, also called meta-models or response surface models [39]. In the broadest sense, a surrogate model is mathematically defined as a function $\hat{f}(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$, approximating a real objective function $f(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$, which models the dependence of a response variable y on one or more (n) independent variables $\vec{x} = \{x_1, \ldots, x_n\}$. The surrogate-model is trained on a collection of realizations $\{\vec{x}_i, y_i\}^N$, with N being the total amount of available data points. Fig. 2.1.1 illustrates how a surrogate model can be fitted to approximate a true objective function. Subsequently, the optimum of the surrogate model can be derived using classical optimization techniques, e.g., differentiation, random search or more sophisticated methods [59]. If the surrogate model is sufficiently accurate, it can be assumed that the found optimum closely corresponds to the optimum of the real objective function. The way in which the accuracy and reliability of surrogate models can be measured will be addressed in Chapter 3.

2.1.1 Types of Surrogate Models

A wide variety of models can potentially be used as surrogate model, depending on the shape and behavior of the objective function in question. In this section, the most widely used surrogate models are featured, briefly discussing their operation and applications.

Support Vector Regression

The idea of support vector machines (SVM's) was first proposed by Boser et al. [11], who considered data points to be vectors in space. Initially, SVM's were only used for classification by introducing an optimally separating hyperplane to divide data points into classes. The idea behind this algorithm is to maximize the distance ϵ between the input vectors that are closest to this hyperplane, also called the support vectors. If the input vectors are not linearly separable, they can be mapped into a higher dimensional space Z, making it possible to linearly separate the data points by means of a new hyperplane.

Drucker et al. [25] showed that the idea of support vectors can also be used for regression, in which the goal is to train a function $f(\vec{x})$, ensuring that for all of the training points, it does not deviate more than the distance treshold ϵ from the actual obtained data points y_i. At the same time, it is ensured that the function is as flat as possible. For example, in case of a simple linear function taking the form

$$f(x) = w \cdot x + b \text{ with } w \in \mathbb{R},\ b \in \mathbb{R}, \tag{2.1.2}$$

a support vector regression model is found by minimizing w

$$\text{subject to } \begin{cases} y_i - w \cdot x_i - b \leq \epsilon \\ w \cdot x_i + b - y_i \leq \epsilon \end{cases} \tag{2.1.3}$$

Fig. 2.1.2 provides a simple graphical representation of a SVM used for linear regression. For a clear and more detailed description of SVM in the regression setting see [65].

Tree-based Regression

Decision trees as first formulated by Breiman et al. [15] have been applied widely in and outside the field of computer science, both for classification and regression. A decision tree can be seen as a flowchart-like graph built out of nodes and branches. In every internal node, an attribute is tested for a given property, to which various branches form the possible outcomes. When there are no more attributes left to test on, so-called leaf nodes form the final outcomes for a given problem. In the classification setting, the tree is

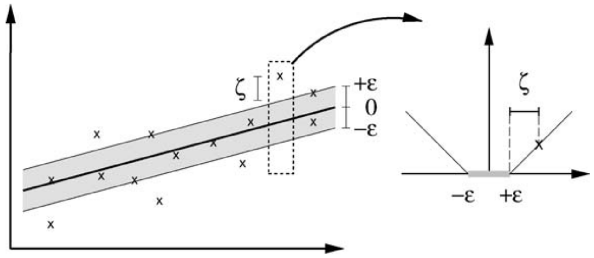
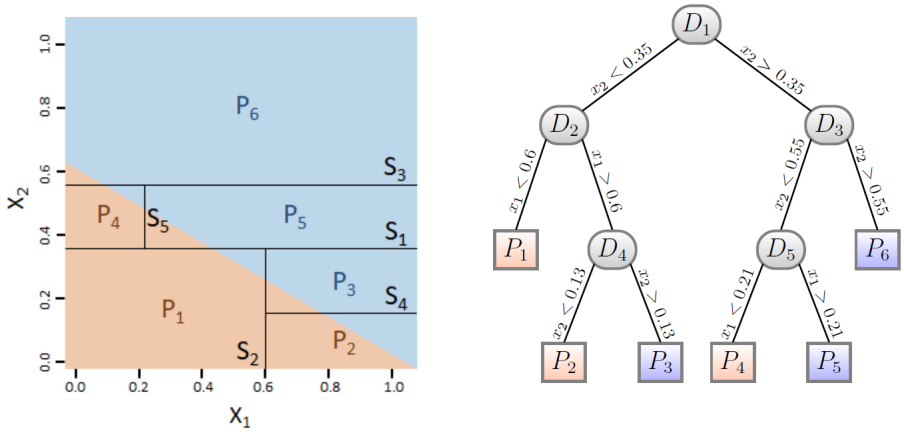


Figure 2.1.2: A linear SVM trained on a small dataset with a linear trend (obtained from [65]).

most often learned by binary decisions in every node, where the leaf nodes state the class to which the data points are assigned.

When learning a decision tree for regression, a slightly different method is applied, in which leaf nodes form partitions of the parameter space of the data vectors $\vec{x}$. The decision tree is formed in such a way that the gained information is maximized at every node, resulting in the same predicted value for configurations that are similar to each other. A clear illustration of a regression tree on a two-dimensional function is given in Fig. 2.1.3.



(a) The shown partitions represent a regression tree that approximates a linear function as indicated by the colors.

(b) The regression tree corresponding to the partitions as shown in (a), where the leaf nodes indicate the designated class for different input values.

Figure 2.1.3: A regression model as approximated by a decision tree. (obtained from [32]) Fig. a denotes the trained partitions in the parameter space, whereas Fig. (b) shows the corresponding decision tree.

Multivariate Adaptive Regression Splines

Multivariate Adaptive Regression Splines (MARS) were introduced by Friedman [30] as a mechanism for dealing with objective functions that exhibit different trends, varying over the parameter space. As opposed to fitting a single model, the MARS algorithm fits multiple models at different domains over the parameter space Ω. This is done by iteratively choosing cutting points in the parameter space, and fitting regression models on the resulting bins in such a way that the error is minimized for all functions. In theory, one could use any regression model to fit a line through the data, but MARS is most often implemented using simple regression models, constants and hinge functions [63]. Fig. 2.1.4 demonstrates how MARS can be effectively applied on data that follows a non-trivial trend, with just one cutting point.

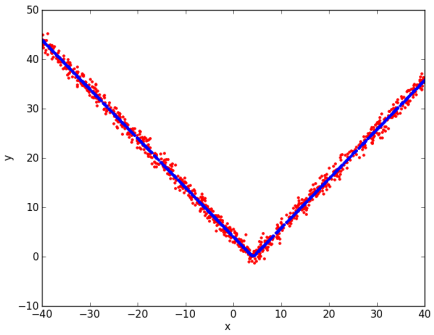


Figure 2.1.4: Multivariate Adaptive Regression Splines fitted on data that follows a non-linear trend (obtained from [63]).

Radial Basis Function Interpolation

A popular method for approximating unknown objective functions is to make use of Radial Basis Functions (RBF's). The main idea of RBF's was first introduced almost fifty years ago by Hardy [35] with the purpose of modelling hills and valleys in cartography. Since then, they have been effectively applied in various ways for a broad range of purposes, making them a popular technique in industrial engineering.

In essence, a radial basis function is any real-valued function $\varphi(||\cdot||)$ measured in terms of the distance from the input data $\vec{x}$ to either the origin, having $\varphi(\vec{x}) = \varphi(||\vec{x}||)$, or some fixed center point c, such that $\varphi(\vec{x}) = \varphi(||\vec{x} - \vec{c}||)$. By interpolating such RBF's, a surrogate model can be created based on available data points. In order to do so, the objective function can be approximated by instantiating multiple radial basis functions and linearly combining their weights such that

$$\hat{f}(\vec{x}) = \sum_{i=1}^{n_c} w_i \varphi(||\vec{x} - \vec{c}^{(i)}||), \tag{2.1.4}$$

in which φ denotes the form of the basis function. $\vec{c}^{(i)}$ is defined as the center of the i^{th} instantiation of this basis function. Several forms of basis functions can be used, e.g. linear, quadratic or Gaussian functions. When creating a surrogate model, the centers of the RBF's are commonly considered to be the values of the available evaluated data points [6]. A major benefit of RBF interpolation is that it is linear in terms of the basis function weights, regardless of the complexity of the used basis function. This makes it relatively easy to approximate the weights w_i by solving the linear system

$$[\mathbf{\Phi}][w] = [\mathbf{f}] \tag{2.1.5}$$

where $\mathbf{\Phi} : \Phi_{ij} = \varphi(||x_i - c_j||), i, j = 1, \ldots, n$ and $\mathbf{f} = f_1, f_2, \ldots, f_n$, determining the weights vector to be:

$$w = \mathbf{\Phi}^{-1}\mathbf{f} \tag{2.1.6}$$

Having the weights w, the corresponding objective value f_* can be computed for any configuration $\vec{x}_*$ by filling in Eq. 2.1.4.

Kriging based Methods

A similar, but slightly more complex model can be made with the Kriging surrogate technique, also called Gaussian Processes. The method was introduced by Krige [44] to improve the accuracy of estimating gold concentration in ore deposits and has been extensively implemented ever since. Although there are multiple realisations of the general Kriging idea, an often used realisation is the *ordinary Kriging* method, which is considered in the following. The ordinary Kriging method is, in essence, quite similar to RBF interpolation with Gaussian basis functions as the basis functions are included in a similar way, here called kernels k. With Kriging, however, the kernels k also account for the the correlation of errors between known points, such that

$$k(\mathrm{x}, \mathrm{x}') = \exp\left(-\sum_{i=1}^{m} w_i |x_i - x_i'|^{p_i} \right) \tag{2.1.7}$$

when using a Gaussian kernel. Here, x_i and x_i' denote two points in the vector $\vec{x}$. In addition to the distance between points x_i and x_i', the correlations depend on the correlation weight parameter w_i and the parameter p_i which denotes the penalization of the distance $|x_i - x_i'|$. By calculating the kernels for all pairwise correlations of the n known function values $\vec{y}$ at configurations $\vec{x}$, a correlation matrix $\mathbf{K}$ can be created:

$$\mathbf{K} = \begin{pmatrix} k(\mathrm{x}^1, \mathrm{x}^1) & \cdots & k(\mathrm{x}^1, \mathrm{x}^\mathrm{n}) \\ \vdots & \ddots & \vdots \\ k(\mathrm{x}^\mathrm{n}, \mathrm{x}^1) & \cdots & k(\mathrm{x}^\mathrm{n}, \mathrm{x}^\mathrm{n}) \end{pmatrix}$$

Subsequently, **K** can be plugged into the probability density function of a multivariate Gaussian distribution, after which the parameters w and p can be estimated by calculating the Maximum Likelihood Estimation (MLE) with respect to y. Doing so, a model is estimated for which the likelihood of the known objective values is maximized. To stay within the scope of this thesis, the complete mathematical derivation of the model is omitted here, but for an extensive overview of the Kriging method we refer to [39] or [6]. In general, Kriging performs very well on many different tasks as it provides local error approximation for new points, but quickly becomes computationally expensive if the magnitude of data increases.

Additional Surrogate Models

In addition to the surrogate models that were discussed so far, it is, in principle, possible to use any regression (or even classification) model to approximate an objective function. Models that, among others, often appear in surrogate literature are polynomial response surfaces, artificial neural networks and basic regression models. However, these are left out of the current thesis as the aforementioned models are expected to capture a sufficiently broad range of function behaviors in order to make diverse ensemble models. Even though neural networks could still add valuable information, it is chosen to leave them out because of their need for large quantities of data in order to perform effectively, which is not available for problems within the scope of this project.

Furthermore, it is worth mentioning that different surrogate models perform very differently on different objective functions, because of the large variation in the forms and shapes the functions can represent [58]. The algorithm that is proposed in this thesis is, however, not bound to specific surrogate techniques and is expected to perform well with any (diverse) collection of surrogate models.

2.1.2 Surrogate Model Handling

The process of fitting and updating surrogate models for optimization can be executed in multiple ways. For instance, in a single evaluation framework, one single surrogate model is chosen based on some criterion, which is fitted to the data in order to obtain a prediction of the objective function. Subsequently, a multi-evaluation framework fits multiple surrogate models to the data, out of which one model with the best fitness is used to predict the objective function. Here, the focus lies on the model combination (ensemble) framework, in which multiple surrogate models are fitted to the data, after which the predictive information of (a selection of) the models is combined in order to obtain an accurate prediction of the objective function.

2.1.3 Efficient Global Optimization

In general, a small amount of initial evaluated data points is not sufficient to obtain a good representation of the overall objective landscape. Therefore, new points have to be sampled to update the surrogate and increase its performance. So, in addition to choosing which (combination of) surrogate models to use for approximating the objective function, a method is required to determine the way in which the selection of new points for evaluation takes place. As the goal is to approximate the optimum of an objective function with as little expensive function evaluations as possible, a method has to be used accordingly.

A widely used and extensively studied algorithm for efficiently optimizing expensive black-box functions is the Efficient Global Optimization (EGO) algorithm as introduced by Jones et al. [42]. The EGO algorithm heavily exploits the proposed surrogate model by sequentially choosing new candidate points for evaluation, based on two main features of the surrogate model. Namely, new candidate points are selected by addressing the prediction of the model at a given point, as well as the uncertainty about the prediction at that point. By considering both of these features, the algorithm provides an autonomous mechanism for balancing between exploration and exploitation of the search space. The EGO algorithm still approximates an expensive objective function by fitting a Kriging model as introduced in Section 2.1. However, in addition to just taking into account the response surface of the surrogate model, the original EGO algorithm introduces an *infill-criterion* which addresses the uncertainty of the predicted response surface as described in Algorithm 1, as obtained from [73].

Algorithm 1 Efficient Global Optimization

1: Fit a Kriging regression model $\hat{f}$ on the initial data points $\mathbf{X}, \mathbf{y}$.
2: **while** stopping criteria not fulfilled **do**
3: Find global optimum of infill-criterion:

$$\mathrm{x}^* := argmax_{\mathrm{x}} EI(\mathrm{x})$$

4: Evaluate $\mathrm{x}^* : y^* = y((x)^*)$ and append x^*, y^* to $\mathbf{X}, \mathbf{y}$
5: Re-estimate model $\hat{f}$
6: **end while**

As shown in Algorithm 1, the original EGO algorithm uses the Expected Improvement (EI) criterion, which computes the expected amount of possible improvement of new points over the current optimal solution. As most of the surrogate models lack an internal measure of variance, EI was originally only proposed in combination with a Kriging model [42]. However, in addition to EI, various other infill-criteria and uncertainty measures have been proposed,

which will be further discussed in section 3.2.

2.2 Multi-Objective Optimization

In this thesis, the focus lies with the simultaneous optimization of multiple objective functions. So-called multi-objective optimization problems (MOPs) as such are mathematically defined as

$$\min_{\vec{x}} \vec{f}(\vec{x}) \text{ , where } \vec{f}(\vec{x}) = [f_1(\vec{x}), \ldots, f_m(\vec{x})] \tag{2.2.1}$$

in which $\vec{f}$ is a collection of m objective functions, where $\vec{x} = [x_1, \ldots, x_n]$ is a solution on n independent variables in the feasible region $\Omega \subseteq \mathbb{R}^n$. In addition to single-objective optimization, the simultaneous optimization of multiple objective functions brings new challenges and dependencies that have to be taken into account. Ideally, Equation 2.2.1 is solved by finding the Utopian solution $\vec{x}^*$, which achieves the minimal value for all of the m objectives. Unfortunately, it is often impossible to find such a perfect solution as the objective functions in most MOPs are at least partly conflicting.

Two objective functions f_i and f_j are conflicting if a solution $\vec{x}$ achieves the optimal value on one of the objective functions, while generating a sub-optimal value for the other. In that case, finding a better value for the one function always happens at the expense of obtaining a worse value on the other objective function.

2.2.1 Pareto-Efficient Solutions

Assuming that a solution $\vec{x}^*$ does not exist where a minimum value is attained for all functions $f_1, \ldots, f_m$, minimization is defined with respect to partial orders. The goal is then, to find solutions for which one objective can not be improved without deteriorating on another objective function. Such feasible, non-dominated solutions $\vec{x}' \in \Omega$ are called *Pareto optimal* or *Pareto-efficient* solutions to MOPs as stated in Equation 2.2.1, if and only if there does not exist a solution $\vec{x}$ in Ω such that $f(\vec{x}) \leq f(\vec{x}')$. The principle of Pareto-optimality originates from economics, first described by Vilfred Pareto, according to [26]. Following this principle, the goal in multi-objective optimization is to find a set of solutions for an MOP, rather than one single point. Such a set of partially ordered (sub-)optimal solutions is called a Pareto-optimal set $\mathcal{P}$, formally defined as

$$\mathcal{P} := \{\vec{x} \in \Omega \mid \nexists \vec{x}' \in \Omega : \vec{f}(\vec{x}') \preceq \vec{f}(\vec{x})\} \tag{2.2.2}$$

where $\preceq$ indicates Pareto-dominance. A solution $\vec{x}$ dominates another solution $\vec{x}'$ if and only if

$$\forall_i (f_i(\vec{x}) \leq f_i(\vec{x}')) \text{ and } \exists f_i(\vec{x}) < f_i(\vec{x}'), \quad i = 1, \ldots, m \tag{2.2.3}$$

The set of solutions $\mathcal{P}$ together form the *Pareto-front*, as illustrated in Fig. 2.2.1.

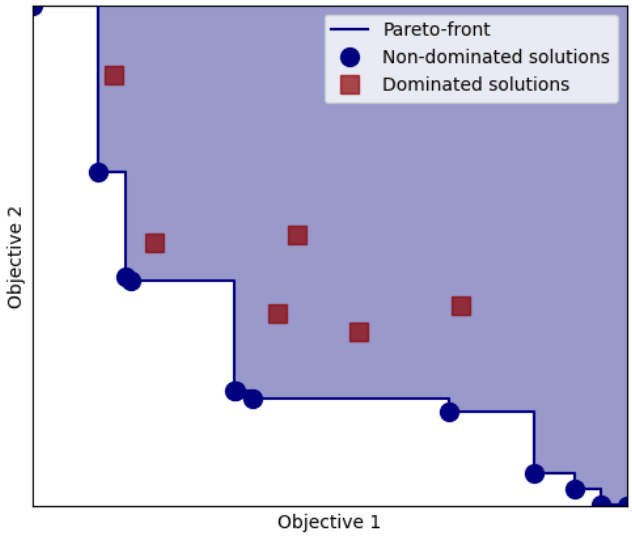


Figure 2.2.1: Illustration of a Pareto-front and related concepts regarding dominance.

2.2.2 Optimization Approaches

Even though multi-objective optimization processes provide a set of Pareto-efficient solutions, often a single final solution is desired for most practical MOPs. Therefore, solution processes are generally concerned with a human *decision maker* who determines the final, most preferred solution. Generally, this decision maker is an expert in the field of the MOP who interprets the solutions using knowledge about the objectives [14]. In general, there are three frameworks in which a decision maker can be included in the optimization process, consisting of a priori, a posteriori and no-preference methods [14].

A Priori Methods

In a priori optimization methods, the decision maker sheds his light on the MOP in advance of the optimization process. Based on knowledge about the various objectives, the decision maker states which of the objective functions might be more important to minimize. Doing so, the decision maker provides a starting point for optimization by, for instance, stating which weights to

give to the objectives. The optimization process can then be performed according to a prior framework.

A Posteriori Methods

Another, very common approach to MOP solving is to consult the decision maker after the optimization process has been finished. In this case, the decision maker is provided with a set of Pareto-optimal solutions, out of which one solution has to be chosen.

No-preference Methods

When little or no information about the objective functions is available, a common approach is to perform the optimization process without further contemplation by a decision maker. In general, no-preference methods are likely to propose solutions that are concentrated somewhere close the knee point of the Pareto front, as no special expectations point the process into a certain specified direction.

Whenever the input of the decision maker is included one single time in the optimization process, it is called a *non-interactive* approach. If the knowledge of a decision maker is consulted multiple times during the optimization process, it is referred to as an *interactive* approach. Both [27] and [14] provide an clear and extensive overview on multi-objective optimization approaches.

–Every new beginning comes from some other beginning's end.

Lucius Annaeus Seneca

3

Related Literature

In the following, a thorough overview of research related to the present work is provided. A large body of academic studies has already been dedicated to ensembles of surrogates, as well as multi-objective optimization. As quite a substantial amount of techniques is incorporated in this thesis, this chapter briefly covers the most relevant approaches, as well as the algorithms, techniques and frameworks that lie at the foundation of the proposed method.

3.1 Ensembles of Surrogate Models

Combining the output of multiple surrogate models into an ensemble has repeatedly been shown to be beneficial to optimization processes in both practical applications and artificial test functions. For instance, Roy and Datta [62] showed that ensembles of surrogate models improved the development of saltwater intrusion management strategies in coastal aquifers. Similarly, Zhang et al. [84] show that an adaptive ensemble of individual models outperforms single surrogate models when optimizing the amount of power generated by a wind farm as function of the distribution of wind turbines across the farm.

Also, Acar and Rais-Rohani [2] succeeded to significantly lower car crash impact based on the shape and wall thickness of car designs with the use of ensembles of surrogate models, as compared to using individual models.

It is needless to say that the use of ensembles of surrogate models is not

limited to the aforementioned cases. A summary of further applications of surrogate ensembles can be found in [39] and [78].

3.1.1 Classical Ensemble Techniques

The automatic selection and combination of surrogate models was initially performed by means of simple techniques. Some of the classical, most basic ensemble techniques are Greedy selection [31], Ranking [31], Bagging [16] and Boosting [29]. These techniques lay at the foundation of ensemble surrogate modelling literature, as they were among the first methods to combine the output from multiple surrogate models. As the current study is focused at more sophisticated ways of ensemble learning, these methods are not further discussed in detail here, but a clear description and comparison of these basic ensemble methods are provided by [31] and [32].

3.1.2 Weighted Averaging of Surrogate Models

A more advanced and very widely used method for ensemble creation is to assign weights to individual surrogate models based on their performance [33]. As a first rule to ensure the creation of a valid ensemble, it is necessary that the combined weights sum up to one. In this way, the output of multiple surrogate models is combined into a new ensemble model. However, the way in which these weights are found and assigned varies enormously between methods, and the performance of these methods heavily depends on the given optimization task [39]. The construction of weighted averaging ensembles generally seems to be consisting of three main components, namely:

1. The used accuracy measure for the performance evaluation of individual models.
2. The chosen resampling method for (ensemble) model evaluation.
3. The scope of the weight factor selection.

Over the following three sections, the choices that can be made regarding these components are summarized. Also, an outline is given on how the most influential methods applied weighted averaging of surrogate models in order to create ensembles.

Accuracy Measures

When generating weighted surrogate model ensembles, weights are assigned based on the contributions of individual models [32]. In general, surrogate models that perform better are given higher weights, and the weights for the worst performing models are reduced to zero. As the weights are based on

the individual model performance, the composition of the found ensembles strongly depends on the choice of performance metric.

The vast majority of ensemble methods make use of basic accuracy measures to gain insight in the performance of the surrogate models. For example, the (General) Root Mean Squared Error (RMSE) is often used to obtain the model accuracies [32, 33, 74]. Furthermore, it has been shown that, among other metrics, the Relative Average Absolute Error (RAAE) [40], Relative Maximum Absolute Error (RMAE) [40] and R-square [83] metrics can provide meaningful insights into model performance [39]. In [40], Chen and Simpson provide a thorough comparison between accuracy measures, and conclude that the usefulness of certain performance measures highly depends on the used surrogate models in question, as well as the given test problems. Note that these accuracy measures can be applied applied both locally and globally, as further specified hereafter in Section 3.1.2. Also, the same metrics can generally be used to obtain insight in the performance of the obtained ensemble models, which should not be confused with the measure used for finding new points in the EGO framework, as further discussed in Section 3.2.

Resampling Methods

As the goal of ensemble surrogate modelling is to generate an ensemble that predicts the real, expensive objective function as accurately as possible, it is of crucial importance that the accuracy of (individual) models is assessed in a valid way. To ensure the creation of valid and robust ensembles, methods for model validation call for reliable resampling strategies. Bischl et al. [8] show that making use of resampling strategies like cross-validation, bootstrapping and sub-sampling significantly improves the accuracy of meta-models, as it prevents overfitting on the often limited amount of data-points. Additionally, it was shown that, for a limited amount of base models, evaluating ensemble performance by performing leave-one-out cross-validation (LOO-CV) can be beneficial, making sure that the available data is used to the biggest extend [32, 58]. However, LOO-CV quickly becomes too computationally expensive if applied to a larger set of base models [32].

In order to automatically filter out base models with large errors, Song et al. [67] evaluate models by introducing their normalized cross-validation error, which gives a penalty to models with high cross-validated errors.

Weight Factor Selection

The assignment of weights to individual surrogate models can either be carried out globally, or locally for different partitions of the design space. In the literature, weighted ensemble methods are roughly divided into two categories: Global weighted averaging, and point-wise weighted averaging of models [39].

Global Weighted Averaging

In global weighted averaging methods, the complete design space is considered altogether in the calculation of individual model performances, and the output of the models is combined using the same weight across the whole input space.

As a first attempt at weighted model combination, Goel et al. [33] combined polynomial regression, Kriging and RBF models into ensembles by globally weighting the models based on their performances to approximate expensive objective functions. This highly influential study has been used as a starting point in the creation of many similar methods.

For example, Ye and Pan [79] applied the same technique in combination with clustering methods to improve efficiency. By reducing the design space, they were able to generate ensembles with similar performances, while requiring less computational effort.

In [2], Acar et al. further exploited the global weighted averaging method by introducing an optimization procedure on the weight factors, such that a weighted ensemble with minimal global error is found.

Similarly, Zhou et al. [87] managed to improve on existing methods by obtaining weight factors by means of a recursive process, in which the weights are updated in each iteration until an ensemble with desirable prediction accuracy is found.

Shi et al. [64] use the covariance matrix of residuals as a base for weighting several radial basis functions in an efficient manner. In order to do so, they use the covariance matrix weighting method as proposed by Bishop [9] and later improved by Viana et al. [74].

Friese et al. [32] performed an exhaustive search over model weights, showing that any ensemble with positive weights can, in terms of RMSE, never perform worse than the worst performing base model, and always has a chance to perform better than either of the individual base models due to the convex nature of the weight combination. Moreover, they investigated a way to efficiently scale up the amount of included models in an ensemble by performing an evolutionary search over the model weights. As they showed that a convex linear combination of model weights can improve on individual models, this study is used as a reference point for the method proposed in this thesis.

Point-wise Weighted Ensembles

More recent approaches have tried to create ensembles based on local accuracy measures, where the model weights are assigned differently across the input space. By doing so, ensembles are better capable of capturing local trends in specified regions of the design space.

For instance, Acar [1] used the cross-validated prediction variance as a local accuracy measure to indicate individual model performance. Model

weights are, however, still fixed over the entire input space, even though the model performances are based on a local accuracy measure.

Laying more focus on local trends, Zhang et al. [84] proposed an adaptive hybrid ensemble technique that combines favorable characteristics of different surrogate models by formulating trust regions based on distance between data points, and assigning weights based on these regions.

In a similar way focusing on local variations, Lee and Choi [46] find the ensemble with optimal weights by only addressing the k nearest points to the prediction point of interest, instead of building a global ensemble model. Song et al. [67] further built on [84] by filtering out badly performing base models and allocating different weights to every data point in the input vector, based on probabilities derived from a Gaussian Process. The accuracy and robustness of this method improved remarkably with respect to individual models, as well as multiple benchmark ensemble models.
A more straightforward point-wise ensemble method was proposed by Yin et al. [81] who divide the design space into multiple pre-defined subdomains and assign a set of optimal weights to all of the domains, while ensuring a smooth transition between the subdomain ensembles.

In general, point-wise weighted ensembles of surrogate models seem to be more accurate than global average weighting methods, as they mainly capture local trends in the objective function. A drawback, however, is that the ensemble models may perform poorly if the local error measure at a certain point is inaccurate, even though the local error measure at other points is accurate. Potentially, this inaccurate region could affect the overall landscape of the predicted function [80]. To overcome this problem, [83] and [80] combined local error measures with global error measures, generating ensembles that cover both local and global trends.

3.2 Uncertainty Quantification

In the EGO framework, new points are iteratively chosen for evaluation based on their predicted values. In addition to only using the values of the predictions, some applications require that the uncertainty of these predictions is also taken into account, e.g. in terms of variances, standard deviations, or confidence intervals. The infill criterion at predicted locations is then a combined measure of the predicted value and the uncertainty of the prediction. Popular infill criteria, also called acquisition functions, are the *Expected Improvement* [42], *Lower Confidence Bound* [18], and *Probability of Improvement* [71], which all take into account the uncertainty of predictions.

Some regression models, like Kriging/GP, automatically address the confidence of predictions as they also provide an estimation of the prediction variance [44, 66]. In this case, the infill criterion can be calculated without a

significant amount of extra effort.
Unfortunately, the majority of regression models is not equipped with built-in variance estimation properties as such, which calls for the implementation of an external uncertainty quantification (UQ) measure in order to be adopted to the EGO framework.

A more generic method which can be applied to any kind of regression model is to fit the model multiple times by bootstrapping, i.e., random sampling with replacement [68]. Doing so, additional variance metrics can be calculated from the set of models to obtain an indication of prediction uncertainty. Although this method generally works well, it is computationally very expensive to apply in the EGO framework.

Van Stein et al. [73] provide a fine UQ measure that operates independently of surrogate modeling assumptions by addressing the empirical prediction error at a given point, as well as the variability of the k nearest neighbours based on the euclidean distance to these neighbouring points.

On the contrary, Rehbach et al. [60] recently provided evidence that in some cases, using infill criteria with uncertainty quantification actually worsens optimization performance in comparison to just using the predicted value as infill criterion. In general, they concluded that exploration loses importance in favor of exploitation when only a small evaluation budget is available, especially with problems of higher dimensionality.

3.3 Model-Based Multi-Objective Optimization

In the multi-objective optimization setting, the role of surrogate models in the optimization process is not always unambiguously [4]. For example, Loshchilov et al. [51] trained surrogate models to distinguish dominated solutions from non-dominated solutions. Likewise, Bandaru et al. [7] proposed to use a multi-class surrogate classification model to determine the dominance information between two candidate points. Additionaly, surrogate models have been used to approximate the increase in hypervolume (further explained in Chapter 4) of new proposed individuals [55].

In the present thesis, multiple objective functions are approximated separately by surrogate models. By simultaneously minimizing these surrogate models, the goal is to find a collection of optimal points, as described in Section 2.2.

Horn et al.[37] provide a detailed taxonomy for Model-Based Multi-Objective Optimization (MBMO), narrowing a broad range of algorithms down to three main categories, namely:

- Scalarization-based approaches
- Pareto-based approaches

- Direct indicator-based approaches

With regard to these categories, the most influential state-of-the-art MBMO algorithms are briefly covered over the following sections. In addition, a comprehensive flow chart of the possibilities in MBMO is provided in Fig. 3.3.1.

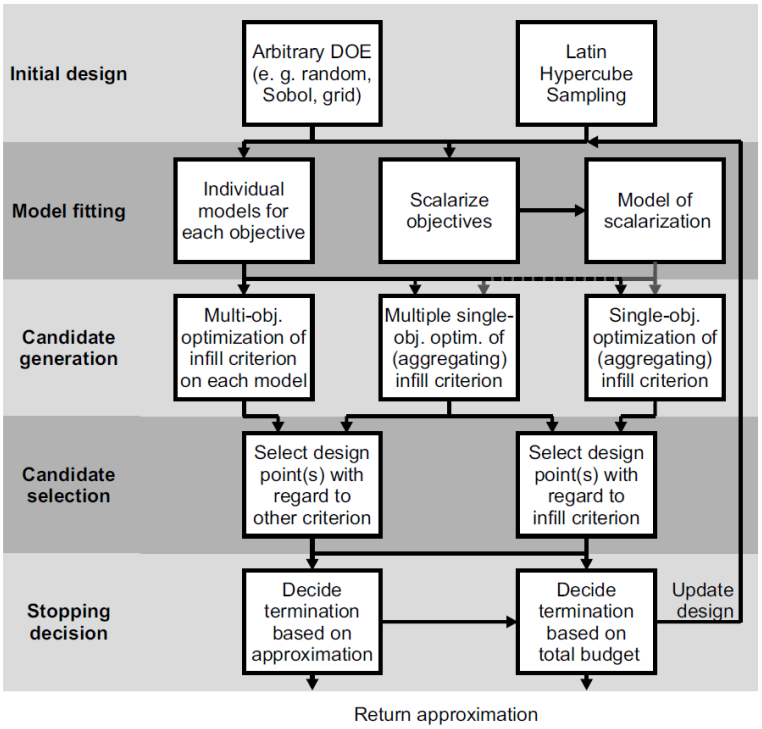


Figure 3.3.1: Stages and challenges in MBMO (obtained from [37]).

3.3.1 Scalarization-Based Approaches

Instead of trying to minimize multiple objectives sequentially, scalarization-based approaches recast a multi-objective problem as a single objective problem. By applying an aggregate function to the surrogates of multiple objectives, the problem can thus be solved by means of a standard single optimization algorithm [53]. A popular scalarization-based approach as such is the weighted sum method, where weights are assigned to every objective to indicate to which extend they contribute to the solution. For instance, the widely used *ParEGO* algorithm as proposed by Knowles [43] randomly selects of set of weights, until either a satisfactory solution has been found or the computation budget has depleted. Note that these weights are not to be confused with the weights given to individual base models in the creation of surrogate ensembles.

Zhang et al. [85] proposed the *MOEA/D-EGO* algorithm, which cuts up the MOP into single objective sub-problems, and searches for points that maximize the combined Expected Improvement values for all of the individual surrogate models.

A drawback for scalarization-based approaches, however, is that they generally just provide a single solution, whereas it might be desirable to obtain a set of Pareto-optimal solutions to choose from.

3.3.2 Pareto-Based Approaches

In Pareto-based approaches, a multi-objective optimization is performed over the infill criteria for all of the objectives. To be more specific, the MOP objectives are not considered as a single scalar, but rather as a vector of objective values. Hence, a collection of Pareto-optimal solutions can be achieved instead of a single solution [37, 41]. In order to find such optimal points, efficient algorithms are used to search the entire design space for points that perform well on all or most of the objectives.

For example, many well-known Pareto-based algorithms, like *MOEA's* [24, 86], *NSGA-II* [21], *NSGA-III* [20] make use of genetic or evolutionary search strategies to perform an efficient search over the design space. Similarly, C-TAEA [48] strives to balance between exploration and exploitation by introducing two competing populations, where the one population greedily searches for optimal solutions, whereas the other population heuristically explores under-represented regions in the design space.

A drawback for Pareto-based approaches is that they often require a lot of redundant evaluations on the surrogates as they mostly consider the complete design space, including regions that lie outside the scope of interest.

3.3.3 Direct Indicator-Based Approaches

To prevent the evaluation of uninteresting points on the surrogate models, indicator-based approaches apply a more efficient and demarcated search. With indicator-based MBMO approaches, the objective functions are approximated with surrogate models as in scalarization and pareto-based approaches. However, instead of maximizing the vector of function values all at once, an internal optimization process is first executed on the separate surrogate models, to find possible 'optimal' data points per objective. Subsequently, for all of these points, the contribution to the approximated Pareto-front is calculated. That is to say, new points are found by maximizing the contribution of a potential solution to the approximated Pareto-front, formulated as a single-criterion optimization problem [37].

The *SMS-EGO* algorithm as proposed Ponweiser [57] evaluates the found approximations directly by generating a collection of minimal values per surrogate function. As this generates a substantial set of candidate points in

the Pareto-hyperspace, the most optimal point can be found by carrying out a maximization process on the additional hyperspace. Only after maximizing the approximated hyperspace, the optimal solution is evaluated on the true objective functions, saving a lot of expensive resources.

Much like the SMS-EGO algorithm, the $\epsilon-EGO$ algorithm [75] introduces an additive ϵ margin, to enforce the search for potential points that have a significant contribution to the approximated Pareto-front. As the performances of the SMS-EGO algorithm have been shown to exceed those of similar algorithms [37], its main framework is adopted in the current work.

3.3.4 Ensemble-Based Multi-Objective Optimization

Even though the literature about surrogate ensembles and model-based multi-objective optimization is quite extensive, the question on how to combine the two topics in a knowledgeable manner has, to the best of my knowledge, rarely been addressed so far.

In [61], Rosales-Perez et al. created hybrid ensembles to predicting multiple objectives by iteratively training new SVMs and adding them to the ensemble. This was shown to be beneficial to the optimization process as ensembles became more accurate as more SVMs were added. Zavoianu et al. [82] created ensembles of various modeling tools to approximate multiple non-linear objectives and concluded that, in many scenarios, ensembles of surrogate models showed better performance than individual surrogate models in the MBMO setting.

–If we can really understand the problem, the answer will come out of it, because the answer is not separate from the problem.

Jiddu Krishnamurti

4

Industrial Design Optimization Problem

Multi-objective optimization problems can differ a lot in terms of input dimensionality or number of objectives and are often presented as black-box optimization problems. Such black-box optimization problems occur in all kinds of applications with different scopes, differing from transportation scheduling [47] to DNA sequencing [38]. In some applications, the number of objectives can run up to more than a hundred objectives.

However, the present work focuses on the optimization of industrial design, in which the amount of objectives is generally somewhere in the range from two to ten [34]. Here, we address the problem of industrial design optimization as an MOP as formulated in Eq. 2.2.1, where the individual objective functions are very expensive to compute, e.g., wind turbine efficiency [58], aerodynamic drag [58] or boat hull stability [19], which are subject to many physical constraints and complex dependencies. Such optimization problems with multiple objectives in the process of industrial engineering are involved with several stages and challenges, which are described in this chapter.

4.1 Obtaining Initial Design Configurations

Before starting an optimization process, it is necessary to obtain a collection of initial design points which serves as a starting point for optimization. In

practice, it is conventional to use former, already evaluated points to start the optimization process. In case that such a set of known points is unavailable, obtaining new initial design is often referred to as Design of Experiments (DoE). DoE is a systematic method to determine the dependence of the output of a process on the affecting factors [5]. Generally, DoE methods are sampling techniques that sample points in the design space based in a certain systematic manner, e.g., random sampling [5], grid sampling [5], Latin Hypercube Sampling (LSH) [52], full factorial sampling [8] and more.

In some cases, additional expert knowledge about the given optimization problem is available, in which the DoE process can be guided by experience.

4.2 Design Evaluation

In industrial engineering, the most time-consuming and expensive part is often the stage of evaluating possible configurations [57]. This evaluation is generally performed by means of either analytical, experimental or computational methods.

4.2.1 Analytical

The most accurate way to evaluate design is by analytical evaluation of a objective function. In this case, a generalized equation for the objective problems is known, which can be solved analytically. However, analytical evaluation is only possible with simple, isolated equations, which are not available in most industrial design applications [28].

4.2.2 Experimental

Another, more expensive way to evaluate industrial design is by creating psychical prototypes, and measuring the objectives in a controlled setting [28]. Even though experimental evaluation generally provides extremely reliable and informative insights, it is infeasible to build a prototype for every iteration. Therefore, experimental evaluation is mostly exclusively conducted in the final design phase.

4.2.3 Computational

Computational evaluation has gained a lot of popularity since computational resources are becoming available at a larger scale. By capturing physical processes and phenomena in computational models containing detailed equations, simulations can be performed to mimic real-world behavior of proposed configurations. For instance, Finite Element Analysis [69] and Computational Fluid Dynamics [28] techniques are used to calculate physical properties of a design without the need for a physical prototype.

Although it is practically impossible to account for every minor external influence, computational evaluation methods can predict physical behavior fairly accurately. Unfortunately, these required calculations are often so advanced that it can still take up to weeks or even months to calculate the performance of one configuration on a single objective function.

4.3 Problem Description

Using either of the aforementioned evaluation methods, practice shows that it is very costly to evaluate a design configuration in case of industrial engineering, especially when it comes to experimental or computational evaluation. Consequently, the main problem is on how to minimize the amount of function evaluations required to obtain satisfactory results. Hence, an algorithm that tackles such an MOP should exploit the known configurations as thoroughly as possible. In principle, the priority is therefore maximizing the prediction accuracy on the individual objectives, and not minimizing the required computation time to propose new data points. Although the problem is scalable, the primary scope of this study is limited to optimization problems with a relatively low amount of objectives.

Furthermore, the behavior of objective landscapes is generally very different for the different objectives in industrial design and little is usually known about the behavior of the landscapes. Accordingly, a second challenge is to include an adaptive/hybrid technique for selecting and combining surrogate models into ensembles in such a way that the objective functions can be approximated well without the need for prior knowledge on the objectives.

Finally, many experimental and computational methods are subject to a certain amount of variability in outcomes, that will say, when provided with the same input, the output can still differ a bit due to the complex nature of the evaluation methods. However, for the sake of experimenting and obtaining conclusive results, the assumption here is that function evaluations are deterministic, reliable and valid, i.e., that it is possible to approximate the objective functions in a deterministic manner.

4.3.1 Hypervolume Indicator Optimization

In practice, when optimizing multiple objectives, the goal is to obtain a Pareto-front that maximizes the amount of gained information (recall Section 2.2. Although there are multiple methods for performance indication, the performance assessment of Pareto-fronts is generally expressed with the Hypervolume Indicator [89].

The Hypervolume indicates the (multi-dimensional) space between the obtained Pareto-front and a chosen reference point. In Figure 2.2.1, the hypervolume is denoted by the surface between the Non-dominated solutions and the upper right corner of the plot. When optimizing more than two

objectives, the hypervolume would denote the hyperspace between a set of solutions and a reference point. Therefore, the main goal in multi-objective optimization is to maximize the Hypervolume indicator by obtaining a well-spread set of solutions with as low values on all objectives as possible.

–I never worry about the problem. I worry about the solution.

Shaquille O'Neal

5

Proposed Solution

In this chapter, the new *Ensemble-based S-Metric Selection Efficient Global Optimization* (E-SMS-EGO) algorithm is introduced, which extends the *SMS-EGO* algorithm by addressing additional predictive power from ensembles of surrogates. The generation of ensembles of surrogate models provides additional insight into the objective functions, which is heavily exploited to force the multi-objective S-Metric Selection optimization process in the right direction for minimization, while still aiming to obtain a Pareto-front with evenly spread points. Consequently, this method can be used to solve expensive multi-objective optimization problems in a highly efficient manner, requiring a minimal amount of expensive function evaluations.

5.1 E-SMS-EGO

The following section provides an extensive overview of the E-SMS-EGO algorithm, which is built out of the following key components: The initial sampling procedure, training the surrogate base models with respect to each objective function, as well as finding the optimal ensemble weights for every objective, followed by the separate minimization of all of the ensemble function predictions to obtain a set of potential points for evaluation, out of which the greatest contributor in terms of Hypervolume addition is chosen for evaluation on the true objective function.

First, a global outline of E-SMS-EGO is given in Algorithm 2, after which the individual parts of the algorithm are decomposed and described in the subsections that follow. Please note that Sections 5.1.2 through 5.1.4 are executed in parallel for the different objective functions, and integrated into the multi-objective setting in Section 5.1.5

Algorithm 2 E-SMS-EGO

Input: $\vec{x}$: initial sample; $\vec{b}$: base models; $\vec{f}$: objective functions
Output: Pareto-optimal solutions

$Y \coloneqq$ evaluate $\vec{x}$ on all functions $f_1, \ldots, f_m$
while $EvalBudget > 0$ **do**
  for all obj in $f_1, \ldots, f_m$ **do**
    $\vec{w}^*_{obj} \coloneqq FindOptWeights(\vec{x}, \vec{y}_{obj}, folds, \vec{b})$ ▷ Optimal Weights
    $\vec{tm}_{obj} \coloneqq TrainModels(\vec{x}, \vec{y}_{obj}, \vec{b})$ ▷ Trained Models
    $\vec{x}^{'}_{obj} \coloneqq MinimizeEns(\vec{tm}_{obj}, \vec{w}^*_{obj}, iter)$ ▷ Potential points
    $\vec{x}^{'}.append(\vec{x}^{'}_{obj})$
  for all obj in $f_1, \ldots, f_m$ **do**
    $Y^{'}_{obj}.append(EnsemblePrediction(\vec{x}^{'}, \vec{w}^*_{obj}, \vec{tm}_{obj}))$
  $x^* = MaxHypervolume(\vec{x}^{'}, Y^{'})$ ▷ Greatest Contributor
  $\vec{x}.append(x^*)$
  $Y.append(\{f_1(x^*), \ldots, f_m(x^*)\})$
  $EvalBudget \coloneqq EvalBudget - 1$
end while
return $ParetoFront(\vec{y})$

5.1.1 Initial Sampling

Before starting the optimization process, an initial set of data points is obtained by means of a sampling procedure. *E-SMS-EGO* uses the optimized Latin Hypercube Sampling method [52], which provides a sample of data points that are equally distributed across the search domain. By dividing the range of each of the n input variables into N equally probable intervals, N samples are placed in the search domain, ensuring only one sample per interval for every variable, as illustrated in Figure 5.1.1. By using LHS, the amount of information that the surrogate models can derive from the sample is maximized. After the LHS sampling step, the found data points are evaluated on the objective functions to obtain the corresponding objective values.

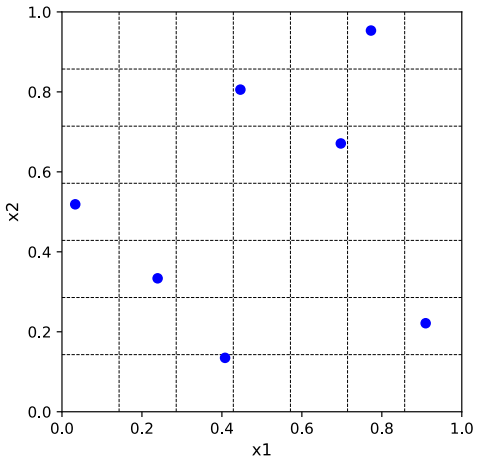


Figure 5.1.1: Example of a sample generated with LHS with $n = 2, N = 7$ on a normalized domain for both variables.

5.1.2 Finding Optimal Ensemble Weights

Next, the goal is to create a well-performing and robust ensemble for every objective function. Therefore, the second step involves finding the optimal linear combination of weights per objective function. In *E-SMS-EGO*, the optimal weights are found per objective by means of 10-fold cross validation (line 4), largely based on the linear combination method proposed by Friese et al. [32].

First, all possible weight combinations are obtained by calculating p possible integer partitions with size k (the amount of base models) out of the integer 10. After dividing these partitions by 10, this results in the weight matrix W:

$$W_{p,k} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0_{1,k} \\ 0.9 & 0.1 & 0 & \cdots & 0 & 0_{2,k} \\ 0.8 & 0.1 & 0.1 & \cdots & 0 & 0_{3,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0.1 & 0.9_{p-1,k} \\ 0_{p,1} & 0_{p,2} & 0_{p,3} & \cdots & 0_{p,k-1} & 1_{p,k} \end{pmatrix}$$

where:

p: number of possible weight combinations.

k: amount of base models.

Out of this collection of possible weight combinations, the optimal weights are found separately for all of the objective functions by executing the following steps according to the 10-fold cross validation procedure:

1. First, the base models are trained on the training partition of the cross-validation fold.
2. Subsequently, all of the trained models are fitted separately to predict the objective value of the configurations in the test partition, resulting in a prediction matrix of size k times the amount of test points in the fold.
3. Next, an ensemble prediction is determined by calculating the weighted average using every possible weight combination, i.e., the rows in matrix W, as in Equation 5.1.1.
4. Finally, the MSE is calculated between the ensemble predictions and the corresponding objective function values as obtained in the initial sampling procedure.

This results in ten MSE scores for all of the possible weight combinations, which are averaged to get the cross-validated MSE score per weight vector. As a result, the combination of weights with the minimal corresponding cross-validated MSE value is selected as optimal weight vector $\vec{w}^*_{obj}$ to create the ensemble for approximating the objective function in question.

5.1.3 Surrogate Model Training

As a next step, the k base models are trained on the full data set in line 5 to obtain an as accurate prediction per base model as possible. By training the models on the full data set after finding the optimal weights, the model predictions can be used with these optimal weights to obtain accurate and robust ensemble predictions in the next step.

5.1.4 Minimizing Ensemble Predictions

Subsequently, a minimization process is carried out in line 6 to obtain the minimum of the ensemble. This is done by predicting the objective function value with the ensemble consisting of the trained base models in combination with the optimal weights that were found earlier. With the intention to encourage exploration and avoid getting stuck in local minima, the ensemble prediction is made up of two components, the predicted value (Equation 5.1.1) and the uncertainty quantification component (Equation 5.1.2), as introduced in Section 3.2.

Predicted Value

The Ensemble Predicted Value (EPV) is computed as follows for a given objective function:

$$EPV(\vec{x}) = \sum_{i=0}^{N} \vec{w}_i \cdot \hat{f}_i(\vec{x})^T, \tag{5.1.1}$$

where:

$\hat{f}_i$: an individual base model.

$\vec{w}_i$: the vector of best weights of the corresponding objective function.

Uncertainty Quantification

In addition to just using the EPV, the final prediction is calculated by introducing an infill criterion. Section 6.3 shows multiple experiments that were conducted to find out which infill-criterion is most beneficial to the proposed algorithm. The k-NN variance infill criterion [73] (as formulated in Equation 5.1.2) was shown to be most beneficial, and is therefore included in the final version of the proposed algorithm.

$$\widehat{U}_{k-NN} = \frac{\sum\limits_{i \in N(\vec{x})} w_i^k \left| EPV(\vec{x}) - y_i \right|}{\sum\limits_{i \in N(\vec{x})} w_i^k} + \frac{\min\limits_{i \in N(\vec{x})} d(\vec{x}_i, \vec{x})}{\max\limits_{\vec{x}_i, \vec{x}_j \in \chi} d(\vec{x}_i, \vec{x})} \hat{\sigma}. \tag{5.1.2}$$

where

$w_i = 1 - \frac{d(\vec{x}_i, \vec{x})}{\sum\limits_{i \in N(\vec{x})} d(\vec{x}_i, \vec{x})}$, $\hat{\sigma} = \sqrt{\mathrm{Var}\left[\{y_i\}_{i \in N(\vec{x})} \cup \{\hat{f}(\vec{x})\}\right]}$.

Resulting the final ensemble prediction per objective function including the uncertainty measure to be:

$$KPV = EPV(\vec{x}) - \widehat{U}_{k-NN} \tag{5.1.3}$$

The actual minimization process on the ensemble prediction function per objective is then done by repeatedly picking a random data point in the search domain and running a simple *scikit-learn* minimization function over it.

5.1.5 S-Metric Selection

As the minimization process in step 5.1.4 is repeated multiple times, a collection of potential points in the search domain is obtained for all of the objectives in parallel. Subsequently, these points are again evaluated in lines 8, 9 using the composed ensemble for all of the objectives to obtain a predicted value per objective. These predictions are then in turn used to estimate a hypervolume score for all potential points, out of which the point with the

biggest estimated hypervolume (greatest contributor) is chosen for evaluation on the actual objective function in line 10. This model-assisted S-Metric selection approach as provided by Ponweiser et al. [57] allows for an accurate identification of new data points by making exhaustive use of the available data, and therefore minimizing the amount of function evaluations as desired.

Moreover, apart from scaling nicely with the amount of objectives [76], the S-Metric selection does not require normalization of objective spaces, which makes it -along with some other nice theoretical properties [88]- ideal to extend with optimally weighted ensemble surrogate models.

The true method of knowledge is experiment.

William Blake

6

Experimental Evaluation

This chapter describes a selection of experiments that were conducted in order to evaluate the proposed solution. Certain important choices for the final proposed framework were based on insights gained from the experiments described in sections 6.2 and 6.3. In addition, the Section 6.4 covers a comprehensive comparison to corresponding state-of-the-art multi-objective optimization techniques.

6.1 General Experimental Setup

All of the experiments described below were performed in an isolated Anaconda environment, running Python 3.8.3 on the *mithril* server in the LIACS Data Science Lab. This machine has 1 Terabyte RAM memory and 64 Intel Xeon E5-4667v3 CPUs @ 2.00 GHz (128 threads). Further implementation details and source code are available at https://github.com/Gitdeon/E-SMS-EGO.

6.2 Linearly Weighted Ensembles

As a starting point for the analysis of the proposed framework, some experiments were first conducted in a single objective optimization setting to test the general performance of linearly weighted ensembles. To show the effectiveness of linearly weighted ensembles of surrogate models, the best performing ensembles were compared to their individual base models under

multiple experimental conditions.

6.2.1 Experimental Setup

Surrogate models and ensemble generation

In the first experiment, an ensemble was created by linearly combining the following five base models as introduced in Chapter 2:

- Kriging/Gaussian Process Regressor (GPR)
- Radial Basis Function (RBF)
- Decision Tree (DT)
- Multi-variate Adaptive Regression Splines (MARS)
- Support Vector Regression (SVR)

To ensure a fair comparison and similar implementation, all of the models were obtained from the scikit-learn package [56], a highly renowned package with a wide variety of implemented methods for pre-processing, model building, model evaluation and several other tasks. Initially, the standard hyper-parameter values were adopted for all of the base models. All base models were trained using 10-fold cross-validation on a small, two dimensional data set with 20 random samples, and ensembles were generated by performing a 10-fold cross-validated exhaustive search over all the ensemble weights according to the method described in section 5.1.2.

Test Functions

The experiments were performed on four single objective test functions with varying landscape properties and different inputs to obtain diverse data and ensure reliable results. Included are the two-dimensional Ackley [3] and Branin [13] functions, as well as the Himmelblau [36] and Sphere [12] functions which take in inputs of variable dimensionality.

Test function 1: Ackley function

$$\begin{aligned} f(x_1, x_2) = -20 \exp\left(-0.2\sqrt{0.5\left(x_1^2 + x_2^2\right)}\right) \\ - \exp[0.5(\cos(2\pi x_1) \\ + \cos(2\pi x_2))] + e + 20, \\ \text{where } x_1 \in [-5, 5], x_2 \in [-5, 5] \end{aligned} \tag{6.2.1}$$

Test function 2: Branin/ Branin-Hoo function

$$f(x_1, x_2) = \Big(x_2 - bx_1^2 + cx_1 - 6\Big)^2 + 10(1-t)cos(x_1) + 10, \quad \text{where } b = 5.1/(4\pi^2), c = 5/\pi, \text{ and } x_1 \in [-5, 10], x_2 \in [0, 15], \tag{6.2.2}$$

Test function 3: Himmelblau function

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2, \quad \text{where } x_1 \in [-5, 5], x_2 \in [-5, 5] \tag{6.2.3}$$

Test function 4: Sphere function

$$f(\vec{x}) = f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} x_i^2, \quad \text{where } x_i \in [-5.12, 5.12] \ \forall i \in [1, \ldots, n] \tag{6.2.4}$$

In the experiments, personal implementations of the Ackley and Branin functions were used. The Himmelblau and Sphere functions were adopted from the MF2 (available at https://github.com/sjvrijn/mf2) and SMT [12] Python packages respectively. Two dimension were used for all objective functions, with ranges as described in Equations 6.2.1 to 6.2.4.

Evaluation Metrics

In an attempt to present a complete comparison, the performance of the ensembles were compared to the base models by means of multiple evaluation metrics. Model performances are expressed in terms of the Mean Absolute Error (MAE), Maximum Error (ME) and Median Absolute Error (Med):

$$\text{MAE} = \frac{\sum_{j=1}^{n} |e_j|}{n}, \tag{6.2.5}$$

$$\text{ME} = \max_{j=1...N} |e_j|, \tag{6.2.6}$$

$$\text{Med} = Md(|E|), \tag{6.2.7}$$

The models are evaluated by means of these three measures as they cover diverse aspects of model performance. Both the MAE and the Med cover the general performance over the global search space. A low MAE value means that a model performs well over the whole search space, but can be influenced by some poorly predicted outliers. As the Med is unaffected by such tail prediction errors, it is insensitive to outliers, indicating a more

robust performance. On the other hand, to gain insight in the outlying points, the Maximum (Absolute) Error shows the maximum deviation of the points at which a model performs worst. Therefore, by combining the information gained from these three measures, the model performances can be summarized in a complete manner.

6.2.2 Results

All of the experiments show results in favor of the ensemble technique using linear weights. Table 6.2.1 sums up the average results over fifty repetitions in all of the experimental conditions, showing that the ensemble approach outperforms all of the base models in every single configuration. Likewise, Figure 6.2.1 shows an example of the boxplots on the model performances in terms of MAE after another fifty generations of the ensemble creation process. These results are as expected, as the chosen weight configuration for the final ensemble is always the one with the best performances, even if a single base model does perform better than any other weight configuration, in which case the weight for the best performing base model is set to one, whereas the other model weights are reduced to zero. Also, the spread in outcomes seems to be a bit smaller in case of the ensemble models, making it a more stable and robust method for surrogate prediction.

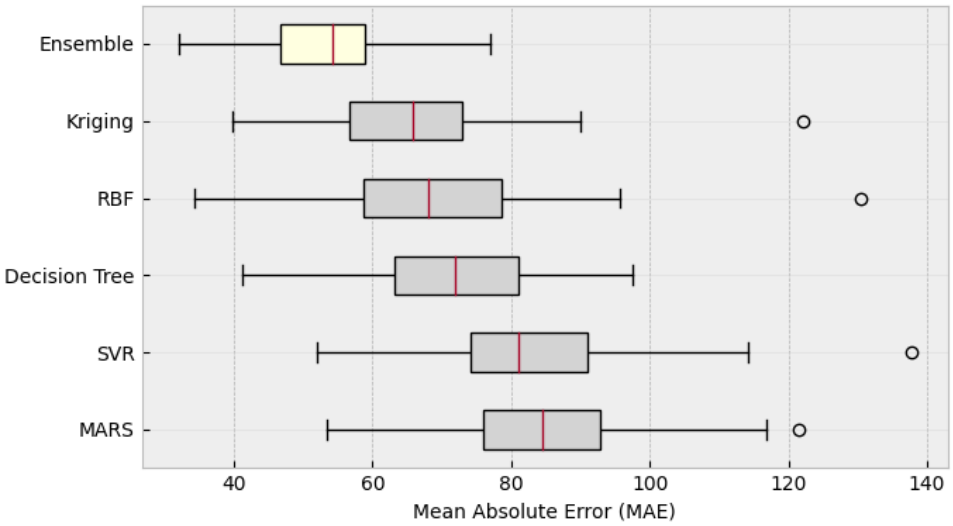


Figure 6.2.1: Boxplots of the MAE values per model on a 2D Himmelblau function with $x_1 \in [-5, 5], x_2 \in [-5, 5]$, sorted based on the mean MAE value.

As can be seen in Table 6.2.1, the performance of the base models largely depends on the objective function that has to be estimated. However, the found ensemble steadily performs best independently of the objective func-

tion. Therefore, this method is likely to perform well on any objective function without the need for any prior knowledge about the landscape.

Even though it would, in theory, be possible for base models to be chosen as the best 'ensemble', the results show that in the vast majority of cases, the ensembles are made up out of diverse combinations of base models. To illustrate this, Figure 6.2.2 shows an example of the distribution of weights in the best performing ensembles after repeating the ensemble generation process fifty times on all of the four test functions. It can be seen that all of the weight is seldomly assigned to a single base model, even further proving the efficacy of the linearly weighted ensembles.

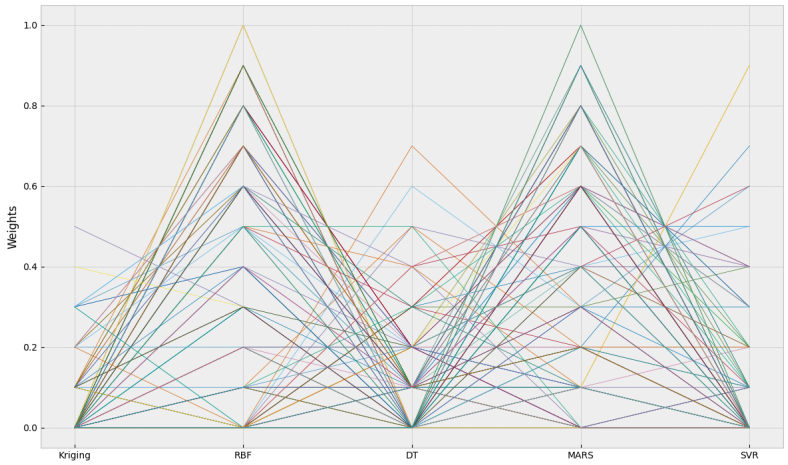


Figure 6.2.2: Distribution of the weights of the best performing ensembles on the four test problems, according to the MAE measure.

Table 6.2.1: Comparison of the performances of each surrogate model expressed in Mean Absolute Error (MAE), Maximum Error (ME) and Median Absolute Error (Med). The best performances per column are displayed in bold.

Model	Ackley			Branin			Himmelblau			Sphere		
	MAE	ME	Med	MAE	ME	Med	MAE	ME	Med	MAE	ME	Med
Kriging	3.47	6.11	2.79	30.80	67.24	19.85	66.57	130.15	44.44	6.38	14.06	4.68
RBF	2.28	4.16	1.75	16.93	36.94	9.74	69.82	134.23	47.67	4.20	9.36	2.78
DT	1.56	2.47	1.33	31.43	59.54	23.25	71.65	123.32	51.35	5.59	10.05	4.92
MARS	0.91	1.48	0.77	25.70	45.62	23.59	85.29	134.55	67.39	2.36	4.07	2.21
SVR	1.07	1.79	0.91	65.01	120.13	51.77	83.47	141.80	67.38	7.36	12.68	6.93
Ensemble	**0.79**	**1.29**	**0.61**	**14.24**	**28.79**	**7.56**	**52.52**	**97.19**	**33.32**	**2.23**	**3.91**	**1.63**

Finally, to illustrate how the linearly weighted ensemble method can be succesfully used to approximate functions, it can be seen in Figure 6.2.3 that the general trends of the Ackley function, despite its capricious nature, can be approximated pretty well using merely fifteen sample configurations. Moreover, the variable values at the found minimum are identical to those at the minimum of the real objective function.

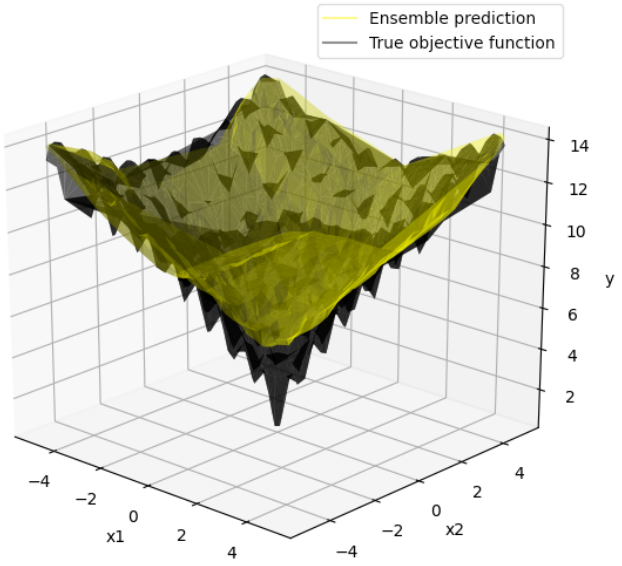


Figure 6.2.3: An example of an ensemble model best predicting the Ackley function based on 15 configurations. Found weights: (Kriging:0.0 , RBF:0.1 , DT:0.2 , MARS:0.2 , SVR:0.5).

6.3 Uncertainty Quantification

Subsequently, some experiments were conducted to investigate the impact of different infill criteria on the performance of the optimization process, when using the ensemble method as described in Section 5.1.2 with the MAE error metric.

6.3.1 Experimental Setup

Several infill criteria were used in combination with the ensemble method to find which methods works best. As prior works show very contrasting results with regards to which infill criteria perform better (recall Section 3.2), experiments were performed for multiple functions with varying dimensionality. As the goal is to approach the global minimum with as little function evaluations as possible, the optimization process is repeated for all infill criteria

with a budget of fifty function evaluations in every experimental setup. To account for randomness, results were averaged over five independent runs per experimental setup.

Sampling

In order to start the optimization process with a fair sample, Latin Hypercube Sampling (LHS) as described in section 5.1.1 was used to obtain a sample that is equally distributed across the search space. In these experiments, the initial sample size is limited to five times the number of input dimensions, taking into consideration that objective function evaluations are extremely expensive.

Infill Criteria

In the EGO setting, the infill criterion is used to integrate the models response surface with the uncertainty of the prediction, which has to be minimized in order to find the predicted minimal value (recall Section 2.1.3). Since the application of the Expected Improvement infill criterion as implemented in the original EGO algorithm is limited to Kriging-based regression models, ensemble-based EGO calls for different infill criteria. This experimental section covers the predicted value without any uncertainty measure, the Knn-variance empirical uncertainty measure [73], the variance between non-zero weighted base models and the variance over cross-validated predictions by the best performing ensemble.

Infill criterion 1: Ensemble Predicted Value (as described in Section 5.1.4.)

$$EPV(\vec{x}) = \sum_{i=0}^{N} \vec{w}_i \cdot \hat{f}_i(\vec{x})^T, \tag{6.3.1}$$

where:
$\hat{f}_i$: an individual base model.
$\vec{w}_i$: the vector of best weights of the corresponding objective function.

Infill criterion 2: Knn-variance

$$KPV = EPV(\vec{x}) - \hat{U}_{k-NN}, \tag{6.3.2}$$

with $\hat{U}_{k-NN}$ as formulated in Equation 5.1.2

Infill criterion 3: Variance between non-zero weighted base models

$$VPV = EPV(\vec{x}) - \text{Var}\left[\{\hat{f}_i(\vec{x}) \cdot \vec{w}_i | \vec{w}_i \neq 0, i = 1, \ldots, k\}\right]. \tag{6.3.3}$$

where

k: amount of base models.

Infill criterion 4: Cross validated standard deviation between ensembles

$$CVPV = EPV(\vec{x}) - \sqrt{\frac{1}{M-1}\sum_{j=0}^{M}(EPV_j(\vec{x}) - \overline{EPV}(\vec{x}))^2}. \tag{6.3.4}$$

where
M: Number of cross-validation folds

By using the cross-validation method, ten ensembles are created, making use of the weights that had the best overall performance. This results in ten predictions per new assigned points, of which the variance can be used as an uncertainty measure.

Test Problems

To gain insight into how the infill criteria perform, they were tested on the Rosenbrock function, a normalized Lp space function, and the Forrester function, all for which the number of input dimensions can vary.

Test function 1: Rosenbrock

$$f(\vec{x}) = \sum_{i=1}^{N-1}[100(x_{i+1} - x_i)^2 + (1 - x_i)^2], \tag{6.3.5}$$

where $x_i \in [-5, 5]\ \forall i \in [1, \ldots, n]$.

Test function 2: Lp Norm

$$f(\vec{x}) = \|\vec{x}\|_p = \sqrt[p]{\sum_{i}^{nx}|x_i|^p} \tag{6.3.6}$$

where $x_i \in [-3, 3]\ \forall i \in [1, \ldots, n]$

Test function 3: Forrester

The third function is a multi-dimensional adaptation of the Forrester function as implemented in [72] with $x_i \in [-2, 2]\ \forall i \in [1, \ldots, n]$.

6.3.2 Results

The infill criteria were tested and compared to each other in terms of evaluated function values and convergence rates.

Minimum Objective Values

Table 6.3.1 shows the minimal objective values obtained after fifty function evaluations, averaged over ten independent runs. Here, it becomes clear the cross-validated variance infill criterion performs worst in almost all cases, even though it was massively more costly to compute both in terms of time and computational expenses. This poor performance is probably due to the detection of different minima over the folds, which, by averaging the results, leads to the proposal of points somewhere in between promising locations.

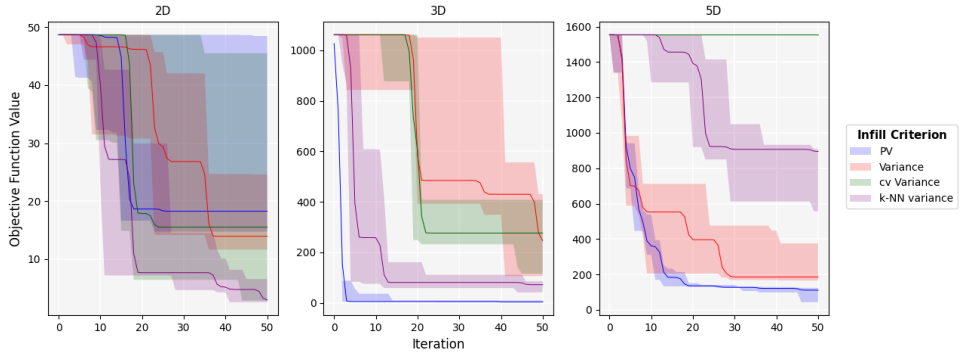
Furthermore, it becomes clear that the $k-$NN infill criterion significantly outperforms the other infill criteria in most cases, except for the Rosenbrock function in higher dimensions. As the dimensionality of the Rosenbrock function increases, the predicted value without any measure for uncertainty easily outperforms all the other infill criteria, corresponding the findings in [60]. This result can most likely be dedicated to the wide search domain and range of the Rosenbrock function, in which case exploitation would be preferred over exploration, especially with a small number of function evaluations. It could also be argued that the $k-$NN infill criterion performs suboptimal in a large search domain of higher dimensionality as data points might be too far away from each other to obtain any beneficial information from their interactions.

Convergence

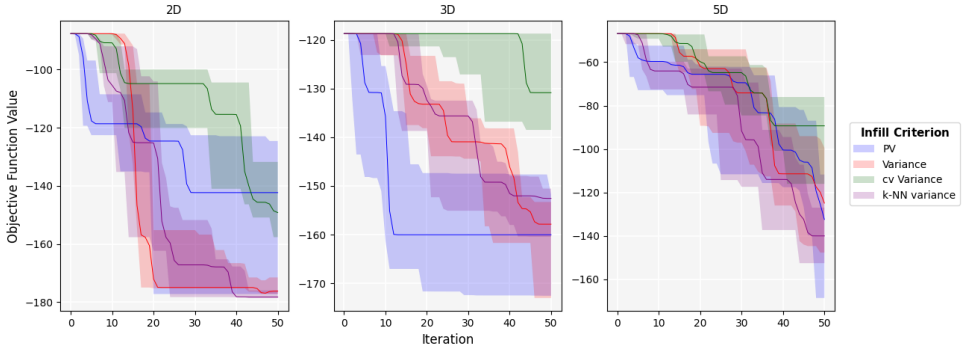
In addition, Figure 6.3.1 shows the convergence plots of Ensemble-based EGO (E-EGO) per infill criterion on the three single objective test functions. In terms of convergence, the infill criterion perform quite differently, depending on the functions they minimize. On the Lp Norm function, the $k-$NN infill criterion shows remarkable and unmatched results, actually reaching the global minimum within 50 evaluations in some case and doing so much faster than the other infill criteria. Also, the $k-$NN infil criterion converges faster than the other algorithms on the Rosenbrock and Forrester functions in lower dimensionality, but seems to perform similarly or worse in higher dimensionality, as becomes clear in table 6.3.1.

Table 6.3.1: Minimal obtained objective function values per infill criterion averaged over 10 independent runs of the algorithm. ($k = 9$ in k-NN, 10 folds in cv-Var) The best achieved values per function are shown in bold.

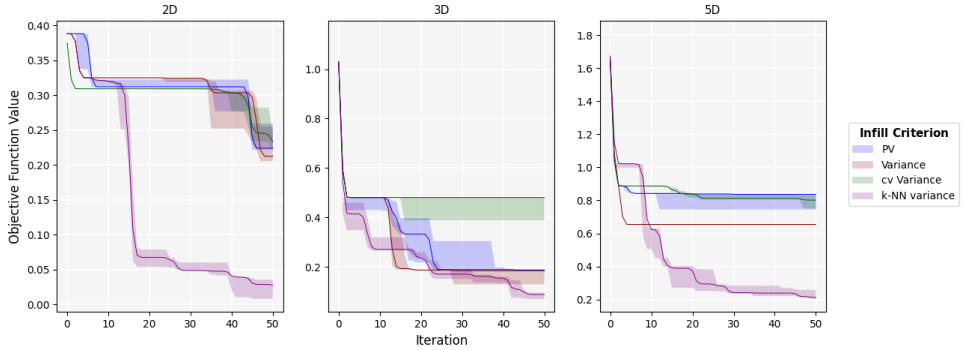
IC	Forrester			Lp Norm			Rosenbrock		
	2D	3D	5D	2D	3D	5D	2D	3D	5D
PV	-149.31	-161.20	-132.91	0.2427	0.1987	0.7782	28.50	**7.99**	**88.02**
Var	-172.59	-155.74	-125.51	0.2365	0.1510	0.6546	19.16	251.22	251.76
k-NN	**−176.14**	**−161.78**	**−139.14**	**0.0224**	**0.0835**	**0.2173**	**5.03**	62.47	730.38
cv-Var	-141.29	-132.42	-94.83	0.2381	0.4191	0.7435	22.89	288.64	1554.15



(a) Rosenbrock function



(b) Forrester function



(b) Lp Norm function

Figure 6.3.1: Convergence plots of single objective E-EGO using the Predicted value, Variance, k-NN variance and Cross-Validated Variance as infill criteria. Experiments involve 2,3 and 5 input dimensions (from left to right). The y-axis shows the best achieved value so far with respect to the global minimum. The center line denotes the median across five repetitions, surrounded by a plane representing all values between the lower and upper quartiles.

All in all, the $k-$NN showed favorable results in most of the cases, except for being outmatched by the Predicted Value in two cases. Even though exploitation might be more important than exploration in case of more dimensions, it is, however, still desired to incorporate some level of uncertainty

in case of multi-objective optimization, as the goal is to find points that perform well on all objectives rather than finding the global minimum of one of the objective functions. Hence, on basis of these experiments, it can be concluded that the $k-$NN infill criterion is capable of exploiting promising locations very well, while still providing a globally accurate model.

6.4 Algorithm Comparison

In this final section on experimental evaluation, the proposed E-SMS-EGO algorithm is compared to similar, widely used multi-objective optimization algorithms in terms of hypervolume and spread. The algorithm is compared to the famous NSGA-II, MOEA/D and C-TAEA algorithms. The realization of this chapter largely depends on the excellent *Pymoo* multi-objective optimization benchmarking package provided by Blank and Deb [10].

6.4.1 Experimental Setup

To compare SMS-EGO to the competing algorithms, all of the algorithms were run ten times with different initial samples and random seeds.

Sampling and Evaluation Budget

Again, LHS was used to obtain an initial sample to start the optimization process. The initial sample size for E-SMS-EGO was set to $5 \times N$, with N : number of input variables. As the goal is to acquire a well-spread Pareto-front in as little function evaluations as possible, the amount of function evaluations was set to 25.

Functions

Inspired by earlier work on multi-objective optimization and the extensive benchmark suite provided by [10], a diverse collection of multi-objective optimization problems was composed to compare the algorithm performances. To ensure a thorough comparison, this collection includes some artificially designed two-objective problems and real-world like problems. Table 6.4.1 provides an overview of the optimization problems in terms of input dimension (n), Lower Bounds (LB) and Upper Bounds (UB) of the input variables, as well as the number of objectives (k) and the hypervolume reference point (ref). In addition, it shows if the problem is artificially designed (AD) or real-world like (RWL). This collection consists of the following functions: BNH [17], TNK[23], CTP1 [22], ZDT4 [23], Kursawe (KSW) [45], Welded Beam (WB) [34], Car Side Impact (CSI) [20].

Table 6.4.1: Artificially designed and Real World Like multi-objective optimization problems as implemented in the *pymoo* package.

Problem	type	k	n	LB	UB	ref
BNH	AD	2	2	[0, 0]	[5, 3]	[140, 50]
TNK	AD	2	2	[0, 0]	[π, π]	[2, 2]
CTP1	AD	2	2	[0, 0]	[1, 1]	[1, 2]
ZDT4	AD	2	10	[0,-5,. . . ,-5]^n	[1,5,. . . ,5]^n	[1,260]
KSW	AD	2	3	[-5, -5, -5]	[5, 5, 5]	[-10, 2]
WB	RWL	2	4	[0.125, 0.1, 0.1, 0.125]	[5, 10, 10, 5]	[350,1]
CSI	RWL	3	7	[0.5, 0.45, 0.5, 0.5, 0.875, 0.4, 0.4]	[1.5, 1.35, 1.5, 1.5, 2.625, 1.2, 1.2]	[42, 4.5, 13]

Algorithms

The competing algorithms are *NSGA-II*, *MOEA/D* and *C-TAEA* as implemented in the *Pymoo* package [10]. As these algorithms make use of populations instead of a single point per iteration, the population sizes and number of generations were both set to 5 to equal the total amount of function evaluations, on top of the initial sample.

6.4.2 Results

Hypervolume

As becomes clear in Table 6.4.2, E-SMS-EGO significantly outperforms NSGA-II, MOEA/D and C-TAEA in terms of the Hypervolume scores of the obtained Pareto-fronts. Also, in most cases, the standard deviation is lower for E-SMS-EGO, suggesting that the proposed method is more robust and stable than the competing algorithms. On some functions, e.g. TNK, WB, some of the competing algorithms show very poor results in terms of Hypervolume, with abnormally high standard deviations. In these cases, the algorithms did not succeed to find enough feasible, Pareto-optimal solutions below the reference point, therefore receiving a Hypervolume score of 0 in some of the runs. On some problems, especially MOEA/D seemed to perform poorly when allowed only a small amount of function evaluations. However, E-SMS-EGO did not seem to suffer from this issue and was able to find a Pareto-front in all of the runs.

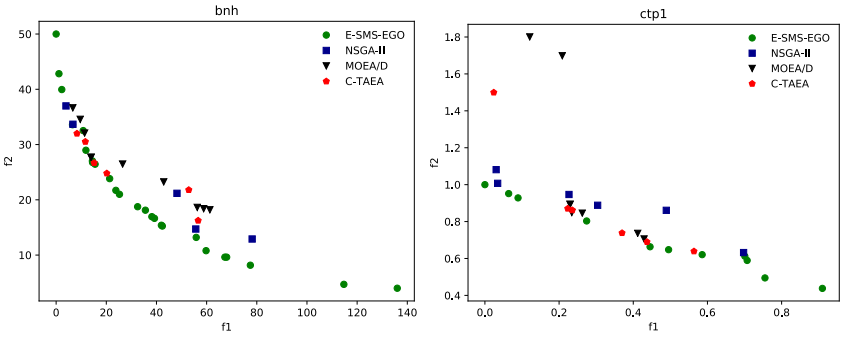
Table 6.4.2: Mean Hypervolume score with respect to the reference point for each test function. The best results per test function are shown in boldface if they were significantly higher according to Welch's t-test with $\alpha : 0.05$.

Problem	Measure	NSGA-II	MOAE/D	C-TAEA	E-SMS-EGO
BNH	HV	4760	4617	4723	**5035**
	Std.	134.4	209.8	157.7	35.85
TNK	HV	1.849	0.000	1.015	**3.926**
	Std.	0.597	0.000	1.134	0.116
CTP1	HV	1.115	1.136	1.103	**1.261**
	Std.	0.088	0.061	0.085	0.016
ZDT4	HV	162.1	116.5	161.0	**176.0**
	Std.	18.90	43.16	17.71	14.62
KSW	HV	41.47	43.04	42.01	**56.09**
	Std.	15.05	19.35	12.97	9.923
WB	HV	32.90	1.217	32.49	**33.63**
	Std.	1.298	5.969	4.690	1.797
CSI	HV	11.14	10.56	16.67	**19.41**
	Std.	3.205	2.092	1.101	0.260

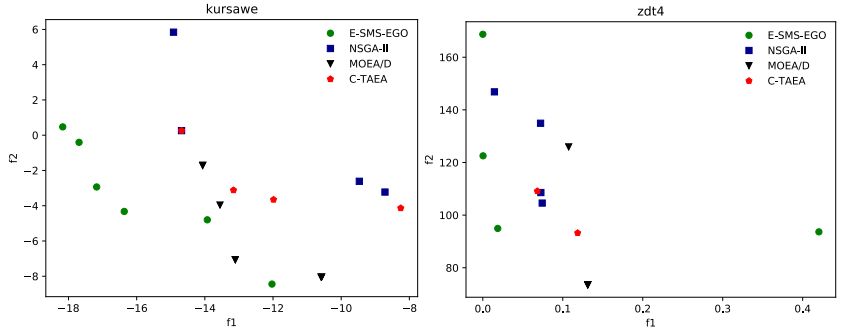
Spread

In addition, Figures 6.4.1 and 6.4.2 show the Pareto-frontiers obtained by the four algorithms on five of the test functions. Here, it can be seen that E-SMS-EGO in general succeeds to find the best Pareto-fronts compared to the competing algorithms, as the solutions are located more towards the minimal values on all objectives. In addition to finding more Pareto-optimal solutions, the solutions found by E-SMS-EGO are well-spread across the objective space, which is demonstrated nicely, especially for the BNH, CTP1, Kursawe and Car Side Impact problems.

Furthermore, it is shown that, for some problems, only a small amount of Pareto-optimal solutions could be found, which is most likely explained by the limited amount of allowed function evaluations. Especially for NSGA-II and MOEA/D, which were only allowed small population sizes, this explains why so little Pareto-optimal solutions were found. However, the vast majority of solutions that were found by the competing algorithms were still inferior to the solutions found by E-SMS-EGO, verifying that the proposed method beats the competing algorithms altogether in terms of efficacy in multi-objective optimization.



(a) Pareto-fronts BNH **(b)** Pareto-fronts CTP1



(c) Pareto-fronts Kursawe **(d)** Pareto-fronts ZDT4

Figure 6.4.1: Pareto frontiers obtained by the four algorithms on four of the test functions.

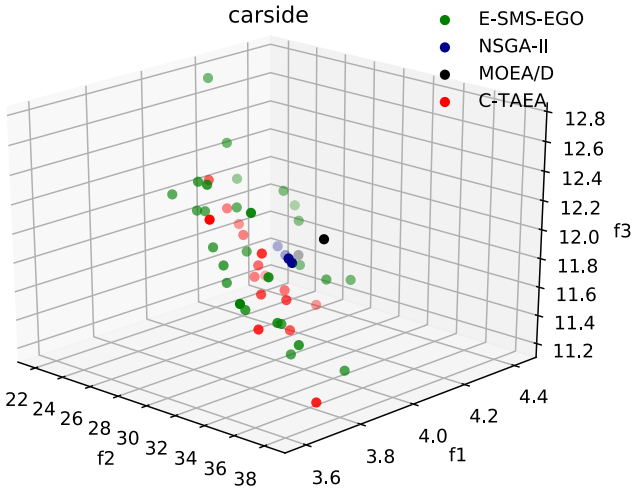


Figure 6.4.2: Pareto frontiers obtained by the four algorithms on the three-objective Car Side Impact problem.

–It's what you learn after you know it all that counts.
John Wooden

7

Conclusion and Future Work

In this thesis, the novel Ensemble-based Efficient Global Optimization $S-$Metric Selection (E-SMS-EGO) algorithm is proposed, and has been shown to be very successful in finding well-performing, Pareto-optimal solutions to multi-objective optimization problems with a very limited evaluation budget. By heavily exploiting already known data points, E-SMS-EGO has been shown to outperform comparable state-of-the art multi-objective optimization algorithms, i.e., NSGA-II, MOEA/D and C-TAEA on a diverse collection of artificially designed and real world like test problems.

7.1 Contributions

Multiple experiments were performed with different techniques to compose the proposed algorithm in an optimal way. This resulted in an algorithm that improves upon the SMS-EGO algorithm by using optimally weighted ensembles of regression models as surrogates. By further extending the algorithm with the k-NN variance measure as a method of uncertainty quantification, E-SMS-EGO was able to find minimal solutions that were nicely spread across the objective space, with just a small number of function evaluations.

7.2 Limitations

A minor limitation of the proposed method is that the creation of ensembles might be biased. The hyperparameters of some models, e.g. Kriging, MARS are already optimized in some way by nature, whereas the hyperparameters of other regression models are not tuned at all. In some cases, this might result in favoring some models over others in terms of weight distribution, thus missing out on beneficial information from the other surrogate models. In order to overcome this problem, an interesting addition to the proposed algorithm would be to tune the hyperparameters of the individual surrogate models before combining them into ensembles. Apart from making the ensemble creation unbiased, adding a hyperparameter tuning step to the algorithm is expected to be beneficial to its performance in any case, as the predictions of the individual base models would be more accurate.

7.3 Future Work

In addition, there are many interesting directions for further research on the proposed algorithm, of which some are mentioned in the following.

In the first place, the way in which ensembles are composed in the proposed algorithm is relatively simple. There is probably some room for improvement in the way the base models are combined into ensembles. Therefore, it could be interesting to experiment with more sophisticated weighting methods, rather than global linear weighted averaging. For instance, weights could be combined locally in a point-wise manner to detect local trends in objective functions.

In the experimental results, the number of base models was limited to five. It is expected to increase performance of the proposed method even further if more surrogate models would be incorporated into the ensembles. The proposed method is, in theory, scalable to a large extent, so adding more surrogate models should not raise any problems, except for an increase in computation time. In case of a large number of surrogate models, a way to decrease computation time could be to find the optimal weights by means of an evolutionary search, as proposed by Friese et al. [32].

Even though the computational requirements for point prediction are generally negligible in comparison with the real function evaluations, some steps could be taken to limit the required computational resources for proposing potential solutions. Now, a new ensemble is created every time a new data point is evaluated. In theory, it would be possible to limit the creation of new ensembles to occur after a fixed amount of newly added data points. This could, however, possibly deteriorate the performance of the algorithm, so such changes would have to be administered with caution.

Also, even though E-SMS-EGO has been shown to perform well on some

constrained multi-objective optimization problems, it does currently not consist of a mechanism for active constraint handling. Potentially, including such a mechanism could further improve the performance of the algorithm.

Finally, a logical next step would be to apply the E-SMS-EGO algorithm on a real world multi-objective optimization problem to test its practical relevance and efficacy in addition to the displayed theoretical results.

Bibliography

[1] Erdem Acar. Various approaches for constructing an ensemble of metamodels using local measures. *Structural and Multidisciplinary Optimization*, 42:879–896, 12 2010. doi: 10.1007/s00158-010-0520-z.

[2] Erdem Acar and Masoud Rais-Rohani. Ensemble of metamodels with optimized weight factors. *Structural and Multidisciplinary Optimization*, 37:279–294, 04 2008. doi: 10.1007/s00158-008-0230-y.

[3] David H Ackley. The model. In *A Connectionist Machine for Genetic Hillclimbing*, pages 29–70. Springer, 1987.

[4] Richard Allmendinger, Michael Emmerich, Jussi Hakanen, Yaochu Jin, and Enrico Rigoni. Surrogate-assisted multicriteria optimization: Complexities, prospective solutions, and business case. *Journal of Multi-Criteria Decision Analysis*, 24, 12 2016. doi: 10.1002/mcda.1605.

[5] Viktor Astakhov. *Design of Experiment Methods in Manufacturing: Basics and Practical Applications*, pages 1–54. 01 2012. ISBN 978-3-642-25858-9. doi: 10.1007/978-3-642-25859-6_1.

[6] Samineh Bagheri, Wolfgang Konen, and Thomas Bäck. Comparing kriging and radial basis function surrogates. pages 243–259, 10 2017.

[7] Sunith Bandaru, Amos Ng, and Kalyan Deb. On the performance of classification algorithms for learning pareto-dominance relations. 07 2014.

[8] B. Bischl, O. Mersmann, H. Trautmann, and C. Weihs. Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary Computation*, 20(2):249–275, 2012.

[9] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., USA, 1995. ISBN 0198538642.

[10] Julian Blank and Kalyanmoy Deb. pymoo: Multi-objective optimization in python. *ArXiv*, abs/2002.04504, 2020.

[11] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 144–152, New York, NY, USA, 1992. Association for Computing Machinery. ISBN 089791497X. doi: 10.1145/130385.130401. URL `https://doi.org/10.1145/130385.130401`.

[12] Mohamed Amine Bouhlel, John T. Hwang, Nathalie Bartoli, Rémi Lafage, Joseph Morlier, and Joaquim R. R. A. Martins. A python surrogate modeling framework with derivatives. *Advances in Engineering Software*, page 102662, 2019. ISSN 0965-9978. doi: https://doi.org/10.1016/j.advengsoft.2019.03.005.

[13] F. H. Branin. Widely convergent method for finding multiple solutions of simultaneous nonlinear equations. *IBM Journal of Research and Development*, 16(5):504–522, 1972.

[14] Juergen Branke, Kalyan Deb, Kaisa Miettinen, and Slowinski Roman. Multiobjective optimization, interactive and evolutionary approaches. 01 2008.

[15] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[16] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. ISSN 0885-6125.

[17] Carlos Coello, David Veldhuizen, and Gary Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems Second Edition*. 01 2007. doi: 10.1007/978-0-387-36797-2.

[18] Dennis Cox and Susan John. Sdo: A statistical method for global optimization. 03 1997.

[19] Roy de Winter, Bas van Stein, Matthys Dijkman, and Thomas Bäck. Designing ships using constrained multi-objective efficient global optimization. In *Machine Learning, Optimization, and Data Science*, pages 191–203, Cham, 2019. Springer International Publishing.

[20] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[22] Kalyan Deb. *Multiobjective Optimization Using Evolutionary Algorithms. Wiley, New York*. 01 2001.

[23] Kalyanmoy Deb, Amrit Pratap, and T. Meyarivan. Constrained test problems for multi-objective evolutionary optimization. volume 1993, pages 284–298, 03 2001. doi: 10.1007/3-540-44719-9_20.

[24] Alan Diaz-Manriquez, Gregorio Toscano Pulido, Jose Barron-Zambrano, and Edgar Tello-Leal. A review of surrogate assisted multiobjective evolutionary algorithms. *Computational Intelligence and Neuroscience*, 2016:1–14, 06 2016. doi: 10.1155/2016/9420460.

[25] Harris Drucker, Christopher J. C. Burges, Linda Kaufman, Alex J. Smola, and Vladimir Vapnik. Support vector regression machines. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 155–161. MIT Press, 1997. URL `http://papers.nips.cc/paper/1238-support-vector-regression-machines.pdf`.

[26] Matthias Ehrgott. Vilfredo pareto and multi-objective optimization. *Doc. math*, pages 447–453, 2012.

[27] Michael T M Emmerich and André H Deutz. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural computing : an international journal.*, 17(3):585–609, 2018. ISSN 1567-7818.

[28] Clive A. J. Fletcher. *Computational Fluid Dynamics: An Introduction*, pages 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 1988. ISBN 978-3-642-97035-1. doi: 10.1007/978-3-642-97035-1_1. URL `https://doi.org/10.1007/978-3-642-97035-1_1`.

[29] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *ICML*, 1996.

[30] Jerome H. Friedman. Multivariate adaptive regression splines. *Ann. Statist.*, 19(1):123–141, 03 1991. doi: 10.1214/aos/1176347973. URL `https://doi.org/10.1214/aos/1176347973`.

[31] Martina Friese, Martin Zaefferer, Thomas Bartz-Beielstein, Oliver Flasch, Patrick Koch, Wolfgang Konen, and Boris Naujoks. Ensemble based optimization and tuning algorithms. 01 2011.

[32] Martina Friese, Thomas Bartz-Beielstein, and Michael Emmerich. Building ensembles of surrogates by optimal convex combination. 05 2016.

[33] Tushar Goel, Raphael Haftka, Wei Shyy, and Nestor Queipo. Ensemble of surrogates. *Structural and Multidisciplinary Optimization*, 33:199–216, 03 2007. doi: 10.1007/s00158-006-0051-9.

[34] Wenyin Gong, Zhihua Cai, and Li Zhu. An efficient multiobjective differential evolution algorithm for engineering design. *Structural and Multidisciplinary Optimization*, 38:137–157, 04 2009. doi: 10.1007/s00158-008-0269-9.

[35] Rolland L. Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research (1896-1977)*, 76(8):1905–1915, 1971. doi: 10.1029/JB076i008p01905. URL `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/JB076i008p01905`.

[36] David Mautner Himmelblau. *Applied Nonlinear Programming*. McGraw-Hill, 1972. ISBN 0070289212.

[37] Daniel Horn, Tobias Wagner, Dirk Biermann, Claus Weihs, and Bernd Bischl. Model-based multi-objective optimization: Taxonomy, multi-point proposal, toolbox and benchmark. In António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos Coello Coello, editors, *Evolutionary Multi-Criterion Optimization*, pages 64–78, Cham, 2015. Springer International Publishing. ISBN 978-3-319-15934-8.

[38] In-Hee Lee, Soo-Yong Shin, and Byoung-Tak Zhang. Dna sequence optimization using constrained multi-objective evolutionary algorithm. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 4, pages 2270–2276 Vol.4, 2003.

[39] Ping Jiang, Qi Zhou, and Xinyu Shao. *Surrogate-Model-Based Design and Optimization*, pages 135–236. 01 2020. ISBN 978-981-15-0730-4. doi: 10.1007/978-981-15-0731-1_7.

[40] R. Jin, Wei Chen, and Timothy Simpson. Comparative studies of metamodeling techniques under multiple modeling criteria. *Structural and Multidisciplinary Optimization*, 23:1–13, 01 2001. doi: 10.1007/s00158-001-0160-4.

[41] Yaochu Jin and Bernhard Sendhoff. Pareto-based multiobjective machine learning: An overview and case studies. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38: 397 – 415, 06 2008. doi: 10.1109/TSMCC.2008.919172.

[42] Donald Jones, Matthias Schonlau, and William Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 12 1998. doi: 10.1023/A:1008306431147.

[43] Joshua Knowles. Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

[44] D.G. Krige. *A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand*. Chemical, Metallurgical and Mining Society of South Africa, 1951. URL `https://books.google.nl/books?id=Mu6SnQAACAAJ`.

[45] Frank Kursawe. A variant of evolution strategies for vector optimization. 02 1999. doi: 10.1007/BFb0029752.

[46] Yongbin Lee and Dong-Hoon Choi. Pointwise ensemble of meta-models using v nearest points cross-validation. *Structural and Multidisciplinary Optimization*, 50:383–394, 09 2014. doi: 10.1007/s00158-014-1067-1.

[47] Deming Lei. Multi-objective production scheduling: A survey. *International Journal of Advanced Manufacturing Technology*, 43:926–938, 08 2009. doi: 10.1007/s00170-008-1770-4.

[48] K. Li, R. Chen, G. Fu, and X. Yao. Two-archive evolutionary algorithm for constrained multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 23(2):303–315, 2019.

[49] D. Lim, Y. Jin, Y. Ong, and B. Sendhoff. Generalizing surrogate-assisted evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 14(3):329–355, 2010.

[50] Haitao Liu, Shengli Xu, Xiaofang Wang, Jigang Meng, and Shuhua Yang. Optimal weighted pointwise ensemble of radial basis functions with different basis functions. *AIAA Journal*, 54, 06 2016. doi: 10.2514/1.J054664.

[51] Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag. Comparison-based optimizers need comparison-based surrogates. 09 2010. doi: 10.1007/978-3-642-15844-5_37.

[52] M. Mckay, Richard Beckman, and William Conover. A comparison of three methods for selecting vales of input variables in the analysis of output from a computer code. *Technometrics*, 21:239–245, 05 1979. doi: 10.1080/00401706.1979.10489755.

[53] Kaisa Miettinen and Marko Mäkelä. On scalarizing functions in multiobjective optimization. *OR Spectrum*, 24:193–213, 01 2002. doi: 10.1007/s00291-001-0092-9.

[54] Juliane Mueller and Christine Shoemaker. Influence of ensemble surrogate models and sampling strategy on the solution quality of algorithms for computationally expensive black-box global optimization problems. *Journal of Global Optimization*, 60:123–144, 10 2014. doi: 10.1007/s10898-014-0184-0.

[55] Azouz Nesrine, Slim Bechikh, and Lamjed Ben Said. Steady state ibea assisted by mlp neural networks for expensive multi-objective optimization problems. 07 2014. doi: 10.1145/2576768.2598271.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[57] Wolfgang Ponweiser, Tobias Wagner, Dirk Biermann, and Markus Vincze. Multiobjective optimization on a limited budget of evaluations using model-assisted S-metric selection. In Günter Rudolph, Thomas Jansen, Nicola Beume, Simon Lucas, and Carlo Poloni, editors, *Parallel Problem Solving from Nature – PPSN X*, pages 784–794, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-87700-4.

[58] Nestor Queipo, Raphael Haftka, Wei Shyy, Tushar Goel, Rajkumar Vaidyanathan, and P. Tucker. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41:1–28, 01 2005. doi: 10.1016/j.paerosci.2005.02.001.

[59] S. Rao. *Classical Optimization Techniques*, chapter 2, pages 57–108. John Wiley Sons, Ltd, 2019. ISBN 9781119454816. doi: 10.1002/9781119454816.ch2. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119454816.ch2`.

[60] Frederik Rehbach, Martin Zaefferer, Boris Naujoks, and Thomas Bartz-Beielstein. Expected improvement versus predicted value in surrogate-based optimization. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 868–876, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371285. doi: 10.1145/3377930.3389816. URL `https://doi.org/10.1145/3377930.3389816`.

[61] Alejandro Rosales-Pérez, Carlos Coello, Jesus Gonzalez, Carlos Alberto Reyes-Garcia, and Hugo Jair Escalante. A hybrid surrogate-based approach for evolutionary multi-objective optimization. pages 2548–2555, 06 2013. ISBN 978-1-4799-0453-2. doi: 10.1109/CEC.2013.6557876.

[62] Dilip Roy and Bithin Datta. A review of surrogate models and their ensembles to develop saltwater intrusion management strategies in coastal aquifers. *Earth Systems and Environment*, 08 2018. doi: 10.1007/s41748-018-0069-3.

[63] Jason Rudy. A python implementation of jerome friedman's multivariate adaptive regression splines. *Github repository*, 2014.

[64] Renhe Shi, Li Liu, Teng Long, and Jian Liu. An efficient ensemble of radial basis functions method based on quadratic programming. *Engineering Optimization*, 48(7):1202–1225, 2016. doi: 10.1080/0305215X.2015.1100470. URL https://doi.org/10.1080/0305215X.2015.1100470.

[65] Alexander J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

[66] Jialin Song, Yuxin Chen, and Yisong Yue. A general framework for multi-fidelity bayesian optimization with gaussian processes. In *AISTATS*, 2019.

[67] Xueguan Song, Liye Lv, Jieling Li, Wei Sun, and Jie Zhang. An Advanced and Robust Ensemble Surrogate Model: Extended Adaptive Hybrid Functions. *Journal of Mechanical Design*, 140(4), 02 2018. ISSN 1050-0472. doi: 10.1115/1.4039128. URL https://doi.org/10.1115/1.4039128. 041402.

[68] Robert A. Stine. Bootstrap prediction intervals for regression. *Journal of the American Statistical Association*, 80(392):1026–1031, 1985. doi: 10.1080/01621459.1985.10478220. URL https://amstat.tandfonline.com/doi/abs/10.1080/01621459.1985.10478220.

[69] B. Szabó and I. Babuška. *Introduction to Finite Element Analysis: Formulation, Verification and Validation*. Wiley Series in Computational Mechanics. Wiley, 2011. ISBN 9780470977286. URL https://books.google.nl/books?id=bbi7cQAACAAJ.

[70] Mohammad Tabatabaei, Jussi Hakanen, Markus Hartikainen, Kaisa Miettinen, and Karthik Sindhya. A survey on handling computationally expensive multiobjective optimization problems using surrogates: non-nature inspired methods. *Structural and Multidisciplinary Optimization*, 52, 07 2015. doi: 10.1007/s00158-015-1226-z.

[71] Holger Ulmer, Felix Streichert, and Andreas Zell. Evolution strategies assisted by gaussian processes with improved pre-selection criterion. *Proceedings of The Ieee Congress On Evolutionary Computation, 2003. Cec '03*, 1, 04 2004. doi: 10.1109/CEC.2003.1299643.

[72] Sander van Rijn and Sebastian Schmitt. Mf2: A collection of multi-fidelity benchmark functions in python. *Journal of Open Source Software*, 5(52):2049, 2020. doi: 10.21105/joss.02049. URL https://doi.org/10.21105/joss.02049.

[73] Bas van Stein, Hao Wang, Wojtek Kowalczyk, and Thomas Bäck. *A Novel Uncertainty Quantification Method for Efficient Global Optimization*, pages 480–491. 01 2018. ISBN 978-3-319-91478-7. doi: 10.1007/978-3-319-91479-4_40.

[74] Felipe Viana, Raphael Haftka, and Valder Steffen, Jr. Multiple surrogates: How cross-validation errors can help us to obtain the best predictor. *Structural and Multidisciplinary Optimization*, 39:439–457, 10 2009. doi: 10.1007/s00158-008-0338-0.

[75] Tobias Wagner. *Planning and Multi-Objective Optimization of Manufacturing Pro- cesses by Means of Empirical Surrogate Models*. Vulkan Verlag, Essen, 2013.

[76] Tobias Wagner, Nicola Hochstrate, and Boris Naujoks. Pareto-, aggregation-, and indicator-based methods in many-objective optimization. *EMO 2007*, pages 742–756, 09 2006. doi: 10.17877/DE290R-9028.

[77] Yuan Wang, Zhong-Hua Han, Yu Zhang, and Wenping Song. Efficient global optimization using multiple infill sampling criteria and surrogate models. 01 2018. doi: 10.2514/6.2018-0555.

[78] Pengcheng Ye. A review on surrogate-based global optimization methods for computationally expensive functions. 2019.

[79] Pengcheng Ye and Guang Pan. Global optimization method using ensemble of metamodels based on fuzzy clustering for design space reduction. *Engineering with Computers*, 33, 10 2016. doi: 10.1007/s00366-016-0490-x.

[80] Yifan Ye, Zhanxue Wang, and Xiaobo Zhang. An optimal pointwise weighted ensemble of surrogates based on minimization of local mean square error. *Structural and Multidisciplinary Optimization*, 02 2020. doi: 10.1007/s00158-020-02508-4.

[81] Hanfeng Yin, Fang Hongbing, Guilin Wen, Matthew Gutowski, and Youye Xiao. On the ensemble of metamodels with multiple regional optimized weight factors. *Structural and Multidisciplinary Optimization*, 58, 01 2018. doi: 10.1007/s00158-017-1891-1.

[82] Ciprian Zavoianu, Edwin Lughofer, Gerd Bramerdorfer, Wolfgang Amrhein, and Erich Peter Klement. An effective ensemble-based method for creating on-the-fly surrogate fitness functions for multi-objective evolutionary algorithms. page In press., 09 2013. doi: 10.1109/SYNASC.2013.38.

[83] Jian Zhang, Xinxin Yue, Jiajia Qiu, Muyu Zhang, and Xiaomei Wang. A unified ensemble of surrogates with global and local measures for global metamodelling. *Engineering Optimization*, pages 1–22, 03 2020. doi: 10.1080/0305215X.2020.1739280.

[84] Jie Zhang, Souma Chowdhury, and Achille Messac. An adaptive hybrid surrogate model. *Struct. Multidiscip. Optim.*, 46(2):223–238, August 2012. ISSN 1615-147X. doi: 10.1007/s00158-012-0764-x. URL `https://doi.org/10.1007/s00158-012-0764-x`.

[85] Qingfu Zhang, Wudong Liu, Edward Tsang, and Botond Virginas. Expensive multiobjective optimization by moea/d with gaussian process model. *IEEE Transactions on Evolutionary Computation*, 14(3):456–474, 2009.

[86] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1:32–49, 03 2011. doi: 10.1016/j.swevo.2011.03.001.

[87] Xiao Jian Zhou, Yi Zhong Ma, and Xu Fang Li. Ensemble of surrogates with recursive arithmetic average. *Struct. Multidiscip. Optim.*, 44(5):651–671, November 2011. ISSN 1615-147X. doi: 10.1007/s00158-011-0655-6. URL `https://doi.org/10.1007/s00158-011-0655-6`.

[88] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

[89] Eckart Zitzler, Lothar Thiele, Marco Laumanns, C.M. Fonseca, and Viviane Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *Evolutionary Computation, IEEE Transactions on*, 7:117 – 132, 05 2003. doi: 10.1109/TEVC.2003.810758.