Compositional Multi-objective Parameter Tuning

Master Thesis

(permalink)

Authors

- **Oleksandr Husak**
  0000-0002-1676-0042 Valavanca
  Faculty of Computer Science, TU Dresden

- **M.Sc. Dmytro Pukhkaiev**
  website
  Faculty of Computer Science, TU Dresden

- **Dr.-Ing. Sebastian Götz**
  website
  Faculty of Computer Science, TU Dresden

- **Prof. Dr. rer. nat habil. Uwe Assmann**
  website
  Faculty of Computer Science, TU Dresden

Abstract

Multi-objective decision-making is critical for everyday tasks and engineering problems. Finding the perfect trade-off to maximize all the solution's criteria requires a considerable amount of experience or the availability of a significant number of resources. This makes these decisions difficult to achieve for expensive problems such as engineering. Most of the time, to solve such expensive problems, we are limited by time, resources, and available expertise. Therefore, it is desirable to simplify or approximate the problem when possible before solving it. The state-of-the-art approach for simplification is model-based or surrogate-based optimization. These approaches use approximation models of the real problem, which are cheaper to evaluate. These models, in essence, are simplified hypotheses of cause-effect relationships, and they replace high estimates with cheap approximations. In this thesis, we investigate surrogate models as wrappers for the real problem and apply Multi-objective evolutionary algorithm (MOEA) to find Pareto optimal decisions.

The core idea of surrogate models is the combination and stacking of several models that each describe an independent objective. When combined, these independent models describe the multi-objective space and optimize this space as a single surrogate hypothesis - the surrogate compositional model. The combination of multiple models gives the potential to approximate more complicated problems and stacking of valid surrogate hypotheses speeds-up convergence. Consequently, a better result is obtained at lower costs. We combine several possible surrogate variants and use those that pass validation. After recombination of valid single objective surrogates to a multi-objective surrogate hypothesis, several instances of MOEAs provide several Pareto front approximations. The modular structure of implementation allows us to avoid a static sampling plan and use self-adaptable models in a customizable portfolio. In numerous case studies, our methodology finds comparable solutions to standard NSGA2 using considerably fewer evaluations. We recommend the present approach for parameter tuning of expensive black-box functions.

Introduction

Motivation

To find solutions to real-world engineering problems, it is often necessary to find and apply adequate parameters. The search for these parameters is a computationally expensive problem and requires optimization. This search is often achieved with the help of the parameter tuning or in other words, parameter optimization process. Traditionally engineers adhere to the manual parameter tuning; they put the effort in searching the optimal objectives guided by experience and intuition. Nevertheless, many of these optimization problems have vast search spaces that could be handled only with automatic tools. These tools can extrapolate and highlight the most perspective parameters from infinite space. At the same time, they struggles with multi-criteria decisions that are critical for engineering problems. For examples: architecture design, test generation, tuning machine-learning algorithms could be stated as multi-objective problems. To understand the space of possible solutions, they are represented on the Pareto front; i.e., the subset of solutions that could be not improved in some objective without degrading another. Multi-objective algorithms allow to find out some Pareto optimal solutions. Still, we require a massive amount of evaluations to obtain those solutions, and that is inappropriate for expensive problems. A common approach in the reduction of the final cost of the optimization algorithm is to replace some expensive estimations with cheaper ones with the help of surrogate models. The conventional algorithms to extrapolate available results are Bayesian Regression model (Kriging), neural networks, Support Vector Regression (SVR) or Tree regressions (Decision) estimators. However, almost all optimizations approaches use static models or aggregate several instances of one model type. These approaches lack variability and cannot be finely tuned.

Ability to change the surrogate model strongly influences the optimization result. There is a demand for a strategy that allows us to combine multiple single-objective surrogate models to extrapolate multi-dimensional search spaces. This composition would make the surrogate model more versatile and capable of describing arbitrary optimization problems. Furthermore, it has been noted that the surrogate is domain-specific, and the same technique might perform differently on different problems. That is why extrapolation variability from the range of models improves final optimization results. However, only few researchers have addressed the solution of dynamic surrogate model selection.

Also, it is essential to obtain the necessary and sufficient number of samples to build an appropriate model. Unfortunately, to choose the optimum number of samples, it is required to have additional knowledge about a problem that is usually unknown. Moreover, arbitrary decisions on the sample size might be a reason that leads to inaccurate models and further wrong results.

Objectives

For this thesis, we have chosen two broad objectives that we tried to achieve. The first objective is to develop strategies that can dynamically compose the surrogate model from several single-objective models. The second objective is to enhance parameter tuning with the best practices from multi-objective optimizations techniques. Successful fulfilment of those objectives means an overall improvement in the area that concerns with optimization of expensive black-box functions. Also, success implies the possibility of application of developed tools to the broader spectre of real-world problems.

Research questions

To achieve our objectives we defined three research questions, which we answer in this thesis. Those research questions are:

- **RQ1**: Does the dynamic composition of different single-objective models improve the extrapolation of multi-objective problems?
- **RQ2**: Does a portfolio of surrogate models enhance optimization results?
- **RQ3**: Does a dynamic sampling plan help accelerate obtaining an optimal solution?

The purpose of this study is to provide a mechanism of a fined-grained models composition that allows making a superior multi-objective decision. Prerequisite for such solution is to reduce the number of actual evaluations while keeping the optimal quality of a decision.

Results overview

In this thesis, we introduce a modular structure for multi-objective parameter tuning that allows us to use various surrogate models and apply various optimization techniques. The overall results were achieved in several stages: 1) In response to RQ1, a composite model was implemented to combine several surrogate models. This solution made it possible to treat multi-objective extrapolation as a combination of single-objective models. 2) In response to RQ2 and RQ3, we developed a surrogate portfolio that enhances our strategy with the possibility to select surrogate models dynamically. Model selection is based on surrogate validation, which is also used as a criterion to check the sampling plan. An approach, combining all the aforementioned key features that answer the research questions was implemented under the name TutorM.

The evaluation results from a wide range of problems showed excellent results and advantage of TutorM strategy over comparable approaches: NSGA2 and Hypermapper 2.0. TutorM also provides a possibility of scalable solutions for problems that demand that.

The results can be used to improve the applicability of model-based optimization to a variety of expensive multi-objective parameter tuning problems.

Background

This chapter presents general background information needed to follow the terms and methods used in this thesis.

Parameter tuning

We start by introducing a parameter tuning problem. We consider a objective function as a black-box $f : \mathbb{S} \rightarrow \mathbb{R}$ with parameter and objective spaces (Figure 1). All feasible combinations of parameters define a parameter space which is intended to be a function input $\mathbb{S}$, and therefore all possible function outputs are defined as an objective space or $f(x), x \in \mathbb{S}$. The minimization of the fitness function could be defined as follows:

$$x^* = \arg\min_{x \in \mathbb{S}} f(x)$$

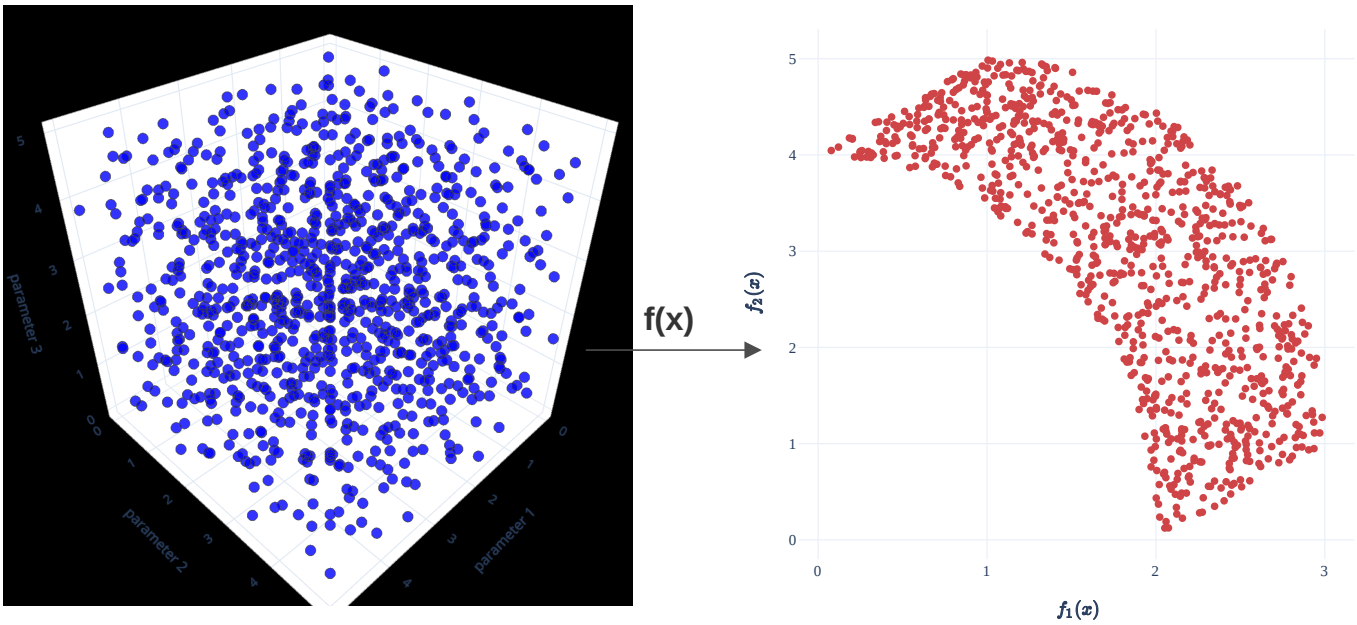


Figure 1: Example with uniformly distributed points in the parameter space (left) with the corresponding values of these parameters in the objective space (right). As can be noted from the results, the optimization of both objectives (f_1, f_2) is contradictory.

The mapping of all points from the parameter space to the points in the objective space is called a fitness landscape. The goal of parameter tuning is to find optimal points on this surface. Depending on the type of landscape, the optimization search often yields qualitatively different behaviour. A single criterion in the parameter tuning process might not be sufficient to characterize the behaviour of the configuration space correctly. Therefore, multiple criteria have to be considered. Typical examples of objectives are to enhance accuracy and performance or minimize runtime, error rate and energy. The parameter tuning process should improve the objectives of those in which none of the objectives can be improved without affecting another objective. By default, we consider minimization for all objectives. In this thesis, we are interested in the following properties of parameter tuning:

- Evaluation is expensive.
- Black-box function and the number of evaluated results are unknown.

- Multi-objectivity with global minimization.

Multi-objective optimization

Common parameter tuning problems require the simultaneous optimization of multiple, usually contradictory, objectives $f = (f_1(x), \ldots, f_k(x))$. Multi-objective optimization deals with such conflicts. It provides a mathematical algorithm with which to obtain an optimal design state that accommodates the various criteria demanded by the situation. Objectives are being improved simultaneously and gradually.

The solution to the multi-objective problem is a group of points which are placed on a Pareto front; i.e. the subset of solutions which are not worse than any other and better at least one goal [1]. A solution is called a Pareto optimum if no other solution dominates it. For example (Figure 2) a solution $A \in \mathbb{S}$ is said to dominate another solution $B \in \mathbb{S}$, denoted $A \preceq B$ if $f_i(A) <= f_i(B)$ for all $i = 1, \ldots, k$ and $f_i(A) < f_i(B)$ for at least one $i \in \{1, \ldots k\}$. All points on the Pareto frontier are not dominated by any other point in the objective space [2].

Awareness of the Pareto front allows appropriate decision-making and the importance of the criteria to be visualized. For the multi-objective problem, we consider the solution as points from the parameter space which lead to non-dominated results in the objective space. Improvement of the solution means finding a set of points that corresponds better with the real Pareto front.

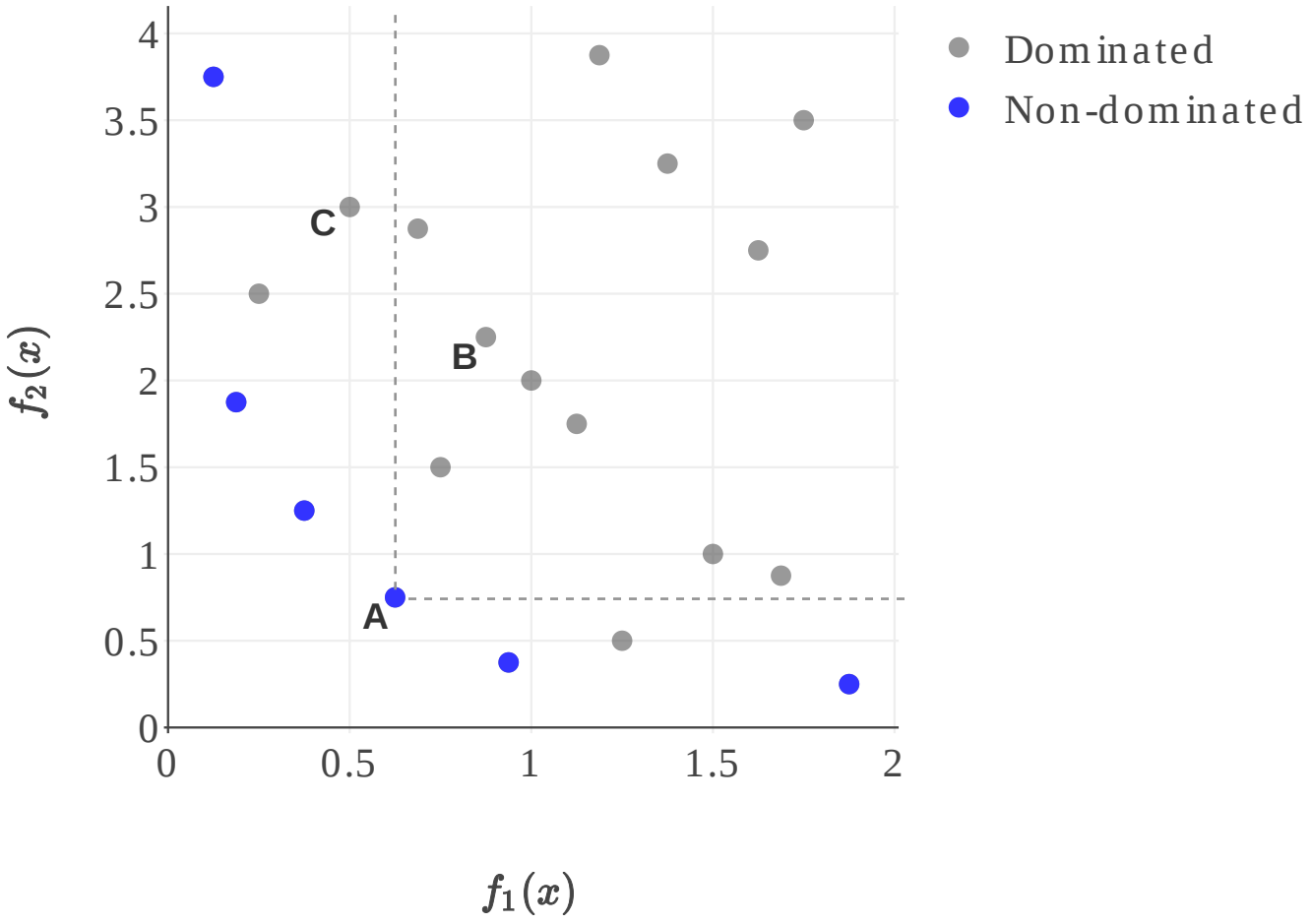


Figure 2: Example of non-dominated points. Point A dominates all points in the internal sector where B is located. Concerning point A, point C is not dominant because it has better value in f_1

Metrics for multi-objective solution

In a single-objective optimization, the quality of a given solution is trivial to quantify. When we consider a solution of a multi-objective problem as a Pareto optimal approximation, the comparison of these solutions is also a multi-objective task. The question of picking metrics for evaluation is essential for comparison of approximated solutions and for selection of the next appropriate set of configurations.

According to [3], a Pareto front approximation should satisfy the following criteria:

- The distance between the Pareto front and its approximation should be minimized.
- A wide distribution of non-dominated points is desirable.
- The range of the approximated front should be maximized, i.e., for each objective, a wide range of values should be covered by the non-dominated points.

The metrics for performance indicators are partitioned into four groups according to their properties [4]:

- *Cardinality.* Estimation of the number of non-dominated points.
- *Convergence.* Estimation of the closeness of a set of non-dominated points to the Pareto front in the objective space.
- *Distribution and spread indicators.* Measurement of the points distributed on the Pareto front approximation or of their allotment in extreme points of the Pareto front.
- *Convergence and distribution indicators.* Capture of both the properties of convergence and distribution.

According to [5], the spread metrics try to measure the areas achieved in a computed Pareto front approximation. This type of metrics is not very useful for comparison of algorithms or for evaluation of optimization convergence because spreadness is not related to improvement the objectives. However, they could be useful for a more detailed analysis of the optimization process or for composing Pareto frontier from several solutions.

The goal of the multi-objective optimization is to obtain an approximated solution set with the reference to the Pareto front, including the following subgoals:

- All solution sets are as close as possible to the Pareto front.
- All solution sets are as diverse as possible in the objective space.
- The proportion of the solution set to the evaluated set is as large as possible.
- Evaluate as few solutions as feasible.

For multi-objective optimization, an algorithm should produce a set of solutions which provide the optimal trade-off between the considered optimization objectives. Therefore, the performance comparison of Multi-objective optimization algorithms is based on their Pareto sets. In this study, four well-known metrics are used to quantify the performance of algorithms.

- **Hypervolume (HV).**[3] *Convergence and distribution indicator.* This metric represents the volume of the objective space which is filled by the individuals points of non-dominated solutions which belong to the Pareto front (Figure 3 4). Two points delimit the volume: one point represents the reference point r ($r \in R^m$) which is defined as the worst solution inside the objective spaceand the other one is the point which represents the Pareto approximation S, for all $z \in S, z \prec r$. The hypervolume metric is defined as follows:

$$HV(S, r) = \lambda_m(\bigcup_{z \in S}[z; r])$$

  where λ_m is m-dimensional Lebesgue measure. Calculating the hypervolume indicator is a computationally expensive task. Furthermore, in the case of a small number of dimensions and a low number of points, there are currently no known algorithms which might return the results fast enough for use due to the computational complexity which is $(|S|^{}) $ [6].

- **Non-dominated Ratio (NDR).** *Cardinality.* This metric is the ratio between the number of non-dominated points and the total number of the evaluated points. Higher values are preferred to lower ones.

- **Spacing [7].** *Distribution and spread.* This metric describe the distribution of Pareto points. As a wide range of similar metrics which are based on the distance to the nearest neighbour, spacing does not cover the holes in Pareto frontier and might compute the distribution in solution clusters.

- **Υ-metric (p-distance)**[8] *Convergence* The average distance of a set of points in relation to the Pareto front. Υ-metric is defined by

$$\Upsilon(S) = \frac{1}{|S|}\sum_{z \in S} g(z) - g(x^*)$$

  where g is a distance function and x^* is the Pareto-optimal decision vector. The lower the $\Upsilon(S)$, the closer the solutions of S are to the solutions of the Pareto front.

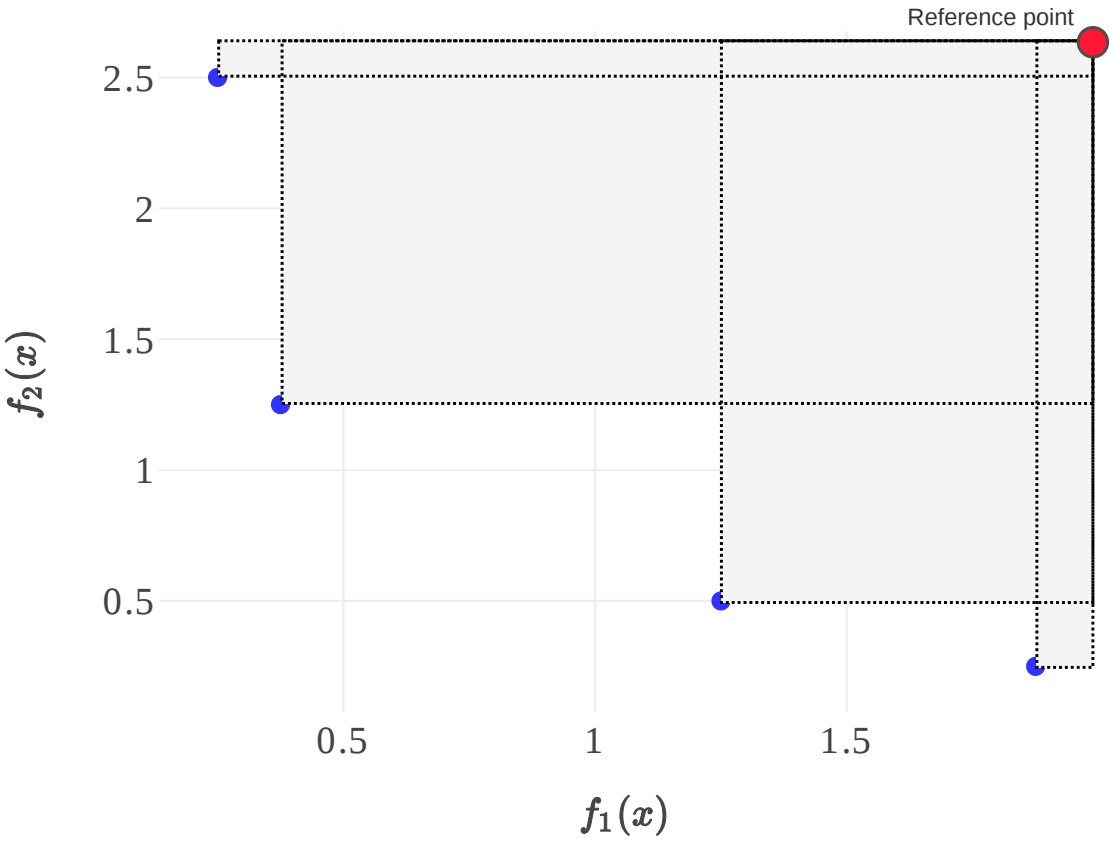


Figure 3: Higher hypervolume values may correspond to a better distribution of solutions or closeness to the Pareto front.

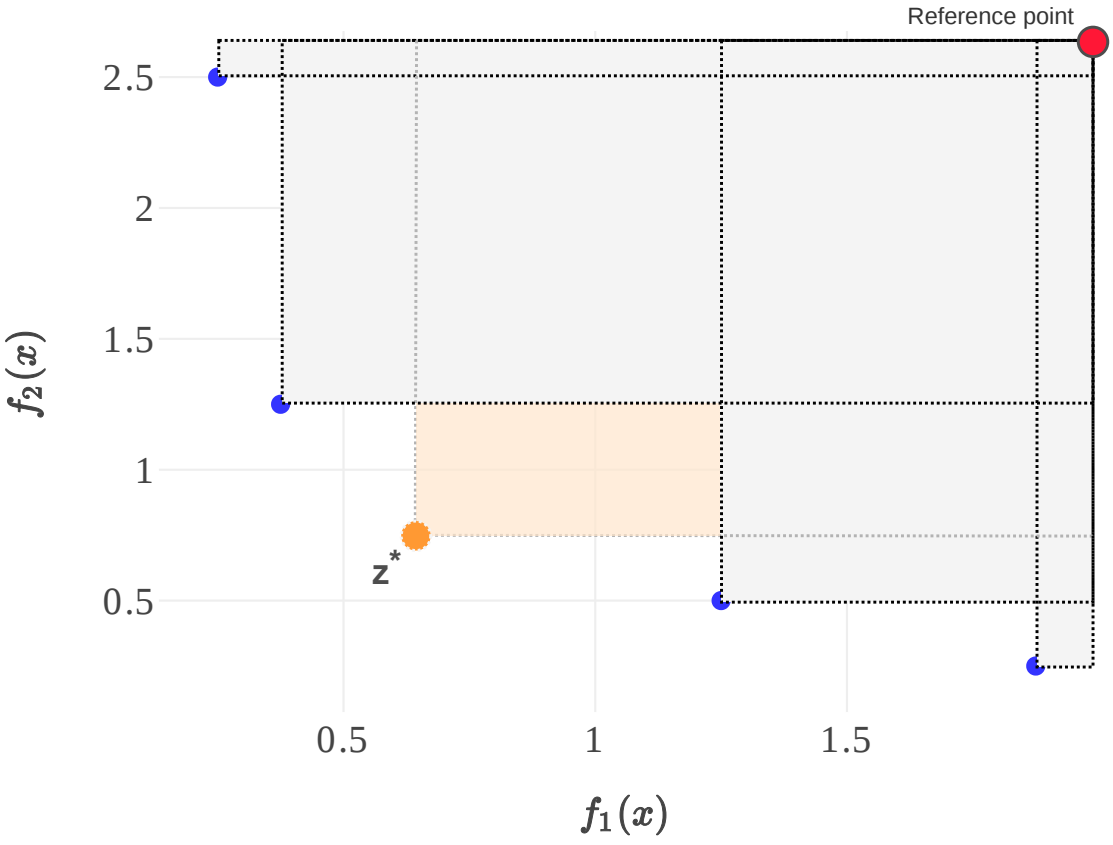


Figure 4: Example of hypervolume improvement with a new non-dominated point z* in the solution set.

Solving methods

Finding a Pareto optimal set is often impractical and computationally expensive. Therefore, many stochastic search strategies have been developed, such as: evolutionary algorithms, tabu search, simulated annealing and ant colony optimization. These algorithms usually do not ensure finding ideal trade-offs, but try to gain a satisfactory approximation. In this thesis, we interpret the Pareto optimum as the optimal solution to the problem. As mentioned, all points on the Pareto front are non-dominated, but not all non-dominated points are Pareto optimal. There are several basic approaches which provide information about how non-dominated points move toward a Pareto-optimal solution. Those approaches include: scalarization and Multi-objective evolutionary algorithm (MOEA). We describe them in next the sections.

Scalarization

The Scalarizing approach is a popular technique for creating a single-objective *parameterized* problem with the composite criteria from multiple objectives. The main advantage of scalarization is the possibility to use a broad range of single-objective techniques on this composite function. After optimization, one Pareto optimal solution is obtained, which depends on the initial scalarization parameters. The weighted-sum method is a well-known type of scalarizing technique. This approach concatenates the objectives into a single criterion by using weighted sum factors. There are difficulties in selecting proper weights, especially when there is no correlation in prior knowledge among objectives [9,10].

Some scalarizing techniques try to improve the exploration of the parameter space by assigning more "intelligent" aggregation to objectives. Such solutions can be fragile; they change dramatically with the modification of algorithm parameters. Moreover, the weighting method cannot provide a solution among underparts of the Pareto surface due to the "duality gap" for non-convex cases. This refers to the replacement of a non-convex original function with convex closure which missed non-convex parts of the initial landscape. Additionally, some scalarizing algorithms are very sensitive to the number of objectives. Analysis of the fitness landscape with different scalarizing techniques might be helpful in the optimization for solving expensive [9].

Multi-Objective Evolutionary Algorithms

Evolutionary algorithms form a class of heuristic search methods which simulate the process of a natural evolution. The evolutionary algorithm is determined by the two basic principles: selection and variation [11]. While selection reflects competition for reproduction and resources among individuals, the other principle, variation, imitates the natural ability to produce new individuals through recombination and mutation. Evolutionary algorithms are suitable for several problems, including multiple conflicting objectives and large and complicated search spaces [12,13]. Evolutionary optimizers explore populations of candidate solutions in each generation. Mutators can make changes to the current population. A select operator then picks the best mutants, which are then combined in some way to become a new population in the next iteration. However, Evolutionary algorithm (EA) still needs many evaluations of the black box system to solve the common multi-objective problem. This problem is crucial for the reason that most multi-criteria problems are expensive to estimate. This massive evaluation budget makes EAs infeasible for costly and multi-objective problems.

Surrogate optimization

Many expensive optimization problems have practical limitation on the number of possible estimations which standard optimization approaches spend very quickly. To get around this drawback, approximation models or surrogate models are often used. This technique is essential to reduce real evaluations by building a regression function based on already evaluated design points. The potential of the application of surrogates is based on the generalization of the entire search space and fast navigations there. This advantage should overrule the disadvantage in time required to build this approximation. In classical model-based optimization, a single surrogate model provides a hypothesis on the relation between the parameter and objective spaces. The approximation of the solution becomes

faster than the real evaluation, so the whole optimization process is accelerated. However, some extra time is needed to build and update the surrogate model during the optimization process. The surrogate model is used to find probable good candidates or to drop the low-quality individuals even before they are exactly evaluated, thereby reducing the number of exact evaluations.

In the literature, the term surrogate or model-based optimization is used in cases when, during the optimization process, some solutions are not evaluated with the original function but rather are approximated using a model of this function. Some of the most commonly used methods are the Response Surface Method [14], Radial Basis Function [15], Neural Network [16], Kriging [17], and Gaussian Process Modeling [18,19]. Surrogates are also used to rank and filter out the offspring according to Pareto-related indicators like a hypervolume [20], or a weighted sum of the objectives [21]. If the model is a single-criterion, it could be expanded to a multi-objective surrogate by considering each criterion in isolation and duplicating the model for each of them [22,23]. The surrogate model is either selected randomly or due to its popularity in the associated domain area [24]. Thus, there are still some open challenges related to the combination of meta-models, such as a definition of a selection criterion or combination techniques. Besides, there are no guidelines for using heterogeneous compositional models for different objective functions [24].

Multi-objective parameter tuning

The categorization of parameter tuning approaches based on the workflow of sequential model-based optimization is presented in Figure 5. The optimization process begins with an initial sampling plan. At this stage, it is necessary to collect fitness results or to evaluate the first parameters which are used to build surrogate models. For an initial sampling plan, or a Design of Experiments plan, the techniques of Latin hypercube sampling, Sobol sampling or random sampling can be used.

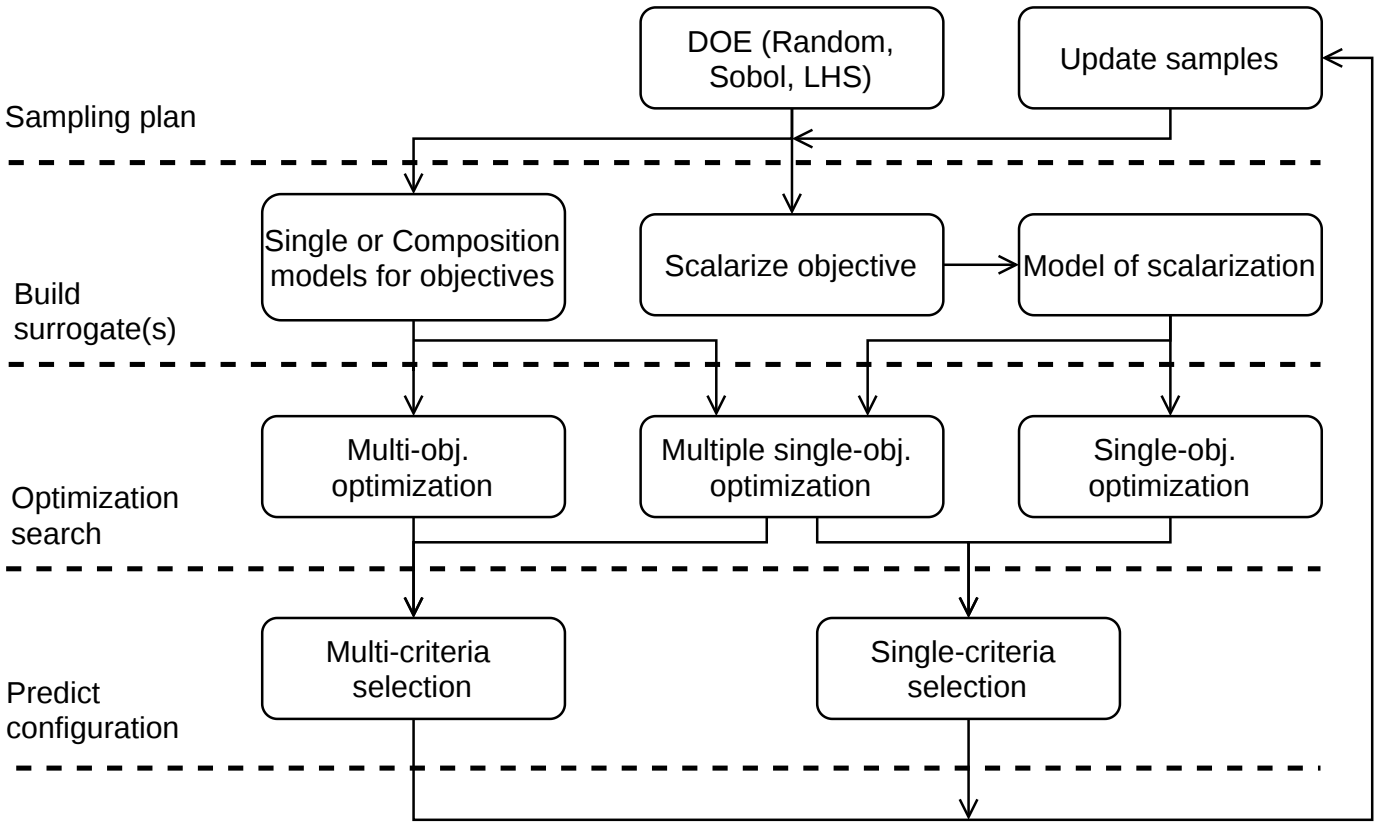


Figure 5: Phases and tasks within a generalized multi-objective parameter tuning

There are two established approaches to extrapolate the samples: 1) to scalarize objectives and produce a surrogate model of this scalarization (In this case, the multi-objective problem transforms into a single-objective one); 2) to keep the original dimensionality of the problem and apply one or several models to hold and infer on the problem landscape.

The next step, optimization, is the search for optimal points within the surrogate model. At this step, the solutions which might be Pareto-optimal are found. In predict configurations phase, sorting and selection are carried out. In the case of multi-criteria selection, it is necessary to select the required number of points which are optimal for all objectives. Theoretically, all non-dominated points are equal regardless of the order they are chosen in. The possible prediction of several parameters instead of a single one is an advantage that improves the exploration of the parameter space and parallelizing of fitness evaluation. The required number of points allows the sample set to be estimated as well as updated. Optimization iterations continue until a stop condition is satisfied.

Variability and extensibility are essential for configurable parameter tuning, as in a software product line. To this effect, the optimization round is consistent and universal. As shown in Figure 5, the potential to reuse components in a workflow is enormous. The same single-objective models can be equally applied to various types of problems in multi-/single-objective optimization. An optimization algorithm weakly depends on the type of surrogate model. By dynamic duplication of the surrogate model or even by the creation of several surrogate hypotheses, we aim to improve the parameter tuning to multiply criteria on-the-fly.

Domain-specific problem

Surrogate models are domain-specific in case of intention to find the best solution with less effort. On the one hand, the surrogate model could perform well while extrapolation one class of problems and guide to the optimal solution. On the other hand, this model could be a reason for a significantly degrading result in another type of problem. That is why the authors prefer using several surrogate models and don't select one for all use cases [24].

It could be an interpreted as a Non-Free-Lunch theorem in model-based optimization. If we extend this argument, then the same optimization problem in different parameter tuning iteration could be interpreted as another optimization problem. In order to reduce an effort and to increase the convergence of an algorithm, we should change the surrogate model depending on how many samples we have. This leads us to the usage of a portfolio with surrogate models. On each optimization iteration, the portfolio tries to build and select several models with the best performance. As a negative consequence, building several models introduces an additional overhead into the optimization time.

Initial sampling set

Initial samples should provide maximal information to build a useful surrogate model. Indeed, the overall result of the optimization depends primarily on how accurate the initial assumption is; an invalid optimization model makes all further optimization results irrelevant. The concept of surrogate validity guarantees that the model can confidently be used to find optimal solutions.

If no valid models are obtained, it is better to use the initial design than to be guided by an incorrect model. With an increasing sample size, in case of proper fitting, a better surrogate model is obtained, and the better results in optimization are reached. Moreover, the initial sample size might be too big, which is a mere waste of resources.

Discussion

Most methods for parameter tuning optimization are related to surrogate-based optimization. One of the main advantages of this approach is the speed of evaluation across the entire search space and possibilities to apply a broad range of optimization techniques. However, there are also disadvantages, such as the extra time required to select and build this surrogate model. It would be desirable to have a way to combine several surrogate models which are sample- and domain-dependent.

Related Work

This section overviews other studies in the area of surrogate-based multi-objective optimization and related approaches of other types of optimization.

Comparison criteria

Many existing approaches can be categorized as multi-objective optimization approaches. Therefore, the comparison criteria for a clear and concise definition of the approach are introduced in this thesis:

- **Sampling plan** specifies the size of a sample set from which to build a surrogate model and the sampling strategy that will pick these samples. The sampling plan can be static when decisions about samples are made ahead of time or it can be dynamic when they depend on optimization success.

- **Surrogate type** is describe extrapolation models and a composition strategy to combine these models. In this context, variability indicates that the surrogate model is exchangeable and can be selected for a specific problem. The extensibility of a surrogate refers to the ability to grow and improve general extrapolations for a particular problem.

- **Optimization algorithm** is applied to find the optimal points in the search space. The architecture of the optimization algorithm and the surrogate model can be tightly coupled (OSI) either when the surrogate model is nested in the optimization algorithm, or when they perform flat architecture with ASO (Figure 1).

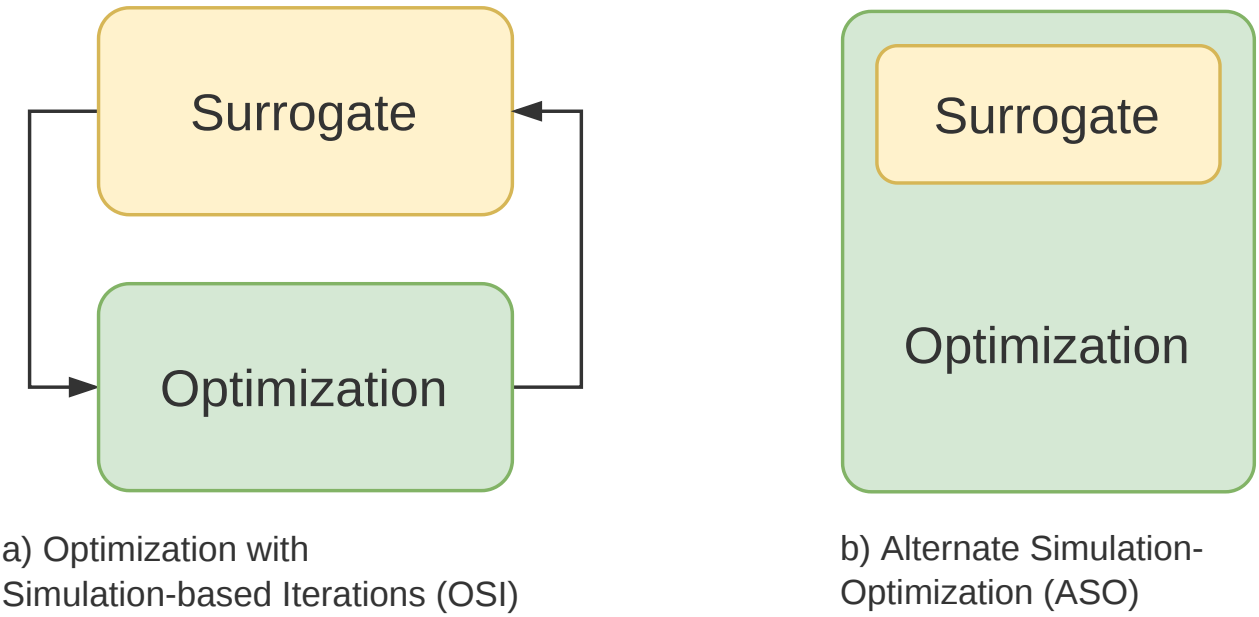


Figure 1: Example of a possible combination the optimization algorithm with the surrogate model [25]

- **Scalability** refers to the dimensionality of problems that were applied to analyze the performance of the algorithm.

- **Multi-point proposal** is a property of yielding the required number of multi-objective solutions.

Almost all related works of parameter tuning could be categorized as Sequential model-based optimization (SMBO) [26]. The general looks as follows (Figure 2):

1. Start with the initial sample plan of evaluation points.

2. Build a regression model to provide a hypothesis about the relation between parameters and objectives.

3. Use the built surrogate model as a parameter for optimization strategy. The solutions from the optimization algorithm are new promising points for evaluation.

4. Evaluate the new predicted points and add them to the existing samples.

5. If the stop criteria are not met, repeat optimization with the updated sample set.

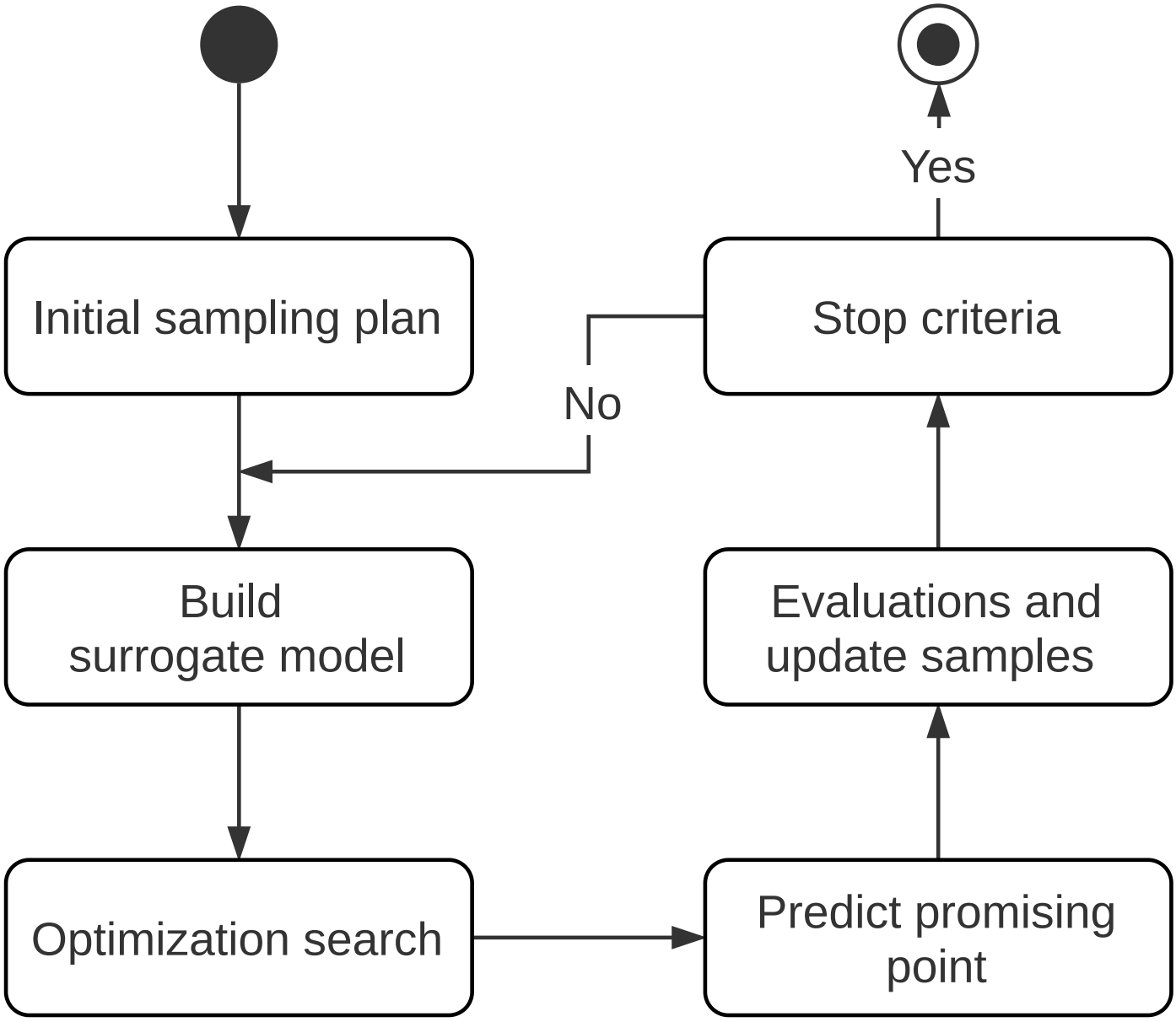


Figure 2: General Sequential model-based optimization}

We present an overview of previous work on model-based multi-objective optimization. We begin the project for model-based optimization (mlrMBO) and continue with related algorithms including various optimization improvements.

Platforms and frameworks

There are many different projects that can perform multi-objective optimization. Frameworks provide multiple multi-objective algorithms, performance metrics, built-in test problems and extra tools such as plotting and benchmarks. Some frameworks focus on efficient calculations and parallelization [27], others on implementation of modern multi-objective algorithms [28,29] and on support of plenty of model-based optimization algorithms [30].

mlrMBO

[30] is a modular framework for model-based optimization of expensive black-box functions. MlrBO extends the SMBO procedure to a multi-objective problem with mixed and hierarchical parameter spaces. Modular structure and integration with *mlr*[1] library allows all regression models to be used and compositional techniques such as bagging and ensembling to be applied. The authors implemented four different model-based multi-objective algorithms that are categorized in three classes: 1) Scalarization-based algorithms that optimize a scalarize version of the black-box functions (). 2) Pareto-based algorithms that build individual models for each objective and complete multi-objective optimization (MSPOT [31]). 3) Direct indicator-based algorithms which also fit individual models, but perform a single objective optimization on the collection of all models (SMS-EGO [32], ϵ-EGO [33]). A special feature of the mlrMBO framework is multi-point proposal prediction on each iteration. However, it does not provide a combination of different surrogate models into one model.

BRISE 2.0

[34] is a software product line for parameter tuning. The core topics of their approach are extensibility and variability with key features such as 1) A *repeater* which automatically decides on the number of repetitions needed to obtain the required accuracy for each configuration. 2) *Stop condition*, which validates a solution received from the model and decides whether to stop the experiment. 3) Association of a sampling plan with model prediction validity which provides a *flexible sampling plan*. The main advantage of this sampling plan is that it requires less initial knowledge about the optimization problem. Extrapolation model for parameter prediction is exchangeable and it combines surrogate and optimization strategies in each implementation. Several general models, such as polynomial regression surrogate with local search and Bayesian optimization with bandit-based methods strategy(BOHB [35]), are implemented. However, the models lack variability and should be designed from scratch for new domain-problem.

SMAC

[36,37] is a framework for parameter tuning with Bayesian Optimization in combination with a simple racing mechanism to decide which of two configurations performs better. SMAC adopted a random forest model and Expected Improvement (EI) to model conditional probability. It applies a local search with several starting points to pick configurations with a maximal value of EI. These points improve the exploration possibilities of SMAC in the search space with higher uncertainty and an optimal value of the objective mean. An interesting feature of SMAC is its support of the termination of unfavourable evaluations that are slowing down the optimization process. However, SMAC is limited to single-criteria optimization and uses a predefined sampling plan.

Hypermapper 2.0

Luigi Nardi et al. [23] presented a multi-objective black-box optimization framework that solves complex parameter tuning problems with the combination of search space, expensive evaluation, and feasibility constraints. The proposed approach can be classified as a direct indicator-based algorithm with constraint validation. During this type of optimization, several identical individual models for each objective are built, then a single-objective optimization is performed on the aggregation from all these models. The authors further extended this idea to predict no feasible configurations with an extra surrogate model.

The Hypermapper 2.0 is based on a surrogate of a random forest. The random forest combines several weak regressors on a subset of samples to yield accurate regression and effective classification models. After scalarizing values from models, framework applies an Bayesian optimization (BO) method to find an optimal value. Since using a single weight vector would only find one point on the Pareto optimal front, a weight vector is chosen randomly for each iteration, ensuring that multiple points on the Pareto optimal front are found. The key features of this approach are the possibility to use prior knowledge, support real variables, predict feasibility, and excellent final adaptation of the implementation to embedded devices. The authors reported that Hypermapper 2.0 provides better Pareto fronts compared to state-of-the-art baseline, i.e. better competitive quality and saving evaluation budget.

Model-based multi-objective algorithms

Fixed optimization components can make general optimization ineffective in the face of different problems. The flexibility achieved by the surrogate construction methodology and multi-objective (MO) algorithms can help to achieve solutions closest to the real Pareto optimal front.

ParEGO

is a scalarization-based multi-objective algorithm [22]. It was developed as extension, which can encompass multi-point proposal, of a classic single-objective algorithm EGO[26] of Jones et al. In its core lays repetition of an algorithm execution with randomly changed scalarization weights for each iteration. This algorithm is based on the Kriging (Gaussian process regression) model and multiple single-objective optimization processes on scalarized objectives. Several runs with random weights guarantee that multiple points on the Pareto optimal front are predicted. This algorithm and its modification are implemented in mlrMBO[30].

An Evolutionary Algorithm(EA) with Spatially Distributed Surrogates

Amitay et al.,[38] presented an EA with spatially distributed surrogates (EASDS). Surrogate models use Radial Basis Function Networks, periodically validating and updating each subset of samplings points. This generates a complex ensemble surrogate with approximations in local areas of the search space. Spatially Distributed Surrogate models are created for all objectives and then evaluated by NSGA-2 [38]. The authors report that their approach achieves better results than single global surrogate models, multiple surrogates. Of note, the authors evaluated their algorithm only on bi-objective problems.

A hybrid surrogate-based approach for evolutionary multi-objective optimization

Rosales-Pérez et al.,[39] proposed an approach based on an ensemble of Support Vector Machines (SVM). The authors describe a model selection process or hyperparameters selection of SVM based on a validation technique and a further injection into the surrogate ensemble. Incremental development of the ensemble includes new information obtained during the optimization and old evidence stored in previous models. The training of a new model represents search on the SVM grid in order to find a kernel with the lowest expected generalization error. This paper presents a model selection process for determining the hyperparameters for each SVM in the ensemble.

Evolutionary optimization with hierarchical surrogates

Xiaofen Lu et al. [40] apply different surrogate modelling techniques based on the motivation to optimize expensive black-box functions without any prior knowledge of the problem. They used a pre-specified set of models to construct hierarchical surrogates during optimization. Also, to verify the surrogates, the general accuracy of the high-level model was used. The process of the proposed method involves splitting the accumulated training samples and model-based optimization, which means that the sample plan is static and requires prior information about the problem.

The authors show that the hierarchical surrogate structure can be beneficial when the accuracy of the high-level model is greater than 0.5. They also noticed that a single modelling technique might perform differently on different problem landscapes. This motivates us to use a pre-specified set of modelling techniques (portfolio with surrogates).

Population-based Parallel Surrogate Search

Akhtar et al.,[41] introduce a multi-objective optimization algorithm for expensive functions that connects several iteratively updated surrogates of the objective functions. The key feature of this algorithm is high optimization for parallel computation and stacking predictions from the set of Radial Basis Function (RBF) models. The algorithm combines RBF composition surrogate, Tabu, and local search around multiple points.

GALE: Geometric Active Learning for Search-Based Software Engineering

Krall et al.,[1] developed an algorithm that uses principal components analysis (PCA) and active learning techniques to perform step-by-step approximation and evaluation of the most informative solutions. The authors notice that MOEAs are not suitable for expensive multi-objective problems because they push a set of solutions towards an outer surface of better candidates, costing many function evaluations. The fundamental idea of the proposed approach is to choose the most informative solutions from a large set of options. This is accomplished by dividing the functional landscape into smaller regions and evaluating only the most informative samples from the regions. As a result of the division of the functional landscape, function evaluations are spared since only a few of the most informative points from the region are evaluated. A drawback of this approach lays in static implementation with MOEA/D.

Table 1: The comparison of related approaches. The component behaviour: S - static, V - variability, E-extensibility, D - dynamic.

Approach	Multi-objective	Sampling plan	Extrapolation models	Composition strategy	Optimization	Mixed search space	Multi-point proposal	Scalability
mlrMBO [30]	✓	static	E	V	V	✓	✓	✓
BRISE 2.0 [34]	✗	flexible	V	V	V	✓	✓	✓
SMAC [36,37]	✗	static	S	S	S	✓	✗	✓
Hypermapper 2.0 [23]	✓	static	S	S	S	✓	✗	✓
ParEGO [22]	✓	static	S	S	S	✗	✗	✓
Distributed Surrogates, EASDS [38]	✓	static	S	E	S	✗	possible	✗
Hybrid surrogate [39]	✓	static	S	E+D	V	✗	possible	✓
Hierarchical surrogates [40]	✗	static	E	V	V	✗	possible	✗
Parallel surrogates, MOPLS [41]	✓	static	S	E	S	✗	✓	✗
GALE [1]	✓	static	S	E	S	✓	possible	✓

Scope of work

As shown in Table 1, surrogate or model-based optimization suit expensive black-box problems. A summary of the related works that we have discussed is shown in the table above. Nevertheless, model-

based approach still has limitations:

- Multi-objective hypothesis. A *limited number* of surrogate models can handle with several parameter and objectives, but they struggle when these parameters and objectives become larger.

- *Surrogate is domain-specific.* Currently, to improve and reach the best prediction, we need to know the objective surface in order to apply a specific surrogate. Universal surrogates might gain optimal results but may not be the most reliable [40,42].

- The quality of predictions depends on *the number of samples* we use for a specific type of surrogate. There is a trade-off between the reduction of sample size and maximization of prediction quality. Overfitting, as well as underfitting, can guide optimization in a wrong direction.

- Often, optimization algorithms and surrogate models are *coupled* extremely tightly. This interdependence makes the general approach biased against specific problems. Reimplementation of these algorithms for each usage scenario becomes time-consuming and error-prone.

After surveying the aforementioned literature, we derive the following research gaps and questions.

- **Surogate combination**

  Previous research has shown that surrogate model selection can profoundly impact the quality of the optimization solution [39]. Furthermore, it has been noted that the surrogate is domain-specific, and the same technique might perform differently on different problems [40]. The overall research gaps of the surrogate model's flexibility can be divided into the following subproblems:

  1. The dynamic combination of different single-criterion models is crucial to solve the multi-criteria problems. It must be feasible to substitute the type of nested surrogate model for an arbitrary criterion component. This would make the surrogate model more versatile and capable of describing arbitrary optimization problems. Therefore, technology that would allow the creation of surrogate compositional models is necessary. Hence we can identify the first research question **[RQ1]**: Does the dynamic composition of different single-objective models improve the extrapolation of multi-objective problems?

  2. After scrupulous investigation of presented works we conclude that access to the range of models improves final results. Unfortunately, most authors used only one type of model with various hyperparameters. Therefore, the next research question **[RQ2]**: Does a portfolio of surrogate models enhance optimization results?

- **Sampling plan**

  A static plan requires additional knowledge of the surface of the problem, which is usually not possible to obtain. Moreover, arbitrary decisions of the sample size might be a reason that leads to inaccurate models and further wrong results. These problems may occur because the surrogate model is overfitted or underfitted, and the result is a waste of samples and resources. Consequently, the last research question **[RQ3]**: Does a dynamic sampling plan help accelerate obtaining an optimal solution?

To our knowledge, there have not been studies of the comprehensive surrogate combinations and the dynamic sampling strategy. Therefore, we focus on the improvement of compositional surrogate models for multi-objective problems. We aim to apply surrogate models to problems that are expensive, multi-objective, and derivative-free/black-box systems without constraints.

The following goals must be achieved:

1. Find diverse multi-objective solutions with minimal distance to real the Pareto front.

2. Develop modular and extensible architecture.

3. Improve the backward computationally with single-objective parameter tuning.

4. Reduce the evaluation budget.

Compositional Surrogate

This chapter introduces a general idea which overcomes the limitations of model-based optimization discussed.

As mentioned previously in section @{sec:background}, fixed components can make the optimization process ineffective. That is why flexibility and variability must be introduced for each optimization step. Our concept focuses on the combination of surrogate models to effectively extrapolation required problems.

Combinations of surrogate models

Let us address the main issue we have observed in multi-objective optimization. The issue is that the solution techniques and parametric selections are usually problem-specific. In addition to that, most surrogate model implementations are static which imposes limitations on existing solutions. We tackle this challenge by improving model variability *in* a surrogate model (compositional surrogate) and model extensibility *with* surrogate hypotheses (surrogate portfolio). Also, we should address an additional question of solution scalability, which is related to the compositional surrogate model. There are some tasks that could require new algorithmic approaches when we add more parameters to them. Therefore, there exists a demand for scalable solutions. We discuss our approach to problems described i+n the next sections.

Compositional Surrogate Model [RQ1]

The concept of the compositional surrogate is the combination of multiple simple models to approximate several independent objectives at the same time. In this model, composite and conventional surrogates have a unified interface that permits us to implement a *composite design pattern*[43]. This design pattern then allows us to operate uniformly with the individual and multi-objective surrogates.

Additionally, a significant advantage of compositional surrogates is a possibility to extend single-objective parameter tuning to multi-objective optimization. This possibility provides the opportunity to reuse single-criterion models for multi-criteria optimization and dynamically reconstruct problem representation from mixed parts. We define *compositional models* as models that combine *various* sub surrogate models for each optimization objective. The *surrogate hypothesis* refinement is also used to emphasize that the surrogate model can completely describe all criteria from the objective space.

The compositional surrogate has multiple opportunities for variability that outperform static models in the face of real black-box problems. For example, choosing a specific set of models is a representation of knowledge about the subject area. If expectations during optimization are not met, the compositional model can be partially updated, which saves time. In contrast, a static model would need to be completely replaced. Such changes might be demanded by newly obtained results or the increased dimensionality of optimization space.

Scalability

The ability to scale the optimization solution can be considered an adaptation to an unknown problem. Solution scalability is the ability to solve problems with a high number of dimensions in parameters and objectives spaces.

Multiple works[1,44] have practically demonstrated that scalability is a problem for surrogate models and optimization algorithms. As an illustration, popular surrogate models such as Gaussian process regression (Kriging) [26] struggle with high dimensional samples but provide excellent results in smaller dimensions. Therefore, another advantage of the composite surrogate model is evident; it provides variability for the extrapolation of scalable search space.

Surrogate model portfolio [RQ2]

In addition to the dynamic variability in the compositional surrogate, we combine several surrogate hypotheses in a surrogate portfolio to dynamically choose one that is best suited to the specific problem. The unified interface and the ability to integrate models into a composite architecture make it possible to uniquely select and combine composite models side by side with static multi-objective models.

Without information about a given problem, it is difficult to say which surrogate hypothesis would be better. Therefore, the model should be selected during optimization based on its usefulness (validity). The validation process involves checking how well the model extrapolates unknown data. For such validation evidence, a small portion of samples should be sacrificed. This process of data sacrifice give us two separate data sets: one for model building and another for its testing. The test score obtained from the test set is used to evaluate the model's accuracy and, accordingly, the quality of the possible corresponding solutions. The validation process allows us to evaluate surrogate models based on how they summarize an unknown problem.

For an optimization algorithm, a portfolio can be considered either as a single model or as a collection of models. This property allows us to determine which optimization algorithms are applied to which surrogate models and how to combine such solutions. Such dynamic variability makes the multi-criteria optimization process also scalable and flexible.

Besides, the surrogate portfolio does not limit to use the latest state-of-the-art optimization algorithms and surrogate models together.

Sampling plan [RQ3]

After surveying the aforementioned related works, we learned that only BRISE use dynamic sampling plan while other approaches use a static sampling plan that determines an optimal number of initial samples using an outside oracle. On the contrary, BRISE is applied dynamic but still domain-dependant sampling plan. Although in most cases, we cannot receive any guidance on an unknown problem. Thus, we need a dynamic sampling plan which adapts to a specific use-case.

To obtain the adaptive sampling size we need to bind the sampling design to a validation process for the surrogate model. An optimization process is guided by sampling design when none of the surrogate models are valid (Figure 1). Validity means that the surrogate approximation can be useful for efficient global optimization.

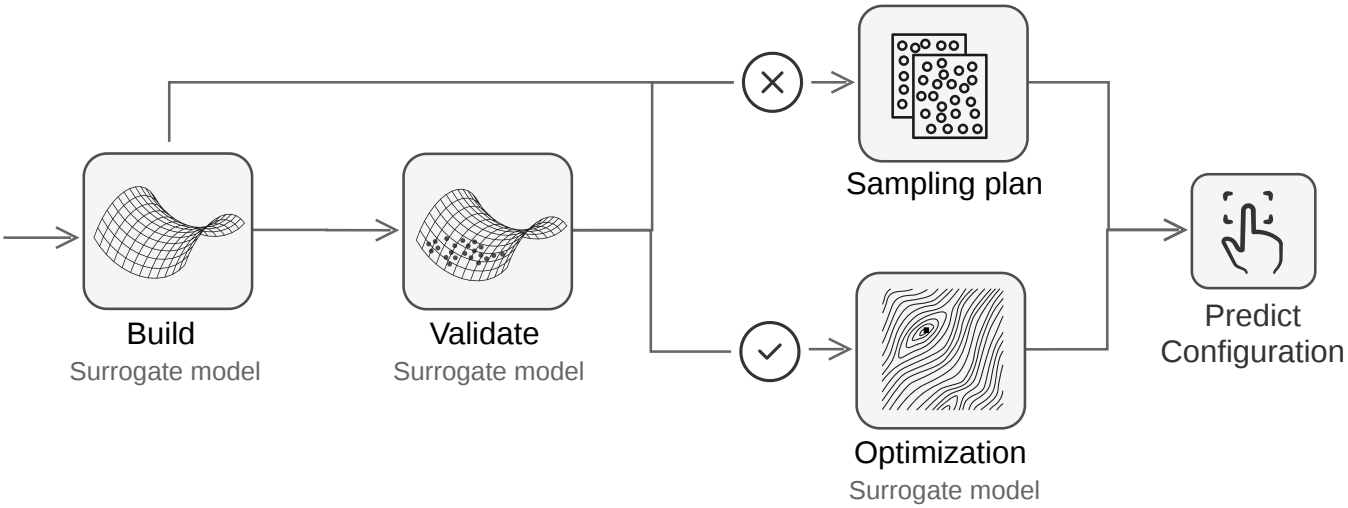


Figure 1: Concept of a sampling plan dependency and model validation. A sampling plan is used if there is no valid model that can be useful for optimization purpose.

Surrogate Validation

In the context of sequential model-based optimization, a common mistake is studying the accuracy of the evaluation of the global search space instead of the search space region of interest. That is why basing the evaluation of surrogate validity only on the coefficient of determination(R2) is incorrect [23]. The global accuracy metric can be used as a threshold value above which the model becomes invalid even with additional estimations.

It is necessary to sacrifice a small portion of samples to check the surrogate model's quality. Based on validation results, we can discard inadequate models and consider the solutions from valid models only. If none of the models are valid, the best decision is to now make a prediction from the sampling plan. This decision is repeated until a valid surrogate model is obtained.

Validation should show how well the model extrapolates the available experiments (variance) and how well it can evaluate the data that is not seen (bias). The central concept in surrogate validation lies in the adaptation of the best machine-learning approaches for the evaluation of a model's performance.

We select surrogate models based on accuracy in the test set, but the selection may not be correct if only one test set is taken into account. Increasing surrogate complexity can lead to obtaining wrong conclusions in a later stage of optimization (Figure 2). This property cannot be neglected in evaluating a surrogate's validity.

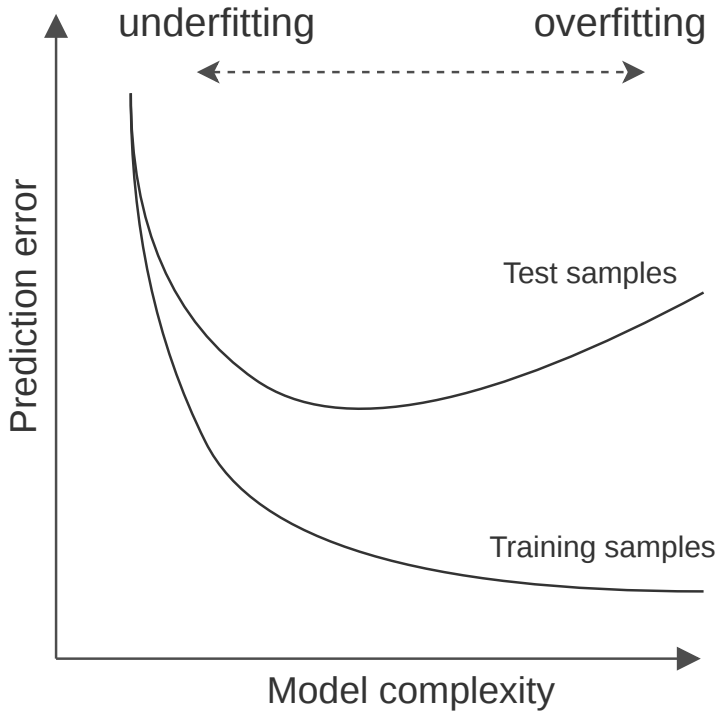


Figure 2: With rising model complexity overfitting on the training set becomes more likely [45,46]

It is necessary to perform the validation in a few phases with separate test sets. The validation process requires two separate test sets: the first one to select surrogate models and the second one to test those selected models. In addition to those two test sets there is a third set for building surrogate models. We obtain those sets by dividing all available samples.

Partitioning the available samples into three sets drastically reduces the number of points that can be used for building the model. A small number of build samples could lead to inadequacy of the model. Also, results can depend on the selective random decision for sample splitting.

However, partitioning the available samples into three sets, are drastically reduce the number of points which can be used for learning the model. Moreover, results can depend on a selective random decision for the samples splitting. The solution is might be cross-validation(CV, Figure 3). This is a procedure that avoids a separate validation set and divides test samples to *k* equal folds. Set of folds are used to train model and in *k* rounds, a new fold selected as a test set. The performance measured by cross-validation is the averaged over the values computed in the loop. This approach can be computationally expensive but requires fewer samples.

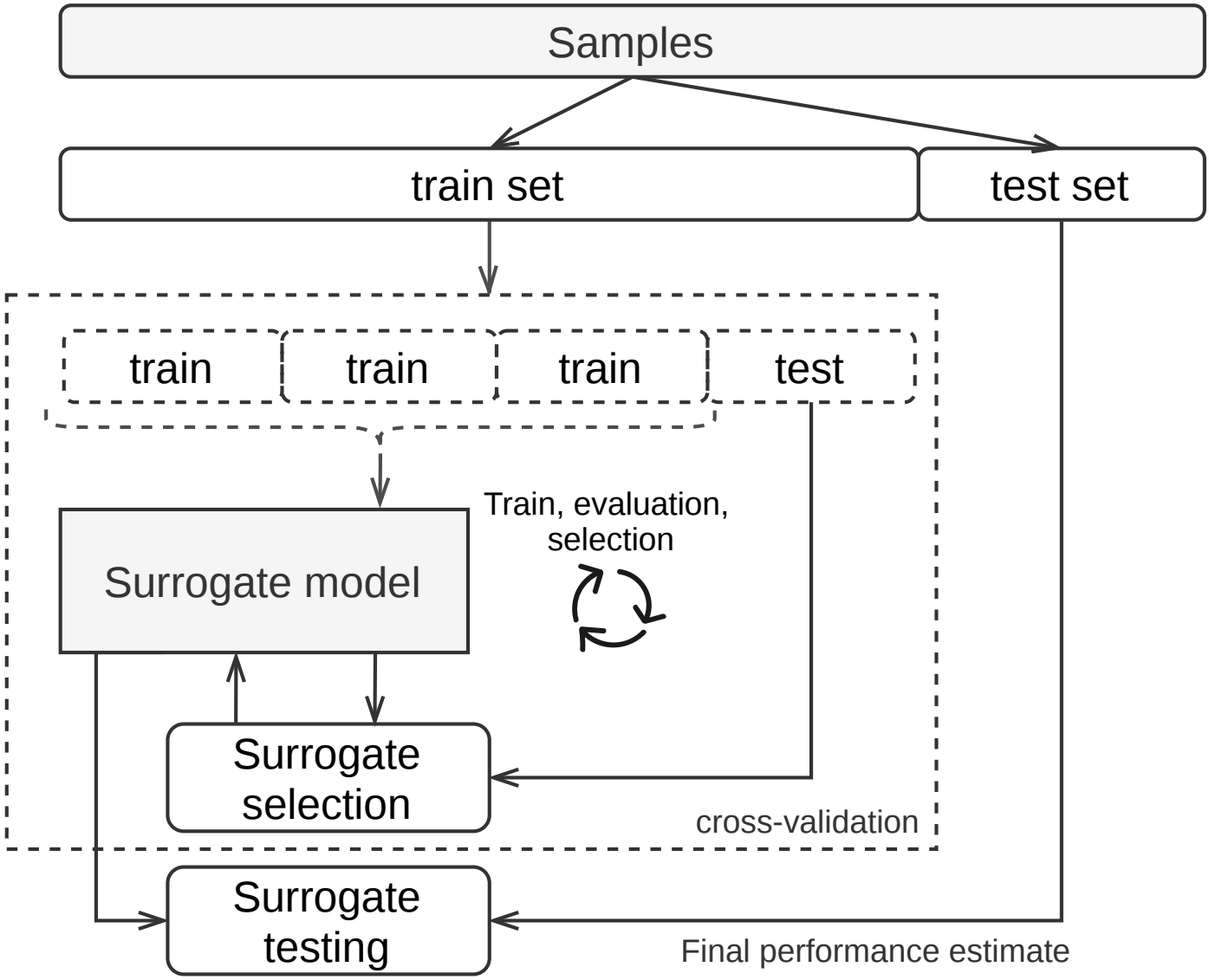


Figure 3: Surrogate models validation. Cross-validation loop performs model selection based on global accuracy. Acceptable models are tested with a focus on optimization region.

To summarize, the model validation is performed in two stages:

1. **Cross-validation.** We check overall accuracy of the surrogate model extrapolation. Also, we discard models that do not achieve the necessary threshold.

2. **Surrogate testing.** We demonstrate the accuracy of the selected models and the corresponding assessment of possible solutions.

We decide which surrogate models to choose based on the information from all stages. If the model does not achieve a sufficient threshold, it is rejected as not valid. If there is no valid model, the assumption about the next configuration is accepted from the sampling plan (Figure 1).

Discussion

Toward answering our research questions, we propose the dynamic combination of surrogate models and a dynamic sampling plan based on surrogate validation.

- **RQ1:** For the dynamic combination of several surrogate models, it is necessary to implement a surrogate compositional model. This design allows uniform handling of individual and compositional surrogate models.

- **RQ2:** The combination of surrogate models in the portfolio is realized through the compositional model and stepwise validation.

- **RQ3:** The sampling plan is chosen to explore new random points when there is no valid model. This relation means that the sampling plan directly depends on whether we have a surrogate model that is capable of describing the optimization problem.

As a result, we extend the idea of classic SMBO using dynamic model selection and stepwise validation to obtain a multi-objective solution on various problem landscapes.

Implementation

In this chapter, we present the implementation details of our proposed decisions.

In order to achieve goals from previous sections, it is necessary to formulate technical requirements for implementation. After a thorough study of the literature [47], we put forward the following requirements:

- **Components.** To meet the needs of flexible architecture, we need to divide the optimization workflow into logical steps and abstract them. These abstractions are interpreted as easily replaceable components. Only in the case of homogeneity of optimization processes with the standard interfaces, it is possible to scale the optimization approach to multi-objective tasks.

- **Separation of concerns.** In order to ensure more considerable optimization variability, it is necessary to evaluate the optimization steps independently.

- **Non-proliferation of classes.** To improve the compatibility of our solution with other frameworks, we need to use a simple structure to share information.

The following frameworks were used to fulfill the criteria.

Scikit-learn [48] is one of the most popular machine learning framework that accomplishes with a variety of learning tasks. The crucial features are excellent documentation and reusable components in various contexts. Extensibility and consistent interfaces resulted in large and active community of library. Scikit-learn integrates well with many other Python libraries.

pygmo2 [27] is scientific library with an effective parallelization for local and global optimization. Key features of this project are efficient implementations of bio-inspired and evolutionary algorithms and unified interface to optimization algorithms and problem definitions.

Next, specific optimization steps will be discussed.

Compositional surrogate

The Composite Surrogate provides the ability to aggregate several simple models to promote multi-objective extrapolation.

To achieve this goal, the Model-union class (Figure 1) class was implemented. It is implement *a compositional design pattern* [43] where several heterogeneous models could be combined. This class is as meta-model that wraps and aggregate surrogates and could be combined in a tree structure. Such an architectural solution is needed to improve the scalability of surrogates as components

A parent class that combines multiple models can combine their approximations in several ways:

- **Stacking.** It is an ensemble approximation technique which average obtain results from each child model. The child regression models are trained based on the whole training samples.

- **Split y.** A straightforward technique to combine several regression models in multi-label prediction case. Each child surrogate is trained on the entire dataset, including only one objective of interest. This functionality allows as to produce multi-objective compositional surrogate from combinations of single-objective models.

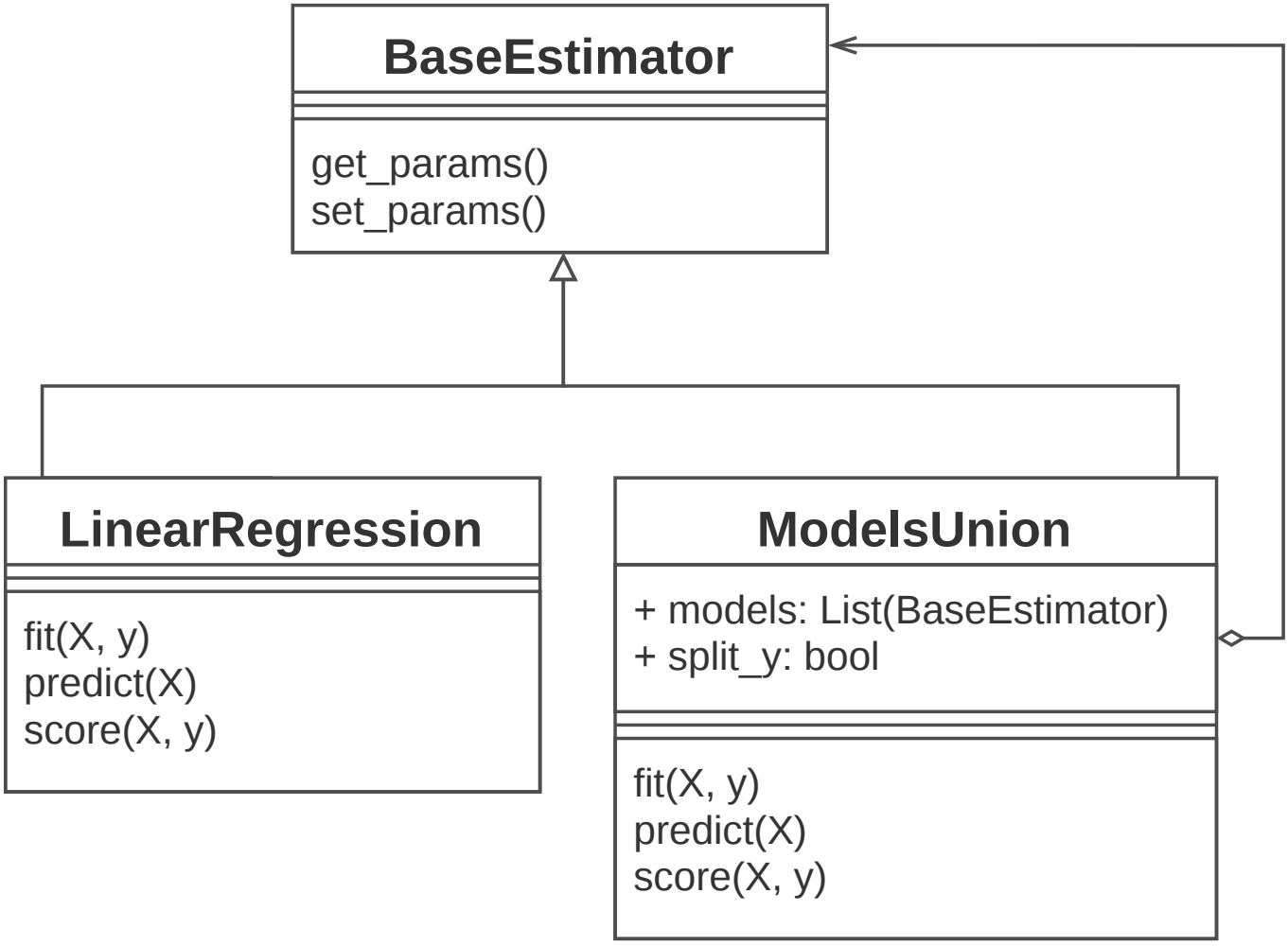


Figure 1: Class diagram of ModelsUnion

So, Model-union class puts the compositional model on one line with other surrogate models. It allows us to independently validate many surrogate models and combine them in a surrogate portfolio.

Optimization orchestrator

The *TutorModel* (TutotM) Class is the orchestrator of all optimization steps. TutorM is responsible for parallel surrogate build, their validation and combination. Also, TutorM provides surrogate models to the optimization algorithm. Due to the principle of separation of concerns, the surrogate model does no depend on the optimization technique. As a result, this extensive combination can provide additional flexibility and the ability to adapt to specific problems. An example of the workflow of TutorModel is presented in the Figure 2, As can we note, there are three surrogate models in the portfolio, from which pass validation only two.

Validation is the primary source of information for deciding on a surrogate model.

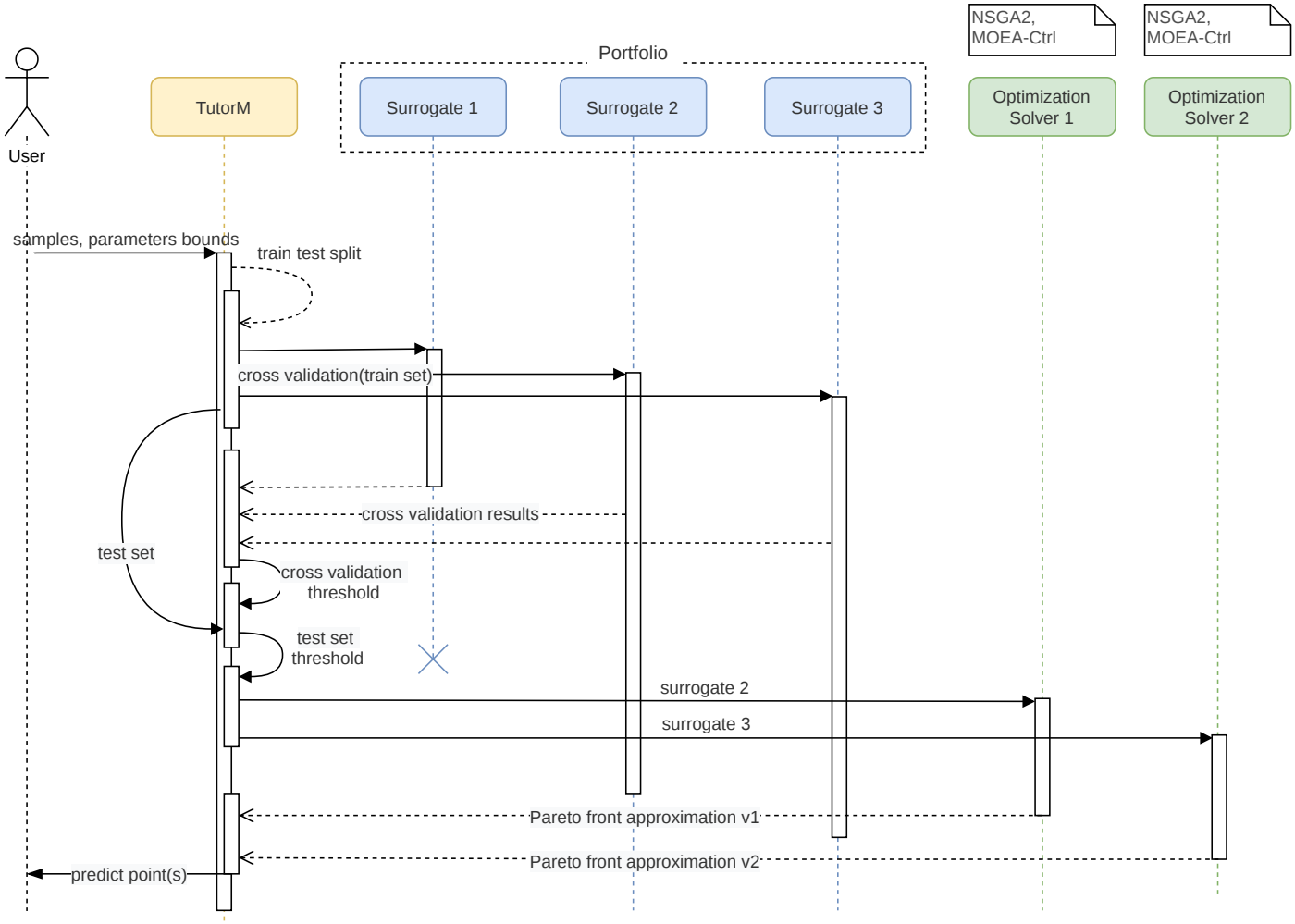


Figure 2: Optimization iteration with three surrogate models and two optimizers

Surrogates validation

To select a surrogate model, we need to check the accuracy of the assumption from unknown experiments(test set). As mentioned in previous Chapter, validation should be done in several stages to avoid overfitting (Figure 3). The validation steps are as follows: At the first stage, models are selected based on *cross validation* technique. In this stage define lower bound for overall accuracy. We notice that pass this threshold does not guarantee that surrogate is useful. 2) On the last stage, valid models from the previous stage are evaluated and selected for optimization.

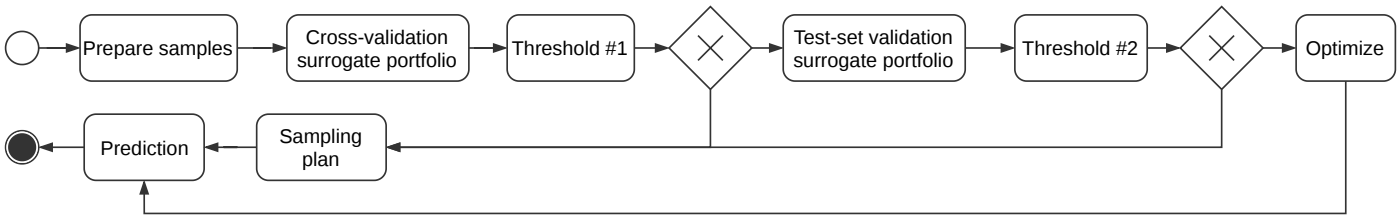


Figure 3: General optimization workflow for model-based optimization iteration with validation in two stages.

Optimization algorithm

The role of the optimization algorithm is to find a near-optimal solution based on surrogate approximation. While other methods also exist, we select as the main solver because it can be applied to a wide range of problems.

Optimization framework requires the definition of custom problems. The optimization problem is not built on top of surrogate models, but are used with the help of surrogate models. In the case of the

genetic algorithm, it produces the population of parameters, that could be treated as a Pareto front approximation. If several surrogates are valid than several Pareto front approximations obtain. There are two approaches to select the most informative solutions: 1) pick Pareto approximation from surrogate with the highest accuracy in non-dominated points. 2) Assume that all approximations are valid and all points could be selected. In this case, intersection predictions from samples have a higher probability of being selected.

Surrogate portfolio

Since the surrogate model can produce different results on a different problem, it is necessary to choose a model from the portfolio.

Surrogate models can be divided into two groups: a multi-output model for all objectives and compositional model with single-output models. All models pass validation equally, but after cross-validation single-objective models should combine with another one to provide multi-objective surrogate hypothesis. During this process, all objectives should be restored from valid surrogates.

Conclusion

We implemented a base class that can deal with a composite surrogate model and can combine arbitrary model to apply for a multi-criteria problem. The TutorM class is required to bring together implemented features such as surrogate portfolio, validation strategy and dynamic compositional model. Also, the requirements for the implementation of the proposed strategy have been identified. Mentioned requirements are intended to improve the further support and benefits of the developed method.

Evaluation

In this chapter, we present the results obtained from our approach. Additionally, we compare the developed approach with state-of-the-art strategies: evolutionary algorithms and static compound model-based optimization.

Experimental setup

We will begin by introducing a description of the selected optimization problems and applicable approaches for their analysis. Those problems include ZDT, DTLZ and WFG problems suits.

Optimization problems

Different optimization approaches need to applied to the numerous types of optimization problems to reduce the comparison bias in the obtained results (direct consequence of the No-free-Lunch theorem). To that mean, we select several comprehensive synthetic benchmark suites for comparison. They are all scalable in the parameter space and some are scalable in the objective space. The problems are designed so that a meaningful comparison can be obtained for optimization techniques. All cases are minimization problems.

According to [49], the following properties characterize the optimization problems:

- *Modality* is a property of the objective surface. Test problems are either unimodal, with one global optimum, or multimodal, with several local optima. Multimodal problems are more complicated than unimodal ones and bear more resemblance with real-world scenarios (Figure 1). Deceptive objective functions have a special kind of multimodality that has at least two optima — a true optimum and a deceptive optimum — but the majority of the search space favors the deceptive optimum [50].

- A *geometry* of the Pareto optimal front can be convex, linear, concave, mixed, degenerate, disconnected, or some combination of the former. It directly influences the algorithm's performance.

- A *bias* in landscape transformations impacts the search process by biasing the fitness landscape. Bias means that uniformly distributed parameters mapped onto a bias area in objective space. This type of problem can cause challenges if the bias region is far from the Pareto optimal front (Figure 1).

- *Many-to-one* fitness mapping means that different parameter vectors can produce the same objective vector. This property makes the search more difficult to optimize because it leads to situation when most solutions produce the same result.

- *Not separability* of the problem means that it can not be solved if consider it as a separate optimization problems for each objective.

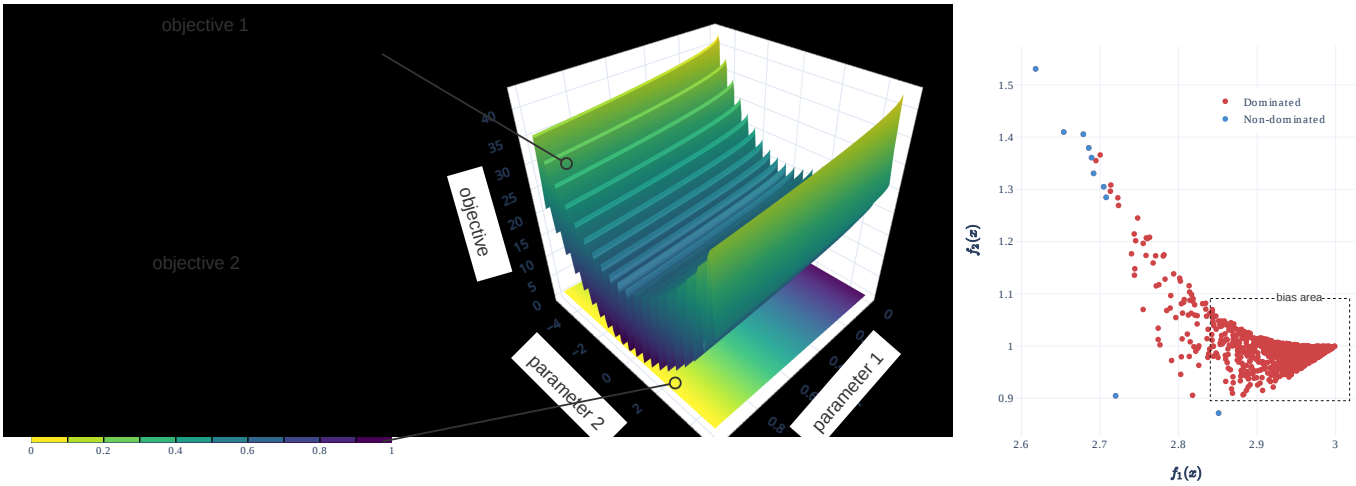


Figure 1: **(a)** Example of ZDT4 problem landscapes with multimodal (#1) and unimodal objectives (#2). **(b)** Example of bias landscape in WFG1 problem. Note how, in this example, the objective vectors are denser far away from Pareto optimal solutions.

ZDT

[3] is a test suite that consists of a set of two-objective problems and takes its name from its authors Zitzler, Deb and Thiele. In their paper the authors propose a set of 6 different scalable problems all originating from a well thought combination of functions allowing, by construction, to measure the distance of any point to the Pareto front. Each test function involves a particular feature that is known to cause difficulties in the evolutionary optimization process, mainly in converging to the Pareto-optimal front. For the evaluation of our combinational surrogate model we selected the following problems:

- **ZDT1** has a convex Pareto optimal front.

- **ZDT2** has a non-convex Pareto optimal front.

- **ZDT3** adds a discreteness feature to the front. Its Pareto optimal front consists of several noncontiguous convex parts. The introduction of a sine function in this objective function causes discontinuities in the Pareto optimal front, but not in the parameter space.

- **ZDT4** has 21 local Pareto-optimal fronts and therefore is highly multimodal. It is also called a *multifrontal* problem.

- **ZDT6** has a non-uniform search space: the Pareto optimal solutions are non-uniformly distributed along the global Pareto front and the density of solutions is the lowest near the Pareto optimal front and highest away from the front.

DTLZ

[51] is an extensive test suite that takes its name from its authors Deb, Thiele, Laumanns, and Zitzler. It was conceived for multi-objective problems with scalable fitness and objective dimensions. All problems in this test suite are box-constrained, continuous, n-dimensional, multi-objective problems.

- **DTLZ1** is one of the most difficult test problems in this test set. DTLZ1 has a flat landscape and the optimal Pareto front lies on a linear hyperplane.

- **DTLZ2** is an unimodal problem with a concave Pareto front.

- **DTLZ3** is a multimodal problem with a concave Pareto front. DTLZ3 is intended to make convergence on the optimal Pareto front more difficult than for DTLZ2.

- **DTLZ4** is an unimodal problem with a bias toward a dense region of solutions.

- **DTLZ5** has a bias for solutions close to a Pareto optimal curve. This problem may be easy for an algorithm to solve. Because of its simplicity, it is recommended to use it with a higher number of objectives.

- **DTLZ6** is a more challenging version of the DTLZ5 problem. It has a non-linear distance function $g()$, which makes it more difficult to find the Pareto optimal front.

- **DTLZ7** is an unimodal problem for its first objective and multimodal for the rest of its objectives. This problem has a disconnected Pareto optimal front, which decreases the likelihood that an Evolutionary algorithm(EA) finds all optimal regions.

WFG

[49] is a test suite designed to outperform the previously implemented test suites. Essential improvements have been achieved in a many problems. Also, non-separable, deceptive, and mixed-shape Pareto front problem are included. The WFG test suite was introduced by Simon Huband, Luigi Barone, Lyndon While, and Phil Hingston. WFG includes the following problems:

- **WFG1** is an unimodal problem with a convex and mixed Pareto front geometry.

- **WFG2** is a non-separable and unimodal problem with a convex and disconnected Pareto front geometry.

- **WFG3** is a non-separable, unimodal problem for all but its last objective. The last objective is multimodal.

- **WFG4** is a multimodal problem with a concave Pareto front geometry. The multimodality of this problem has a landscape with large hills that makes optimization more complicated.

- **WFG5** is a separable problem with a deceptive landscape and a concave Pareto front geometry.

- **WFG6** is a non-separable and unimodal problem. Its Pareto front geometry is concave.

- **WFG7** is a separable, unimodal problem with a concave Pareto front geometry. WFG7 and WFG1 are the only problems that are both separable and unimodal.

- **WFG8** is a non-separable, unimodal problem with a concave Pareto front geometry.

- **WFG9** is a multimodal, deceptive, and non-separable problem with a concave Pareto optimal geometry. Similar to WFG6, the non-separability of this problem makes it more complicated than WFG2 and WFG3.

Base on the properties, we decide that ZDT4, ZDT6, DTLZ4, WFG1, and WFG4 can represent a broader spectre of possible problems (Table 1). Also, solutions to these problems provide meaningful insight into how our optimization strategy performs. Therefore, for brevity and more comprehensible discussion, we present full evaluation only of these problems. However, we have condensed results for all mentioned problems in appendix.

Table 1: Selected multi-objective test problems.

Problem	Objective	Modality	Geometry	Bias	Many-to-one mappings
ZDT4	bi-objective	unimodal, multimodal	convex	-	-
ZDT6	bi-objective	multimodal	concave	+	+
DTLZ4	multi-objective	unimodal	concave	+	+
WFG1	multi-objective	unimodal	convex, mixed	polynomial, flat	+
WFG4	multi-objective	multimodal	concave	-	+

Optimization search

In this thesis, we do not perform explicit parameter tuning for optimization algorithms. While various optimization algorithms could have been used, we selected MOEAs as default optimization techniques for surrogate models. The advantage of EAs are that they can be easily modified and can operate on a set of solutions candidates that are well-fitted to approximate the Pareto front. Also, EAs can estimate highly complex problems in various use-cases. In this thesis, we used two types of EA:

1. The popular evolutionary multi-objective algorithm *NSGA2* [52]. We chose this algorithm due to its popularity in Multi-objective optimization (MOO). In all cases, default parameters for the algorithm were used (population size = 100, crossover probability=0.95, distribution index for crossover=10.0, mutation probability=0.01, distribution index for mutation=50.0)[27].

2. As an alternative MOEA algorithm for optimization, we define *MOEA-Ctrl* that combines MOEA/D [53] and NSGA2 algorithms. The characteristic of such an optimization process based on a common solutions population for both algorithms. Our intuition behind this choice is the following: NSGA2 gives stable results with well-distributed points on the Pareto front while MOEA/D has great exploration quality with low generation count. The combination of this algorithm should gain a better trade-off in exploration and exploitation in contrast to individual algorithms' application.

Surrogate portfolio

Based on our awareness, we selected the most popular models for a default surrogate portfolio.

1. **Gaussian Process Regressor**[2] it is a multi-objective surrogate model that commonly used in the Bayesian optimization. For this type of model, the initialization should be specified by passing a kernel object, the hyperparameters of which are optimized during extrapolations of the samples. The kernel for benchmarks is selected from the GPML[18]. Even though this kernel is from another domain, it

does give good extrapolation quality for the regression model. Unfortunately, the build time is significant and grows with samples size and dimensionality.

2. **Support Vector Regression (SVR)**[3] single-objective model with Radial-basis function(RBF) kernel. Surrogate based on RBF and SVR are preferred choice for high dimensional problems [41].

3. **Multi-layer Perceptron regressor (MLPRegressor)**[4] A neural network is a popular and influential approach to approximate the functions landscape [16].

4. **Gradient Boosting Regressor**[5] is a single-objective model that uses an ensemble decision tree regressors to produce a single model.

As a result, for bi-objective problems, there are no more than ten possible surrogate hypotheses (including multi-objective Gaussian Process Regressor). For a benchmark purpose, at each optimization round the surrogate portfolio does not change.

Benchmark baseline

We compare our approach (TutorM) with Hypermapper 2.0[23] that was considered in the related work. Hypermapper focuses on multi-objective parameter tuning with various types of parameters. It uses several randomized forest models, one for each objective. The general idea is to scalarize several surrogate models to single-objective criteria and to optimize them as a single-objective problem. In addition, a Bayesian model is used to assist the search for solutions. Hypermapper has successfully been used in autotuning computer vision applications and database optimization. Since the sample size is not specified, we chose to use the default population size for MOEA (100 points).

In addition to Hypermapper, NSGA2 was chosen as it is one of the most well-known evolutionary algorithms [13]. It is therefore a suitable reference point to which to compare other approaches. As benchmarks, we evaluate two versions of the algorithm that are nearly identical but have a different budget for evaluation:

- Small evaluation budget (1000 functions evaluations) and used as a competing algorithm

- Large evaluation budget (10000 and 50000 functions evaluations) and used as a baseline. NSGA2 with 10000 functions evaluations budget used as a baseline for figures with runtime performance, whereas 50000 budget used for final results.

Benchmark 1: Portfolio with compositional surrogates. Dynamic sampling plan

For this first evaluation step, our approach (TutorM) was compared to related approaches (Hypermapper and NSGA2) while solving all three sets of problems described above (ZDT, DTLZ, WFG) for 2 objectives and 2 or 3 parameters. The TutorM includes all features such as dynamic compositional models, surrogate portfolio and validation.

The solution quality was evaluated using the following metrics: hypervolume, p-distance, spacing, and number of available non-dominant solutions (ndf size). The results we present are the average values obtained after five repetitions. It should also be noted that baseline NSGA2 10k is a static value that is obtained after 10000 function evaluations.

One case studies: ZDT6

We start by comparing the runtime performance of the approaches. Let us consider runtime optimization for the ZDT6 problem. In Figure 2, optimization progress and average distance to the Pareto front is shown.

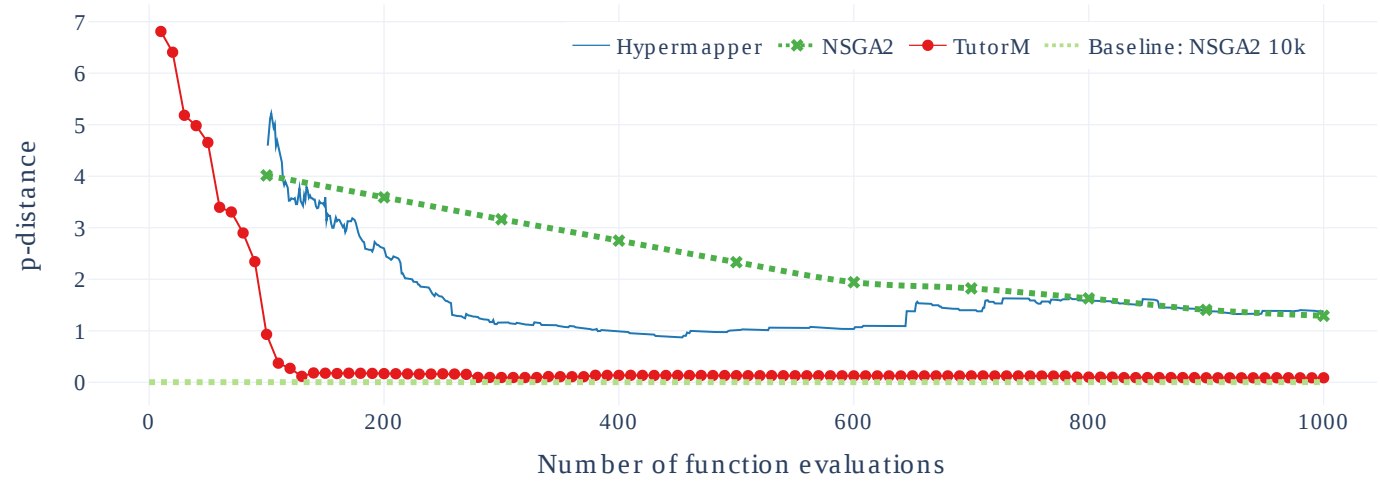


Figure 2: Results of 5 repetitions for ZDT6: Average distance of non-dominant solutions to the real Pareto front.

It is evident that TutorM considerably outperforms NSGA2 and Hypermapper 2.0 right from the start of the optimization process. Our algorithm quickly reaches optimal and stable results after 300 function evaluations. As can be seen from another approach, Hypermapper began to improve values confidently, but then deteriorated and matched with NSGA2. The presented p-distance is measured from non-dominated solutions (ndf size) that can be found in Figure 3.

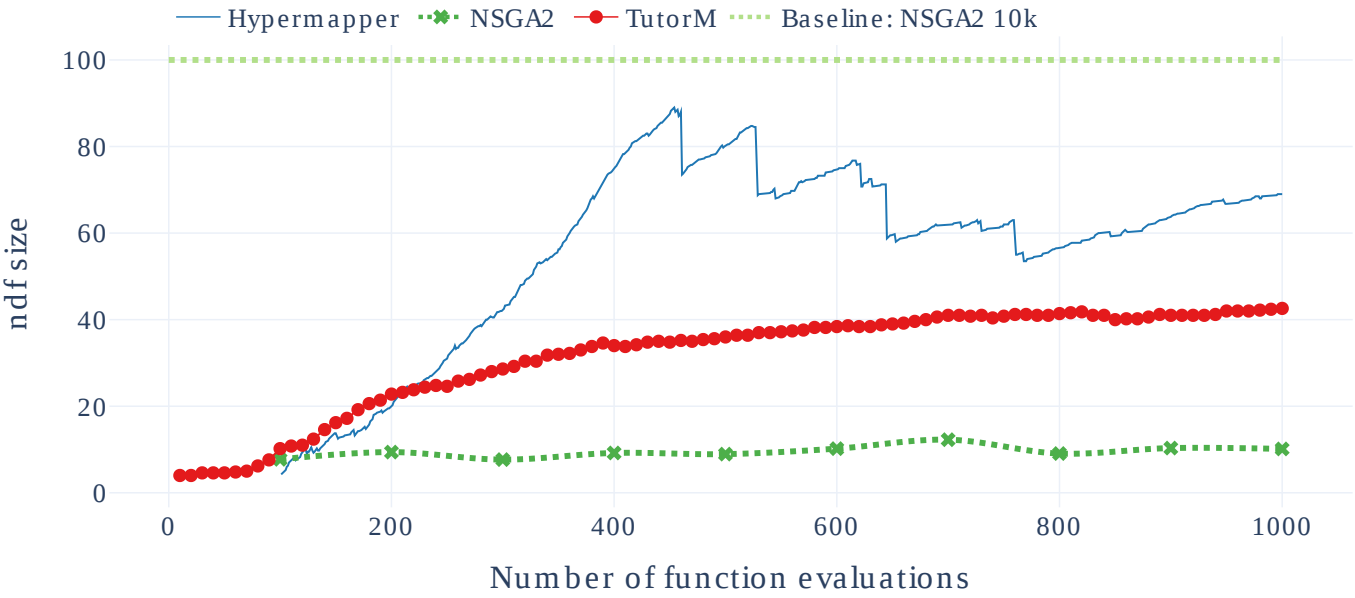


Figure 3: Results of 5 repetitions for ZDT6: Size of non-dominant solutions across the entire set of the measured solutions.

The count of solutions from TutorM grows evenly, reflecting the stability of the search and the ascent to the real Pareto front. On the contrary, Hypermapper has a serrated, unstable track that corresponds to solutions that are stuck in local Pareto fronts. Repeated drops occur upon the discovery of a new point in the other Pareto optimal fronts.

Figure 4 shows that during the first six optimization iterations, a sampling plan was used until a valid model appeared. This suggests that, for a given problem with this surrogate portfolio, the 60 samples obtained from the sampling plan are enough to begin the optimization process. As can be noted, for this

problem and with this portfolio, the most suitable compositional surrogate is a *Gradient Boosting Regressor*.

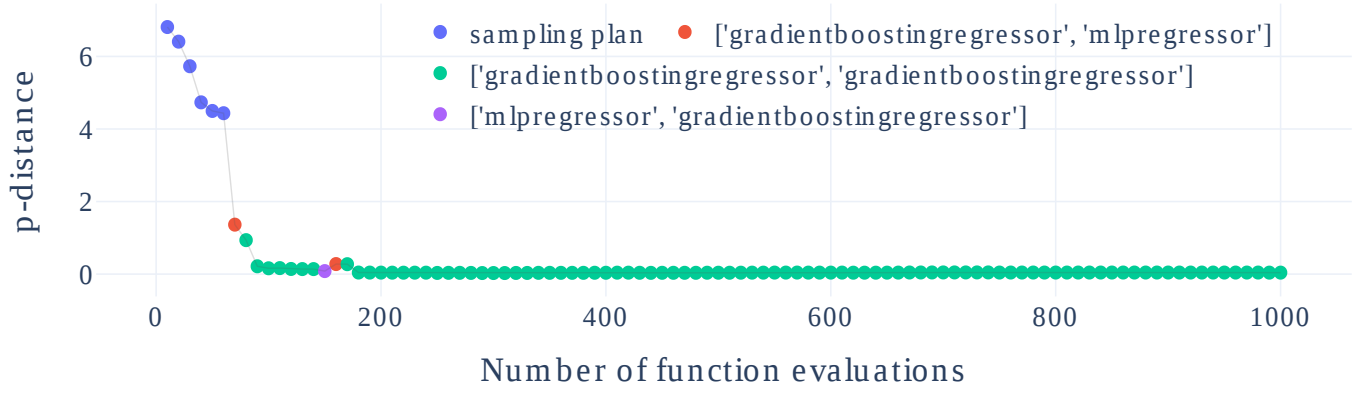


Figure 4: ZDT6: Best models from the portfolio.

The final solution consists of non-dominant configurations that give an idea of the Pareto front. In terms of overall Pareto front approximations (Figure 5)), only TutorM solutions reach the baseline (NSGA2 10k) while other solutions are close and distributed (Hypermapper) or far away and clustered (NSGA2).

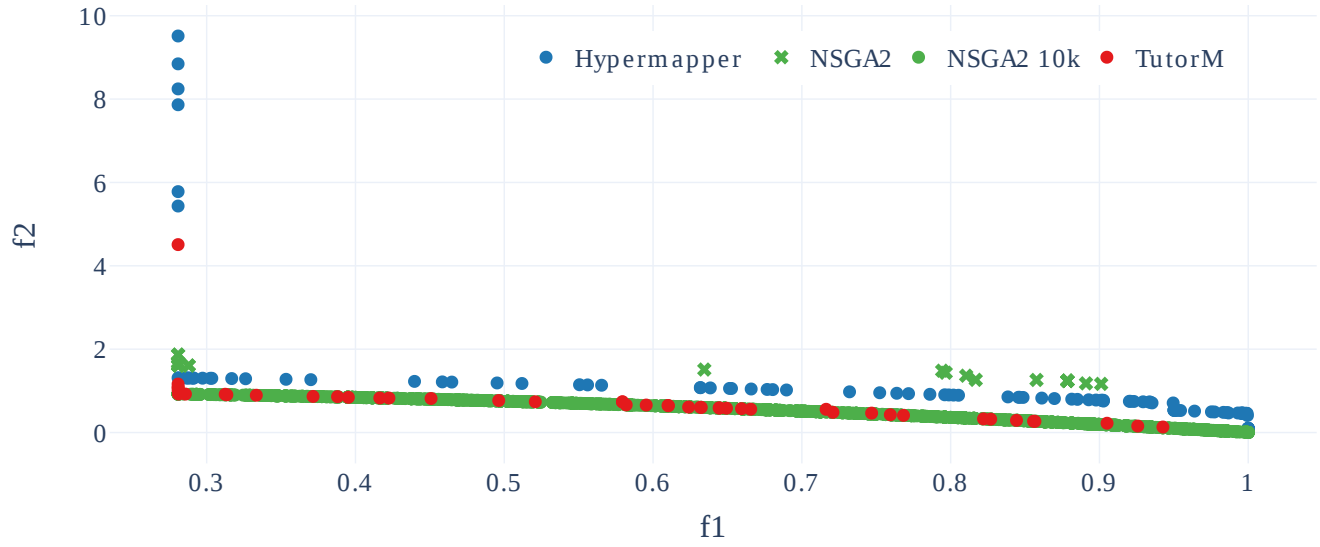
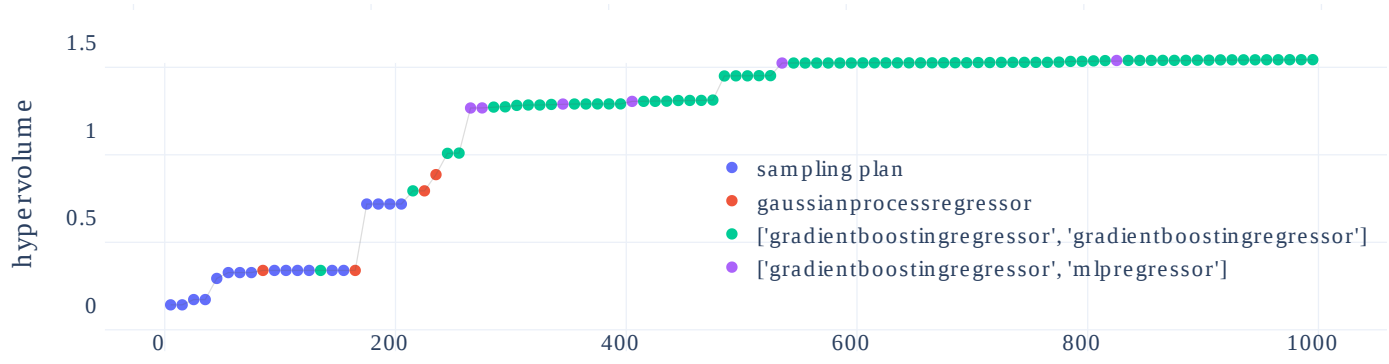


Figure 5: The final assumption of the Pareto optimal solutions (1000 function evaluations) for ZDT6.}

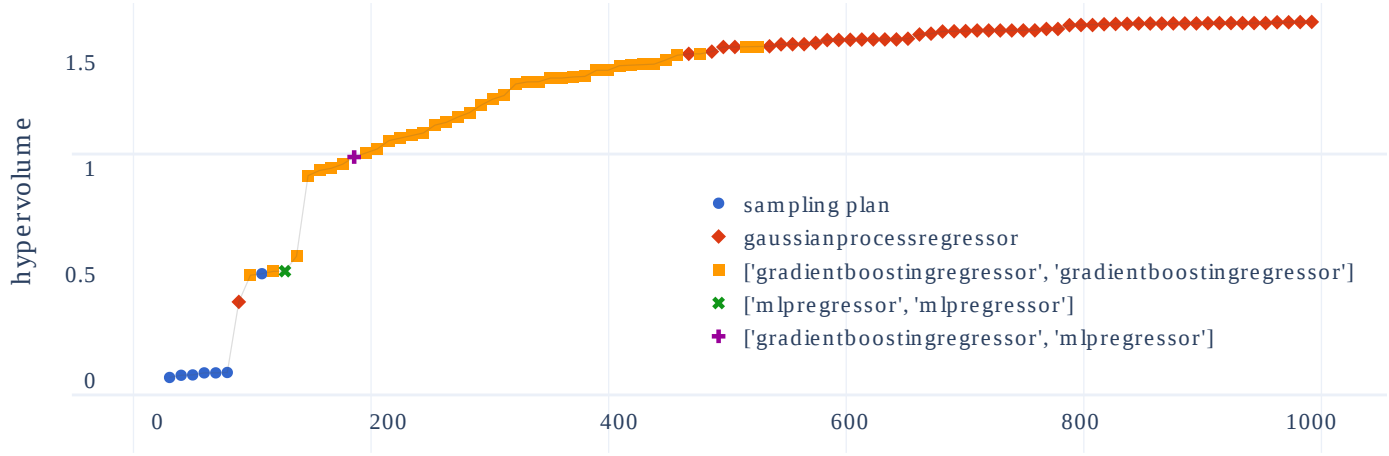
Case studies: WFG1, WFG4, DTLZ4

Below, we compare how the optimization process varied across several problems. The key feature of the method we developed is the dynamic sampling plan, which depends on the quality of available surrogates. As mentioned before, in Figure 4, when a static number of random samples is estimated, it is possible to make optimal decisions much earlier.

This approach is used in TutorM for all optimization problems. By interpreting the end of the sampling plan and the availability of valid models, we can estimate the cross-grain complexity of the unknown problem. Figure 6 shows a difference in the adaptation of initial samples to problems (DTLZ4, WFG1) and a corresponding improvement in hypervolume.



(a) DTLZ4: sampling plan to the 210 configuration



(b) WFG1: sampling plan to the 60 configuration

Figure 6: Optimization process with dynamic sampling plan and surrogate portfolio. Plots show in which steps a sampling plan was used or which model gives the best accuracy on the test set. The order of the models in the composite surrogates corresponds to the order of the optimization objectives that this model extrapolates.

In the case of WFG1, a valid model was quickly obtained and reduced the initial sampling. This may indicate a convenient and unimodal optimization landscape. On the contrary use-case of DTLZ4, the sampling plan lasted longer and alternated with valid models. This may reflect the complexity of the problem, such as the multimodality or bias of the landscape. It should also be noted that in each case we considered, the best surrogate model was different and might change during optimization. Thus, for the case of DTLZ4, a clear advantage was observed in the choice of composite surrogacy with *Gradient Boosting Regressor*, whereas for WFG1, the multi-objective *Gaussian Process Regressor* was the preferred choice.

In the next comparison, we look at WFG1 and WFG4. Figure 7 8 illustrates how the evaluation budget can be spent to find Pareto optimal solutions. Let us look at the WFG1. It can be seen that TutorM slowly increases the number of solutions during optimization (7). Furthermore, the final result even exceeds the solutions given by the NSGA2 after 10k function evaluations (8). Turning now to Hypermapper, the non-dominated plot is highly serrated, which indicates that the approach falls within the local optima. Additional information is revealed in the final Figure 8, which shows that most final Hypermapper solutions are strongly clustered, reflecting a waste of resources.

For the WFG4 use-case, all approaches produce near-optimal solutions, but only TutorM provides such an extensive set of near-optimal solutions (non-dominated 400 solutions from 1000 function evaluations). This property of TutorM means that the optimization process can stop earlier and save on the evaluation budget.

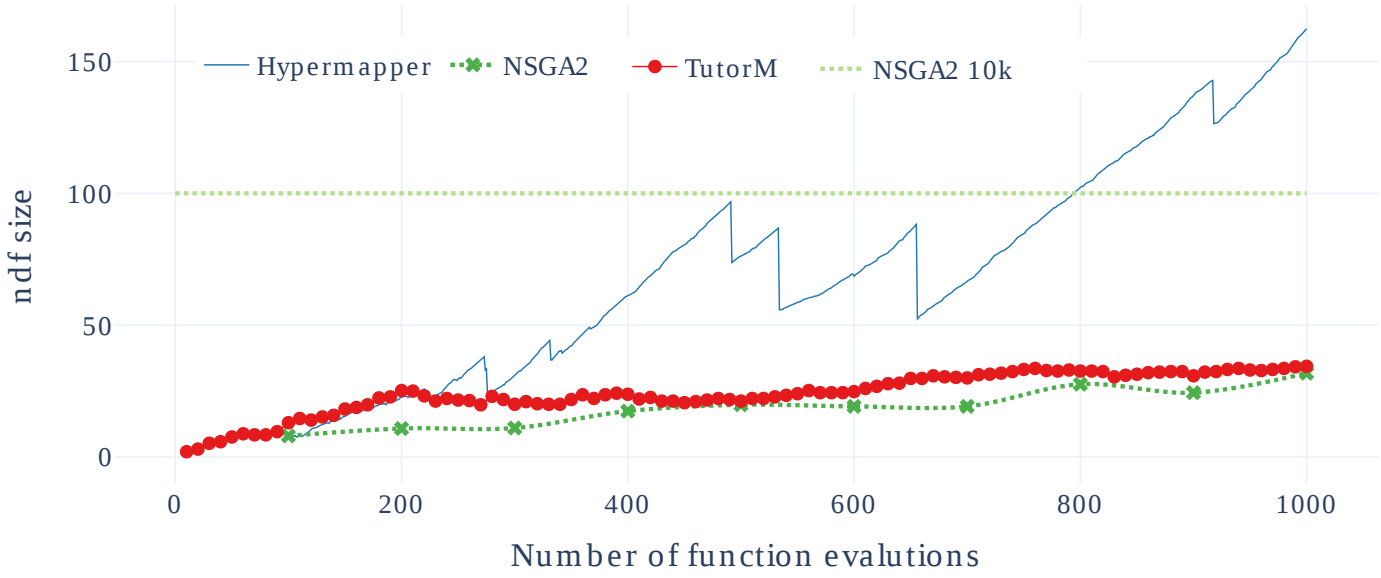


Figure 7: Results of 5 repetitions for WFG1: Size of a non-dominated subset of evaluated examples

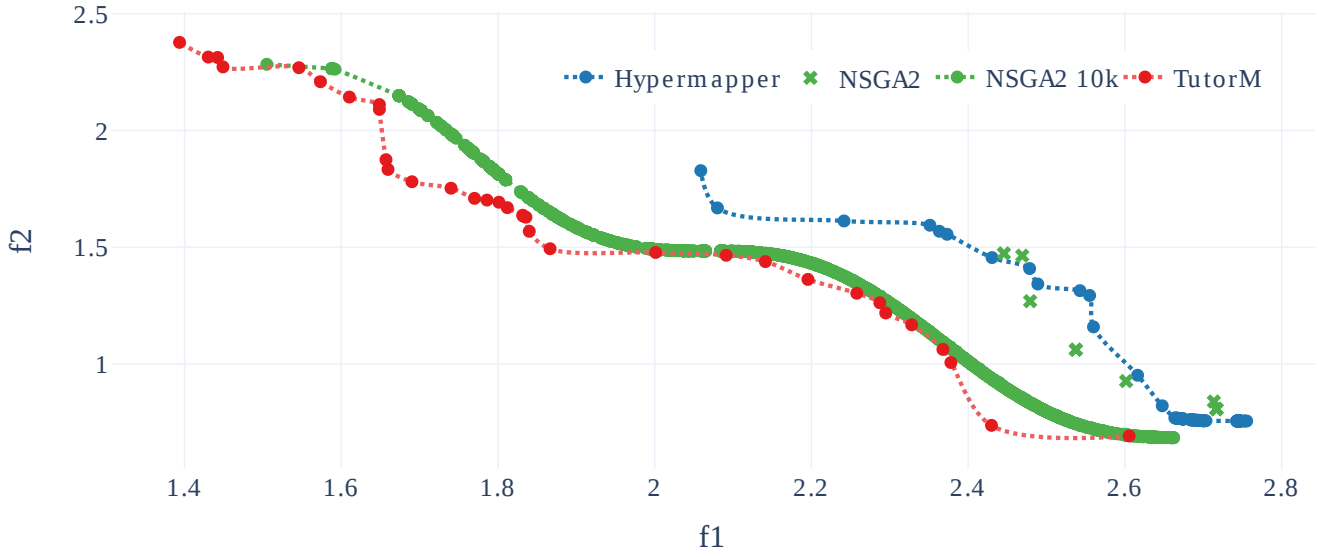


Figure 8: WFG1: Pareto front approximation

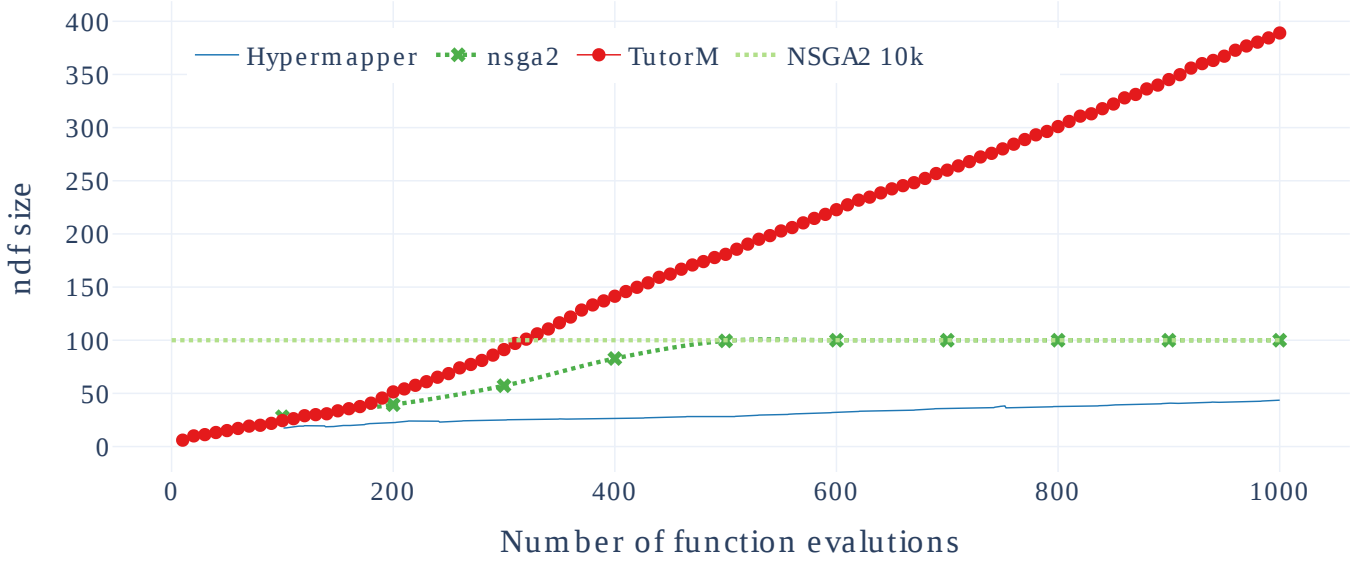


Figure 9: Results of 5 repetitions for WFG4: Size of a non-dominated subset of evaluated examples

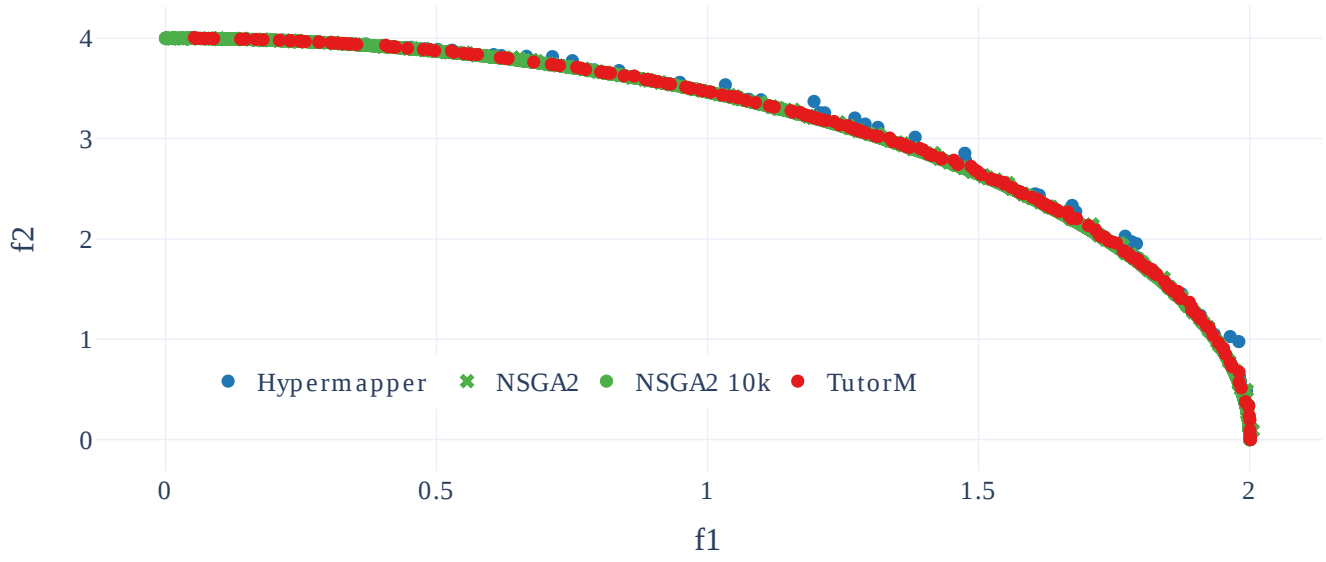


Figure 10: WFG4: Pareto front approximation

Results

For benchmark 1, we analyze 21 problems from three test sets. They have 2 or 3 parameters and 2 objectives. We repeat the experiments 5 times and average them. We consider the following metrics:

- **Hypervolume** This metric is calculated for each comparison because the hypervolume requires a single reference point for a group of competitors. Hypervolume metric is given as a percentage where 100% corresponds to the maximum volume in the competition and 0% corresponds to the hypervolume of a random set of solutions.

- **p-distance** The primary metric for evaluation that corresponds to average distance to real Pareto front. Unfortunately, it is not available for WFG problems.

- **Non-dominated font (ndf) size** The ratio of the number of final non-dominated decisions to the number of spent function evaluations.

- **Spacing** The inverse spacing metric is calculated, where 1 corresponds to the most cohesive decisions among competitors.

In following table (Table 2), we present a subgroup of problems that have varying complexity. The full list of results is provided in the appendix.

Table 2: Comparison of results after 1000 function evaluations.

		ZDT4	ZDT6	DTLZ4	WFG1	WFG4
TutorM	**Hypervolume** ↑	99,80%	99,43%	99,83%	95,75%	99,28%
	p-distance ↓	0,01	0,09	0,001	-	-
	ndf-size ↑	50%	4,26%	0,2%	3,44%	38,9%
	space-metric ↑	0,78	0,17	0,666	0,51	1
NSGA2	**Hypervolume** ↑	83,43%	83,84%	87,81%	30,52%	83,95%
	p-distance ↓	0,04	1,29	0,002	-	-

		ZDT4	ZDT6	DTLZ4	WFG1	WFG4
	ndf-size ↑	8,77%	1,01%	9,600%	3,18%	10%
	space-metric ↑	0,19	0,04	0,323	0,28	0,58
Hypermaper 2.0	**Hypervolume** ↑	97,32%	82,86%	64,579%	44,12%	84,39%
	p-distance ↓	0,9	1,12	0,059	-	-
	ndf-size ↑	5,42%	6,25%	1,17%	10,24%	3,26%
	space-metrics ↑	0,11	0,08	0,029	0,31	0,06
NSGA2 50k (Baseline)	**Hypervolume** ↑	100%	100%	100%	100%	100%
	p-distance ↓	2,04e-05	0,0003	8,81e-06	-	-
	ndf-size ↑	0,72%	0,72%	0,360%	0,72%	0,72%
	space-metric ↑	1	1	1,000	1	0,6

To summarize, it follows from our results that our strategy generally gives optimal or better results than the baseline on the majority of investigated problems.

We assume that our positive results were due to the new features we implemented, such as a surrogate model portfolio and adaptive sampling plan. These features have yielded significant results on almost all problems. However, we did not apply inner parameter tuning: in all experiments, TutorM was used with default parameters.

Benchmark 2: Inner parameters

For the second benchmark, we investigated whether it is possible to further improve the performance of TutorM by tuning its parameters. We examine the effect of internal parameters on the performance and quality of optimization. As was mentioned in the previous section, it was applied with a default setting.

TutorM parameters

Besides the standard model-based parameters, it is necessary to investigate the impact of additional TutorM parameters such as validation thresholds, test-set and prediction size. This research is needed to select the configuration that can improve results of the existing system. Unfortunately, there is insufficient information available about how to configure model-based parameter optimization [39,46]. Filling this gap in knowledge will be useful not only for the availability of TutorM but also for general tuning of model-based optimization. Due to limited time, we consider only the ZDT4 and ZDT6 problems using the surrogate portfolio from the first benchmark, but without the *Gaussian regression model*. This model takes a long time to train and the full factorial design did not fit within our time frame. The following parameters are exposed in the developed TutorM class:

- **Initial dataset** [`0`, 100, 500, 750]. It is the initial number of points obtained from sampling plan. At the same time, the total budget for measurements remains unchanged and equals 1000. The default value is `0`.

- **Surrogate validation.** Criteria and parameters for evaluating the usefulness of the surrogate model.

  - **Train/test split** [`75/25`, 90/10] is a splitting proportion in which the samples available for training and testing are divided. Train and test sets are crucial to ensure that the surrogate model is able to generalize well to new data. The default value is `75/25`.

- **Cross-validation threshold**[0.2, `0.65`, 0.9] is a minimum accuracy threshold for any round in cross-validation(CV). CV is used to select valid surrogate models and avoid overfitting. The default values is `0.65`.

- **Test threshold** [0, `0.6`, 0.9] is a minimum accuracy threshold for the test set. The accuracy obtained from the test set and is used to verify the validity of models based on how they extrapolate unknown data. The default value is 0.6.

- **Optimization search algorithm** [NSGA2, `MOEA-Ctrl`] optimization algorithm for multi-objective solutions. The default value is `MOEA-Ctrl`.

- **Solution combinations** [Non-dominated front score (ndf score), `Stacking`] approach for choosing a set of solutions from a valid surrogate model. Since several models can be valid and each one provides its own set of decisions, we have a range of options. *Non-dominated front score (ndf score)* prefers the surrogate model with the highest precision for non-dominant solutions, whereas the *stack* integrates all available surrogate solutions into one set of solutions. The default value is `Stacking`.

- **Prediction count** [`10`, 100] number of random solutions for the real evaluation that are selected from the set of solutions. The default value is `10`.

As a result of the full factorial design, 576 possible combinations were obtained. Each combination was repeated five times and averaged. Conclusions were made based on the 40 best and worst combinations.

First, we will consider the ZDT6 problem. Inspection of Figures 11 12 indicates that the *solution combination* made the most significant impact on the result.

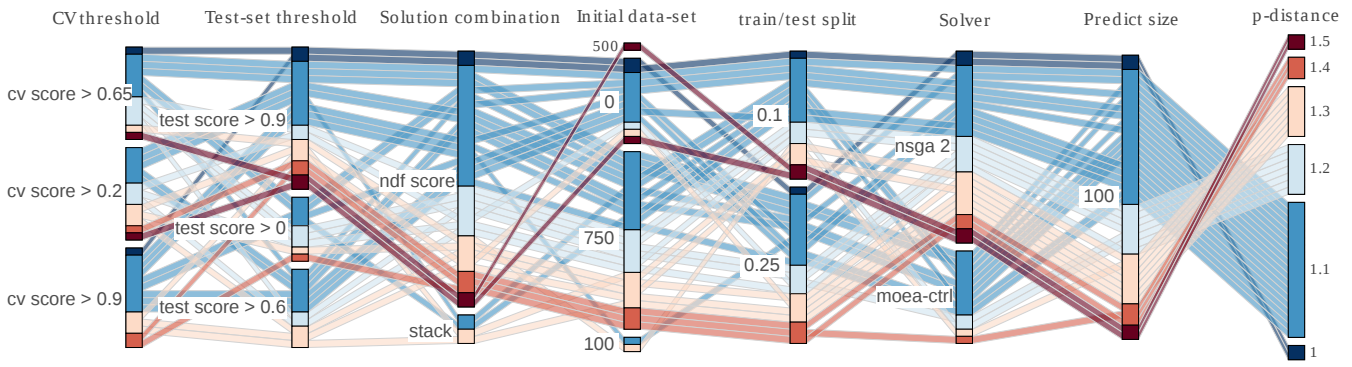


Figure 11: ZDT6: Average result of the worst configurations

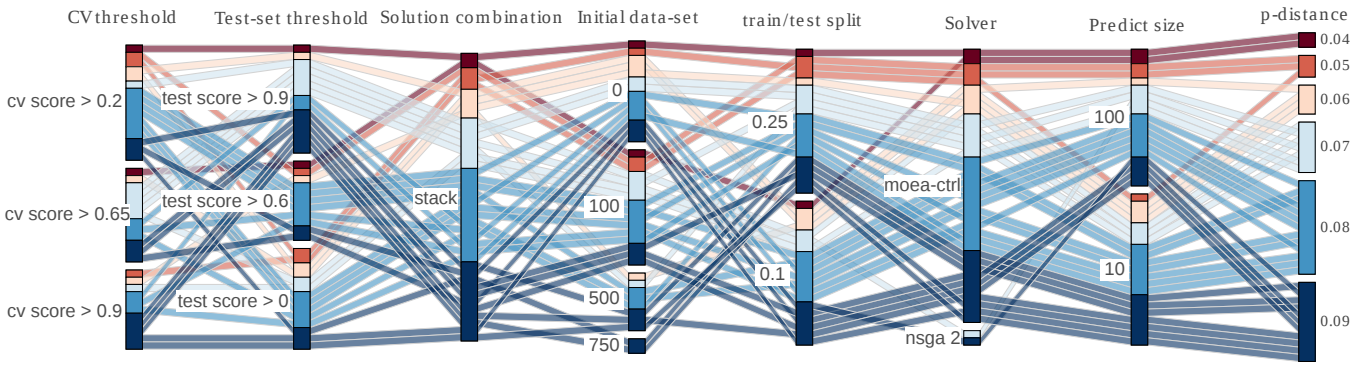


Figure 12: ZDT6: Average result of the best configurations

There is a definite advantage in combining solutions into a stack. The second most important parameter is the *Optimization search algorithm*(Solver). The best configurations prefer to pick a combination of Genetic Algorithms (MOEA-Ctrl) for optimization.

Let us look at the *solution combination* and the *Optimization search algorithm* options in more detail (Figure 13).

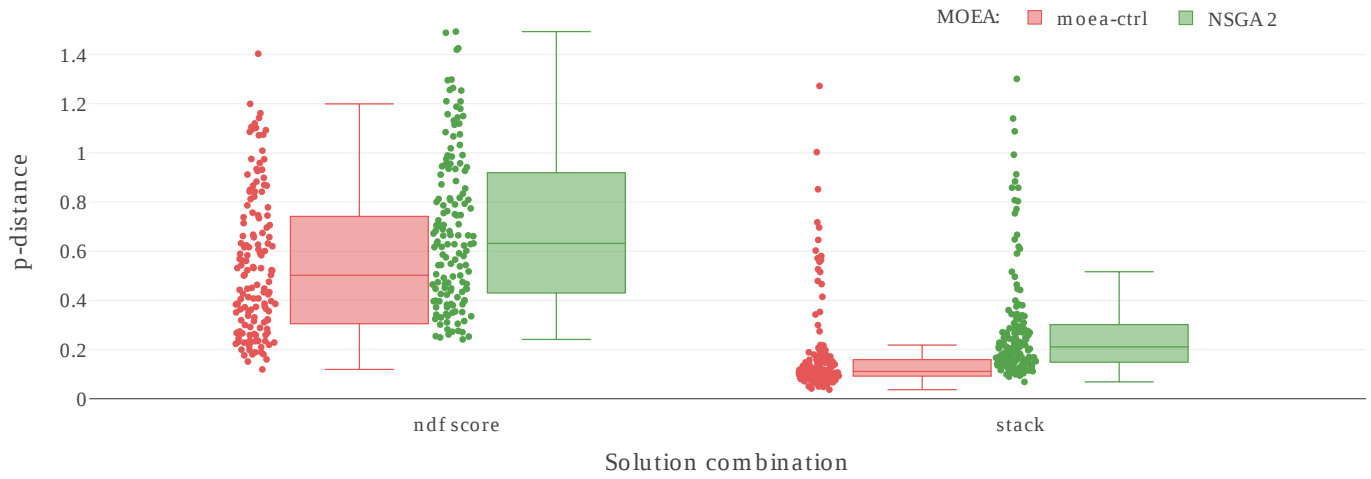


Figure 13: Correlation between the most influenceable parameters for the ZDT6: solution combination strategy and optimization algorithm

The impact of changing the algorithm is highly dependent on the solution combination strategy. Improvement in results for MOEA-Ctrl is more significant when the results are combined into a stack. This advantage can be explained by the fact that the stack reduces the bias of surrogate models while the combination of genetic algorithms decreases prediction variance. Now we will look at the ZDT4 problem (Figure 14 15).

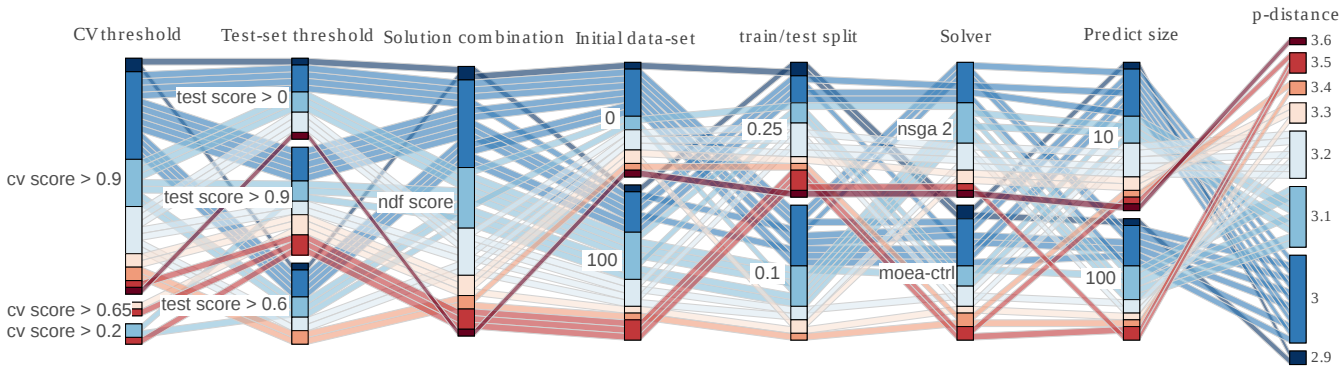


Figure 14: ZDT4: Average results of the worst configurations

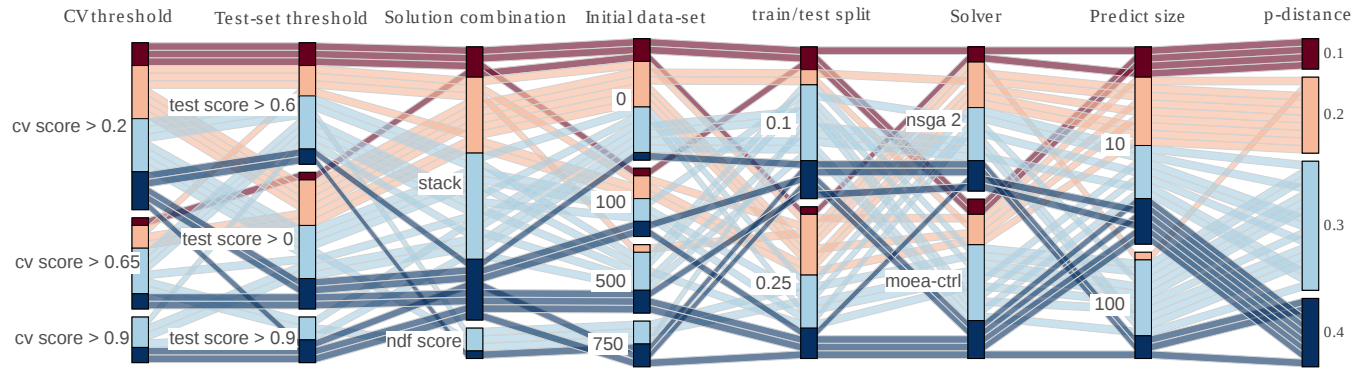


Figure 15: ZDT4: Average results of the best configurations

The results are similar to those obtained with the ZDT6 problem: the solutions stack take part almost in all of the best configurations. However, for this problem, there is no clear dominance of a single algorithm. Yet, the validation thresholds have an impact on results (Figure 16).

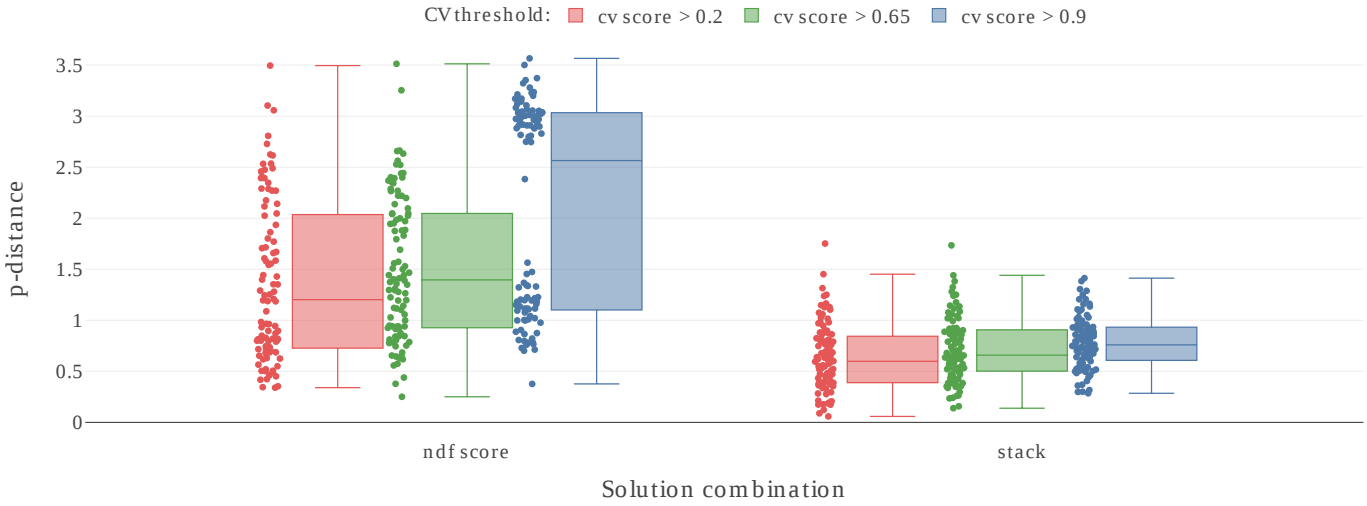


Figure 16: An impact of the cross-validation threshold for the ZDT4.

A significant difference is seen for the cross-validation threshold in the case of *ndf score solution combination* set (Figure 14). It should be noted that the stack makes the validation threshold impact less significant, as evident from Figure 16. This influence is related to this technique's ability to reduce the bias of solutions.

Another interesting conclusion can be made from the *initial sample size*. The worst and the best configurations are most affected by tha absence of a sampling plan. The reason for this is that the small number of samples may lead to a surrogate model fallacy in extrapolation the search space while, at the same time, the small number of samples provide more opportunities for optimization search.

Sampling plan size

The purpose of this experiment is to review the dependencies between the optimization results and the sampling plan size. The Hypermapper was selected as a foundation for analysis because it has a static implementation of the optimization algorithm with the surrogate model.

The results are shown in the following Figure 17.

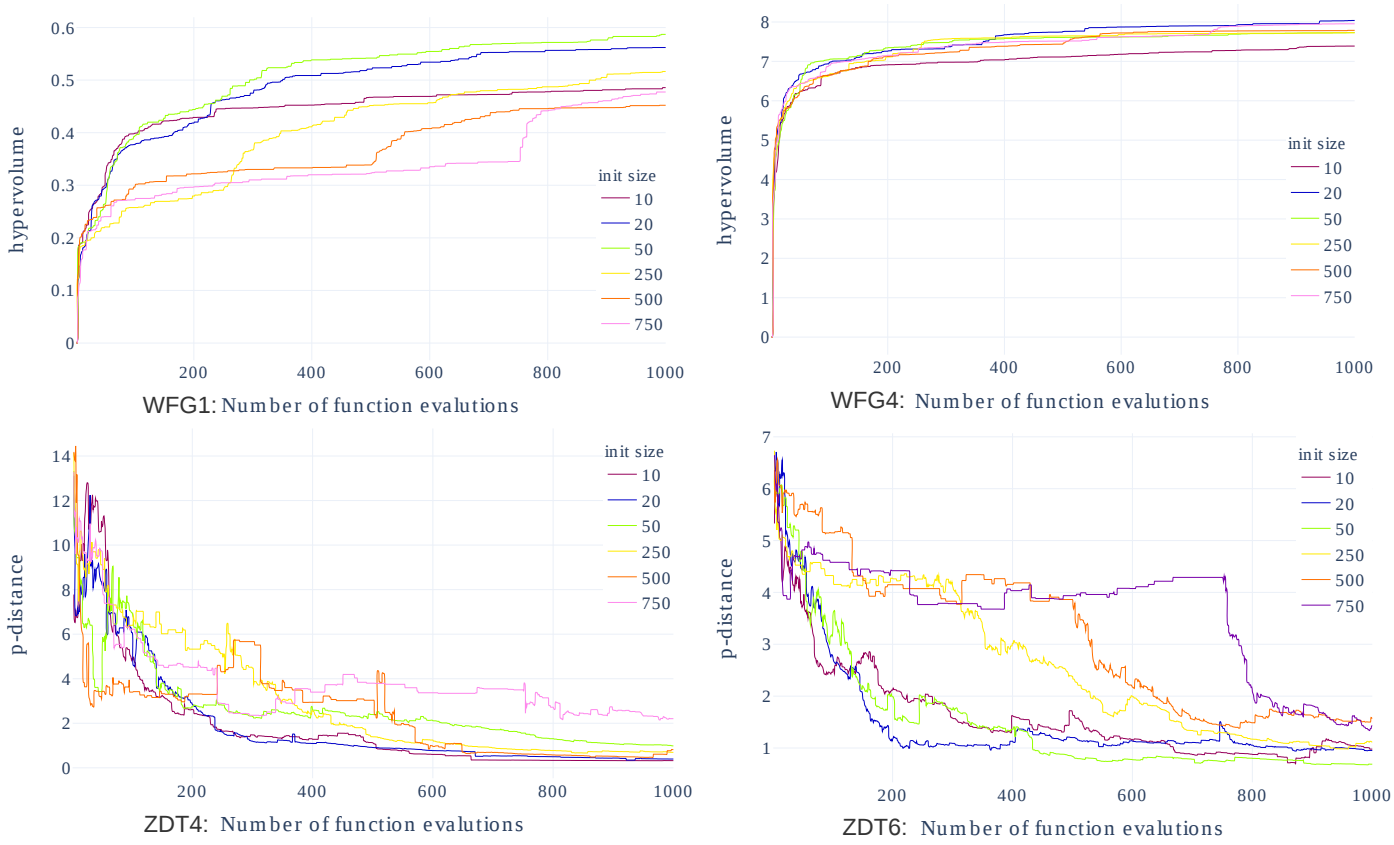


Figure 17: Influence of the initial sample plan on optimization process with Hypermapper 2.0

For WFG problems, the criterion is hypervolume and for ZDT problems it is p-distance. Of all the results, the initial sampling plan has the smallest effect on the WFG4. Since this problem is unimodal, the model requires fewer samples for extrapolation. Other problems have a more complicated multimodal landscape that is shown by unstable results.

Results

We investigated the parameters of TutorM and determined which ones produce the best results. Also was noticed that *Solution combinations* and *Optimization search algorithm* had the most significant impact on solution quality. The *stack* of solutions with MOEA-Ctrl is the best combination of parameters to use as a default for TutorM. The other parameters tested have a much smaller effect.

Benchmark 3: Scalability of surrogate models

Not only the type of the problem landscape but also its dimensions are essential factors for picking a surrogate model. The advantage of a surrogate model can be lost if the number of parameters or criteria is changed. The goal of this experiment is to find out the scalability of surrogate models.

The following surrogates were selected for evaluation:

- *Gaussian Process Regressor* with kernel design from GPML[18]. Gaussian process models are well-known and are commonly used in Bayesian optimization for a wide variety of problems [20,54].

- *MLPRegressor* is a neural network implementation from the *sklearn* framework. Neural networks can automatically discover useful representations in high-dimensional data by learning multiple layers [55]. Because this model simultaneously extrapolates all objectives, we chose an architecture that consists of 5 layers and 100 neurons per layer.

- The surrogate portfolio includes Gradient Boosting Regressor, MLPRegressor, and *SVR (RBF kernel)*, as mentioned in Benchmark 2.

The DTLZ2 problem was selected to evaluate the scalability of the surrogate models. It is an unimodal problem with multiple global optima and concave geometry of the Pareto front. During experimentation with DTLZ2, the number of optimization criteria changed with a constant number of parameters. Figure 18 shows the three selected surrogate strategies with the average distance to the Pareto front (first row) and time spent per optimization iteration (bottom row). For all cases, the experiment was repeated five times.

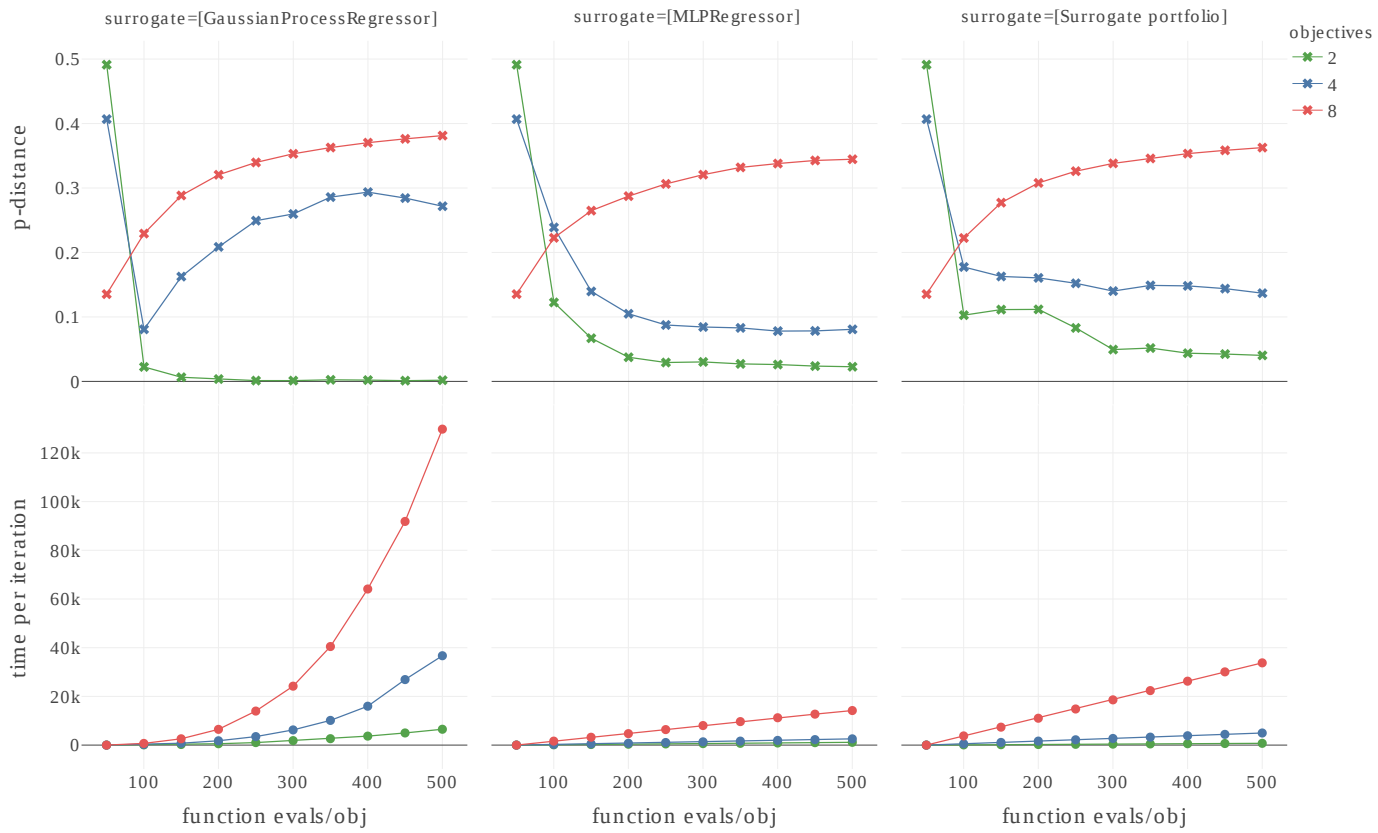


Figure 18: Scaling example for the three variants of the surrogate on the DTLZ2 with the 9-dimensional parameter space

As illustrated by Figure 18, *Gaussian Process Regressor* model provides significantly better results relative to other approaches, but only for the bi-objective problem. Increasing the number of objectives to four leads to only the *MLPRegressor* and surrogate portfolio converging on an optimal solution. Further increasing the number of objectives makes the search space too complicated, and all approaches fail to find solutions.

Results

The ability of models to estimate the search space depends on their hyperparameters. As an example: *Gaussian Process Regressor* are highly dependent on the kernel while *MLPRegressor* depends on a number of layers. In turn, for the surrogate portfolio, the parameters determine how to *build and select* surrogate models. In the portfolio, a single model with varying parameters is evaluated as a set of separate entities. Thus, the scalability required to solve multi-dimensional problems can be generated by surrogate portfolios.

Discussion of results

The purpose of this thesis is to investigate the use of a cross-grained compositional model for solving multi-objective problems. In this evaluation section, we provided an extensive comparative analysis of the performance of our and other techniques on a wide range of problems. The analysis also included an exhaustive study of the possible parameters and the impact of the sampling plan on results. The possibility of scaling surrogate models by increasing the number of objectives was also tested.

We draw the following conclusions from our evaluation experiments:

1. TutorM is far ahead of its counterparts (Hypermapper 2.0 and NSGA2) and achieves optimal results while sparing function evaluations.

2. Parameter analysis for TutorM shows that results are significantly improved by combining solutions with multiple surrogates (solution stack).

3. When the possibility of scaling up the surrogate portfolio was tested, we determine that dynamically selecting an appropriate surrogate model for the specific dimensionality of the problem is essential.

Conclusion

In this thesis, we propose a strategy for dynamic composition of surrogate models which allows the use of a surrogate portfolio for tuning black-box functions. Our investigation revealed that current surrogate-based optimization operates with a single type of model or the static combination of several varieties. This type of approach lacks variability and cannot be adapted for arbitrary problems. Our research goal was to decompose model-based multi-objective optimization into reusable and comparable components. To achieve this goal we make following research contributions:

1. First, we developed a compositional model for an arbitrary type of surrogate model. We established and implemented a component that combined several models into one surrogate hypothesis [**RG1**]. Nevertheless, for an arbitrary, unknown problem, we still require dynamic integration of surrogates into a composite model.

2. Second, we adapted the cross-validation technique to validate and compare surrogate models. A multi-step validation is essential to avoid the model underfeed and overfeed. Validation information enables us to dynamically decide on picking the right models or use the sampling plan as a default variant [**RG3**].

3. Third, we implemented a surrogate portfolio that combines the functionality from the preceding paragraphs. The portfolio allows the dynamic selection and combination of multiple surrogate models that are concerned with a concrete problem. This property means that a portfolio can offer more than one surrogate hypothesis for optimization [**RG2**].

4. Fourth, we improved the variability and extensibility not only of surrogate models but also of optimization algorithms. This improvement creates the possibility to combine solutions into a stack to reduce overall error.

In sum, these contributions enabled us to achieve results comparable to the state-of-the-art NSGA2 optimization algorithm in a wide range of optimization tasks. For almost all problems, our approach has demonstrated a significant advantage over all solution criteria. Analysis of the parameters showed that the most significant influence on results was made by solution combination (assumptions about the Pareto front). We have implemented a dynamic sampling plan that selects additional random points if there is no valid model. This strategy improved exploration-exploitation balance, which is determined for each optimization problem independently, and that led to the overall improvements in the results. The next crucial issue that we addressed is the optimization of multidimensional space. We have shown that a surrogate model can be applied to a small number of objectives but can be inappropriate if the

objectives are multiplied. The optimal solution for this issue is a flexible combination of better models at each optimization iteration.

We consider that the results accomplished in this thesis can be useful for improving parameter tuning and for overall model-based optimization.

Future Work

In this thesis we have developed the strategy that has a component structure and a unified interface. All major components are easily replaceable and scalable. A strong feature of our solution is the adaptation of optimization to a scaled unknown problem. That is why further integration with the *software product line* is a promising improvement. The right solution for this is - a software product line for parameter tuning. It has the necessary key features such as stop condition, noisy experiments and distributed architecture. The integration of this thesis into BRISE will improve its variability and scalability.

There are other several directions that we aim to focus on in the future.

- Promising results have been obtained for the combination of optimization techniques with surrogate modes. Further investigation in extensive parallel combination of *surrogate models and optimization algorithms* could significantly improve optimization results.

- It is advisable to change the composition of the portfolio to discard those models that are performing poorly. This *dynamic model collection* for the surrogate portfolio could improve the exploration of new models and reduce time costs.

Appendix

Table 1: Results of 5 repetitions for ZDT problem set: Function evaluation budget is 1000. The baseline is the NSGA2 with 50000 evaluations (100 population size in 500 generations)

Problem	Approach	↓ p-distance	↑ Hypervolume %	↑ ndf size %	↑ ndf space
ZDT1	Baseline	1,08e-05	99,78	0,16	0,22
	TutorM	**4,74e-05**	**100**	**90,33**	**1**
	NSGA2 1k	0,02	89,86	9,66	0,08
	Hypermapper 2.0	0,12	98,93	10,36	0,04
ZDT2	Baseline	1,04e-14	99,79	0,16	0,29
	TutorM	**0,00013**	**100**	**86,87**	**1**
	NSGA2 1k	0,01	88,78	8,82	0,06
	Hypermapper 2.0	0,18	97,31	5,12	0,04
ZDT3	Baseline	1,69e-08	100	0,16	0,39
	TutorM	**0,00012**	**99,47**	**86**	**1**
	NSGA2 1k	0,02	89,92	9,82	0,28
	Hypermapper 2.0	0,31	92,03	5,64	0,12
ZDT4	Baseline	2,04e-05	100	0,72	1
	TutorM	**0,01**	**99,80**	**50**	**0,78**
	NSGA2 1k	0,04	83,43	8,77	0,19
	Hypermapper 2.0	0,90	97,32	5,42	0,11
ZDT6	Baseline	0,0003	100	0,72	1
	TutorM	**0,09**	**99,43**	4,26	0,17
	Hypermapper 2.0	1,12	82,86	**6,25**	0,08
	NSGA2 1k	1,29	83,84	1,01	0,04

Table 2: Results of 5 repetitions for DTLZ problem set: Function evaluation budget is 1000. The baseline is the NSGA2 with 50000 evaluations (100 population size in 500 generations)

Problem	Approach	↓ p-distance	↑ Hypervolume %	↑ ndf size%	↑ ndf space
DTLZ1	Baseline	0,800	100	0,24	1
	NSGA2 1k	**3,277**	56,577	**1,56**	0,046
	TutorM	51,611	**98,163**	0,54	0,058
	Hypermapper 2.0	74,251	86,173	0,78	0,049
DTLZ2	Baseline	5,19e-06	98,603	0,24	0,39
	TutorM	**0,0004**	**100**	**82,56**	1
	NSGA2 1k	0,003	80,415	10	0,301
	Hypermapper 2.0	0,058	76,103	2,84	0,063
DTLZ3	Baseline	0,4	100	0,24	1
	NSGA2 1k	**4,430**	74,937	0,82	0,037

Problem	Approach	↓ p-distance	↑ Hypervolume %	↑ ndf size%	↑ ndf space
	TutorM	38,735	**97,743**	0,40	0,045
	Hypermapper 2.0	92,228	95,010	0,70	0,047
DTLZ4	Baseline	8,81e-06	100	0,36	1
	TutorM	**0,001**	**99,829**	**30,68**	0,666
	NSGA2 1k	0,002	87,807	9,60	0,323
	Hypermapper 2.0	0,059	64,579	1,18	0,029
DTLZ5	Baseline	1,62e-05	98,631	0,24	0,486
	TutorM	**0,0004**	**100**	**80,88**	1
	NSGA2 1k	0,002	81,729	10	0,434
	Hypermapper 2.0	0,058	78,463	3,02	0,06
DTLZ6	Baseline	0,009	100	0,24	1
	TutorM	**0,123**	**98,064**	3,70	0,142
	NSGA2 1k	1,011	54,258	2,88	0,128
	Hypermapper 2.0	1,657	18,355	2,22	0,084
DTLZ7	Baseline	2,42e-07	99,938	0,24	0,364
	TutorM	**0,0003**	**100**	**87**	1
	NSGA2 1k	0,160	92,891	3,04	0,128
	Hypermapper 2.0	0,781	91,129	2,24	0,081

Table 3: Results of 5 repetitions for WFG problem set: Function evaluation budget is 1000. The baseline is the NSGA2 with 50000 evaluations (100 population size in 500 generations)

Problem	Approach	↓ Hypervolume %	↑ ndf size %	↑ ndf space
WFG1	Baseline	100	0,72	1
	TutorM	**95,75**	3,44	**0,51**
	Hypermapper 2.0	44,12	**10,24**	0,31
	NSGA2 1k	30,52	3,18	0,28
WFG2	Baseline	100	0,08	0,63
	TutorM	**98,64**	**29,22**	**1**
	NSGA2 1k	85,96	6,44	0,35
	Hypermapper 2.0	62,35	1,20	0,10
WFG3	TutorM	**100**	**55,5**	**1**
	Baseline	99,05	0,08	0,29
	NSGA2 1k	84,46	9,72	0,15
	Hypermapper 2.0	73,31	2,44	0,02
WFG4	Baseline	100	0,72	0,60
	TutorM	**99,28**	**38,90**	**1**
	Hypermapper 2.0	84,39	3,26	0,06
	NSGA2 1k	83,95	10	0,58

Problem	Approach	↓ Hypervolume %	↑ ndf size %	↑ ndf space
WFG5	Baseline	100	0,20	0,24
	TutorM	**98,01**	**87,60**	**1**
	Hypermapper 2.0	84,83	34,74	0,06
	NSGA2 1k	82,70	10,00	0,18
WFG6	TutorM	**100**	**52,68**	**1**
	Baseline	99,30	0,20	0,33
	NSGA2 1k	86,59	10	0,27
	Hypermapper 2.0	83,21	2,36	0,03
WFG7	TutorM	**100**	**46,30**	**1**
	Baseline	99,30	0,20	0,33
	NSGA2 1k	86,39	10	0,26
	Hypermapper 2.0	83,14	2,36	0,04
WFG8	Baseline	100	0,20	1
	TutorM	**95,24**	**20,70**	**0,26**
	Hypermapper 2.0	86,74	2,80	0,07
	NSGA2 1k	79,63	9,54	0,20
WFG9	Baseline	100	0,20	0,85
	TutorM	**92,17**	**12,92**	**0,63**
	Hypermapper 2.0	80,80	7,30	0,24
	NSGA2 1k	73,56	10	1

References

1. **GALE: geometric active learning for search-based software engineering**
   Joseph Krall, Tim Menzies, Misty Davies
   IEEE Trans. Software Eng. (2015) https://doi.org/10.1109/TSE.2015.2432024
   DOI: 10.1109/tse.2015.2432024

2. **Nonlinear multiobjective optimization**
   Kaisa Miettinen
   Kluwer (1998)
   ISBN: 978-0-7923-8278-2

3. **Comparison of multiobjective evolutionary algorithms: Empirical results**
   Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele
   Evolutionary Computation (2000) https://doi.org/10.1162/106365600568202
   DOI: 10.1162/106365600568202

4. **Performance indicators in multiobjective optimization**
   Charles Audet, Sébastien Le Digabel, Dominique Cartier, Jean Bigeon, Ludovic Salomon
   (2018)

5. **Direct multisearch for multiobjective optimization**
   A. L. Custódio, J. F. Aguilar Madeira, A. Ismael F. Vaz, Luís Nunes Vicente
   SIAM Journal on Optimization (2011) https://doi.org/10.1137/10079731X
   DOI: 10.1137/10079731x

6. **On the complexity of computing the hypervolume indicator**
   Nicola Beume, Carlos M. Fonseca, Manuel López-Ibáñez, Luís Paquete, Jan Vahrenhold
   IEEE Trans. Evolutionary Computation (2009) https://doi.org/10.1109/TEVC.2009.2015575
   DOI: 10.1109/tevc.2009.2015575

7. **Fault tolerant design using single and multicriteria genetic algorithm optimization.**
   Jason R. Schott
   (1995)

8. **The asynchronous island model and NSGA-II: study of a new migration operator and its performance**
   Marcus Märtens, Dario Izzo
   Genetic and evolutionary computation conference, GECCO '13, amsterdam, the netherlands, july 6-10, 2013 (2013) https://doi.org/10.1145/2463372.2463516
   DOI: 10.1145/2463372.2463516

9. **Scalarizing functions in bayesian multiobjective optimization**
   Tinkle Chugh
   CoRR (2019) http://arxiv.org/abs/1904.05760

10. **On the impact of multiobjective scalarizing functions**
    Bilel Derbel, Dimo Brockhoff, Arnaud Liefooghe, Sébastien Vérel
    CoRR (2014) http://arxiv.org/abs/1409.5752

11. **Tutorial on evolutionary multiobjective optimization**
    Dimo Brockhoff, Tobias Wagner
    Proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary

computation (2015) https://doi.org/10.1145/2739482.2756574
DOI: 10.1145/2739482.2756574 · ISBN: 9781450334884

12. **A survey of multiobjective optimization in engineering design johan**
    Johan Andersson
    (2000)

13. **A survey of many-objective optimisation in search-based software engineering**
    Aurora Ramírez, José Raúl Romero, Sebastián Ventura
    J. Syst. Softw. (2019) https://doi.org/10.1016/j.jss.2018.12.015
    DOI: 10.1016/j.jss.2018.12.015

14. **Response surface methodology: Process and product in optimization using designed experiments**
    Raymond H. Myers, Douglas C. Montgomery
    John Wiley; Sons, Inc. (1995)
    ISBN: 0471581003

15. **Gaussian processes in machine learning**
    Carl Edward Rasmussen
    Advanced lectures on machine learning: ML summer schools 2003, canberra, australia, february 2 - 14, 2003, tübingen, germany, august 4 - 16, 2003, revised lectures (2004) https://doi.org/10.1007/978-3-540-28650-9_4
    DOI: 10.1007/978-3-540-28650-9_4 · ISBN: 978-3-540-28650-9

16. **Development of a multi-objective optimization algorithm using surrogate models for coastal aquifer management**
    George Kourakos, Aristotelis Mantoglou
    Journal of Hydrology (2013) http://www.sciencedirect.com/science/article/pii/S0022169412009511
    DOI: https://doi.org/10.1016/j.jhydrol.2012.10.050

17. **Interpolation of spatial data: Some theory for kriging**
    Roger Woodard
    Technometrics (2000) https://doi.org/10.1080/00401706.2000.10485731
    DOI: 10.1080/00401706.2000.10485731

18. **Gaussian processes for machine learning (GPML) toolbox**
    Carl Edward Rasmussen, Hannes Nickisch
    J. Mach. Learn. Res. (2010) http://portal.acm.org/citation.cfm?id=1953029

19. **Gaussian processes for machine learning**
    Carl Edward Rasmussen, Christopher K. I. Williams
    MIT Press (2006) http://www.worldcat.org/oclc/61285753
    ISBN: 026218253X

20. **Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels**
    Michael T. M. Emmerich, Kyriakos C. Giannakoglou, Boris Naujoks
    IEEE Trans. Evolutionary Computation (2006) https://doi.org/10.1109/TEVC.2005.859463
    DOI: 10.1109/tevc.2005.859463

21. **Practical solutions for multi-objective optimization: An application to system reliability design problems**
    Heidi A. Taboada, Fatema Baheranwala, David W. Coit, Naruemon Wattanapongsakorn

Rel. Eng. and Sys. Safety (2007) https://doi.org/10.1016/j.ress.2006.04.014
DOI: 10.1016/j.ress.2006.04.014

22. **ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems**
J. Knowles
IEEE Trans. Evolutionary Computation (2006) https://doi.org/10.1109/TEVC.2005.851274
DOI: 10.1109/tevc.2005.851274

23. **Practical design space exploration**
Luigi Nardi, David Koeplinger, Kunle Olukotun
2019 ieee 27th international symposium on modeling, analysis, and simulation of computer and telecommunication systems (mascots) (2019)

24. **A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms**
Tinkle Chugh, Karthik Sindhya, Jussi Hakanen, Kaisa Miettinen
Soft Computing (2017-12)
DOI: 10.1007/s00500-017-2965-0

25. **Hybrid simulation-optimization methods: A taxonomy and discussion**
Gonçalo Figueira, Bernardo Almada-Lobo
Simul. Model. Pract. Theory (2014) https://doi.org/10.1016/j.simpat.2014.03.007
DOI: 10.1016/j.simpat.2014.03.007

26. **Efficient global optimization of expensive black-box functions**
Donald R. Jones, Matthias Schonlau, William J. Welch
J. Global Optimization (1998) https://doi.org/10.1023/A:1008306431147
DOI: 10.1023/a:1008306431147

27. **Esa/pagmo2: Pagmo 2.12.0**
Francesco Biscani, Dario Izzo, Wenzel Jakob, GiacomoAcciarini, Marcus Märtens, Micky C, Alessio Mereta, Cord Kaldemeyer, Sergey Lyskov, Giacomo Acciarini, … Andrea Mambrini
Zenodo (2019-12) https://doi.org/10.5281/zenodo.3582877
DOI: 10.5281/zenodo.3582877

28. **JMetalPy: A python framework for multi-objective optimization with metaheuristics**
Antonio Benitez-Hidalgo, Antonio J. Nebro, Jose Garcia-Nieto, Izaskun Oregi, Javier Del Ser (2019)

29. **PlatEMO: A MATLAB platform for evolutionary multi-objective optimization**
Ye Tian, Ran Cheng, Xingyi Zhang, Yaochu Jin
IEEE Computational Intelligence Magazine (2017)

30. **mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions**
Bernd Bischl, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, Michel Lang (2017-03-09) http://arxiv.org/abs/1703.03373

31. **A case study on multi-criteria optimization of an event detection software under limited budgets**
Martin Zaefferer, Thomas Bartz-Beielstein, Boris Naujoks, Tobias Wagner, Michael Emmerich
Evolutionary multi-criterion optimization - 7th international conference, EMO 2013, sheffield, uk, march 19-22, 2013. proceedings (2013) https://doi.org/10.1007/978-3-642-37140-0_56
DOI: 10.1007/978-3-642-37140-0_56

32. **Multiobjective optimization on a limited budget of evaluations using model-assisted $\mathcal{S}$ -metric selection**
Wolfgang Ponweiser, Tobias Wagner, Dirk Biermann, Markus Vincze
(2008-09)
DOI: 10.1007/978-3-540-87700-4_78

33. **Planning and multi-objective optimization of manufacturing processes by means of empirical surrogate models**
T. Wagner
Vulkan Verlag (2013)

34. **Parameter tuning for self-optimizing software at scale**
Dmytro Pukhkaiev, Uwe Aßmann
CoRR (2019) http://arxiv.org/abs/1909.03814

35. **BOHB: robust and efficient hyperparameter optimization at scale**
Stefan Falkner, Aaron Klein, Frank Hutter
CoRR (2018) http://arxiv.org/abs/1807.01774

36. **Sequential model-based optimization for general algorithm configuration**
Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown
Learning and intelligent optimization - 5th international conference, LION 5, rome, italy, january 17-21, 2011. selected papers (2011) https://doi.org/10.1007/978-3-642-25566-3_40
DOI: 10.1007/978-3-642-25566-3_40

37. **SMAC v3: Algorithm configuration in python**
Marius Lindauer, Katharina Eggensperger, Matthias Feurer, Stefan Falkner, André Biedenkapp, Frank Hutter*< github.com automl smac3>; GitHub* (2017)

38. **An evolutionary algorithm with spatially distributed surrogates for multiobjective optimization**
Amitay Isaacs, Tapabrata Ray, Warren Smith
Progress in artificial life (2007)
ISBN: 978-3-540-76931-6

39. **A hybrid surrogate-based approach for evolutionary multi-objective optimization**
A. Rosales-Pérez, C. A. C. Coello, J. A. Gonzalez, C. A. Reyes-Garcia, H. J. Escalante
2013 ieee congress on evolutionary computation (2013)
DOI: 10.1109/cec.2013.6557876

40. **Evolutionary optimization with hierarchical surrogates**
Xiaofen Lu, Tao Sun, Ke Tang
Swarm Evol. Comput. (2019) https://doi.org/10.1016/j.swevo.2019.03.005
DOI: 10.1016/j.swevo.2019.03.005

41. **Efficient multi-objective optimization through population-based parallel surrogate search**
Taimoor Akhtar, Christine A. Shoemaker
(2019)

42. **A reinforcement learning hyper-heuristic in multi-objective single point search with application to structural fault identification**
Jiong Tang Pei Cao
CoRR (2018) http://arxiv.org/abs/1812.07958

43. **Design patterns: Elements of reusable object-oriented software**
Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
Addison-Wesley Longman Publishing Co., Inc. (1995)
ISBN: 0201633612

44. **Bayesian optimization with robust bayesian neural networks**
Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, Frank Hutter
Advances in neural information processing systems 29: Annual conference on neural information processing systems 2016, december 5-10, 2016, barcelona, spain (2016)
http://papers.nips.cc/paper/6117-bayesian-optimization-with-robust-bayesian-neural-networks

45. **The elements of statistical learning: Data mining, inference, and prediction**
Trevor Hastie, Jerome H. Friedman, Robert Tibshirani
Springer (2001) https://doi.org/10.1007/978-0-387-21606-5
DOI: 10.1007/978-0-387-21606-5 · ISBN: 978-1-4899-0519-2

46. **A subjective review of the state of the art in model-based parameter tuning**
Tobias Wagner
(2010-01)

47. **API design for machine learning software: Experiences from the scikit-learn project**
Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, … Gaël Varoquaux
(2013)

48. **[Scikit-learn: Machine Learning in Python]{.nocase}**
F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, … E. Duchesnay
Journal of Machine Learning Research (2011)

49. **A review of multiobjective test problems and a scalable test problem toolkit**
S. Huband, P. Hingston, L. Barone, L. While
IEEE Transactions on Evolutionary Computation (2006-10)
DOI: 10.1109/tevc.2005.861417

50. **Multi-objective genetic algorithms: Problem difficulties and construction of test problems**
Kalyanmoy Deb
Evol. Comput. (1999) https://doi.org/10.1162/evco.1999.7.3.205
DOI: 10.1162/evco.1999.7.3.205

51. **Scalable test problems for evolutionary multiobjective optimization**
Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, Eckart Zitzler
Evolutionary multiobjective optimization (2005) https://doi.org/10.1007/1-84628-137-7_6
DOI: 10.1007/1-84628-137-7_6

52. **A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II**
Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, T. Meyarivan
Parallel problem solving from nature - PPSN vi, 6th international conference, paris, france, september 18-20, 2000, proceedings (2000) https://doi.org/10.1007/3-540-45356-3_83
DOI: 10.1007/3-540-45356-3_83

53. **MOEA/D: A multiobjective evolutionary algorithm based on decomposition**
Qingfu Zhang, Hui Li

IEEE Trans. Evolutionary Computation (2007) https://doi.org/10.1109/TEVC.2007.892759
DOI: 10.1109/tevc.2007.892759

54. **GP-DEMO: differential evolution for multiobjective optimization based on gaussian process models**
Miha Mlakar, Dejan Petelin, Tea Tusar, Bogdan Filipic
European Journal of Operational Research (2015) https://doi.org/10.1016/j.ejor.2014.04.011
DOI: 10.1016/j.ejor.2014.04.011

55. **Deep kernel learning**
Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, Eric P. Xing
Proceedings of the 19th international conference on artificial intelligence and statistics, AISTATS 2016, cadiz, spain, may 9-11, 2016 (2016) http://proceedings.mlr.press/v51/wilson16.html

---

1. scikit-learn.org: sklearn.gaussianprocess.GaussianProcessRegressor↩
2. scikit-learn.org: sklearn.gaussianprocess.GaussianProcessRegressor↩
3. scikit-learn.org: sklearn.svm.SVR↩
4. scikit-learn.org: sklearn.neural network.MLPRegressor↩
5. scikit-learn.org: sklearn.ensemble.GradientBoostingRegressor↩