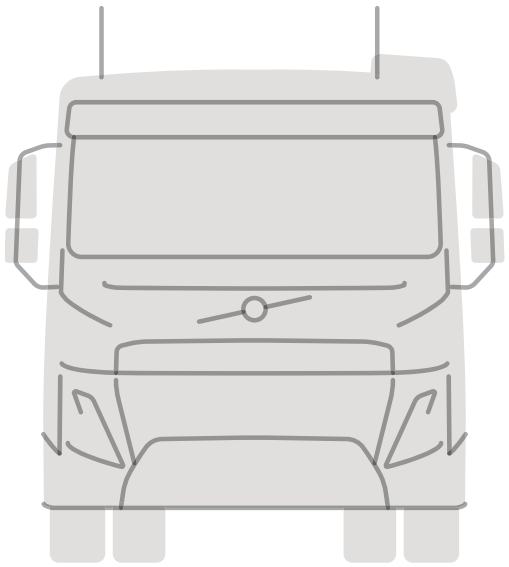


Multi-Objective Optimization Under the Hood

Engine Calibration via Metaheuristics and Probabilistic Methods

Master's thesis in Computer science and engineering

Sara Borg & Victor Hui

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Multi-Objective Optimization Under the Hood

Engine Calibration via Metaheuristics and Probabilistic Methods

Sara Borg & Victor Hui

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Multi-Objective Optimization Under the Hood
Engine Calibration via Metaheuristics and Probabilistic Methods
SARA BORG, VICTOR HUI

Supervisor: Firooz Shahriari Mehr, Data Science and AI
Advisor: Julien Ferri, Volvo GTT Powertrain Engineering
Examiner: Ashkan Panahi, Data Science and AI

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of Volvo truck.

Typeset in LaTeX
Gothenburg, Sweden 2025

Multi-Objective Optimization Under the Hood
Engine Calibration via Metaheuristics and Probabilistic Methods
SARA BORG, VICTOR HUI
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

The automotive industry faces the complex task of optimizing engine performance across diverse and often competing metrics, including fuel consumption and emissions. To effectively address this challenge and manage the necessary trade-offs, accurate engine calibration is essential. This thesis investigates the application of optimization methods, specifically Genetic Algorithms (GA) and Bayesian Optimization (BO), as a promising solution for engine calibration. Single-objective optimization targets the best solution for one goal, while multi-objective optimization balances trade-offs between conflicting goals to approximate the Pareto front, the set of optimal solutions where no objective can be improved without worsening another. This work explores both single-objective and multi-objective optimization implementations for GA and BO. In addition, a hybrid approach combining BO followed by GA is proposed for multi-objective optimization. The methods were evaluated in an experimental study. In the single-objective case, both GA and BO outperformed established internal benchmark values (provided by Volvo). For multi-objective optimization, GA, BO, and the hybrid method also achieved superior results. Across both single-objective and multi-objective problems, BO consistently delivered the best performance. These findings demonstrate that GA, BO, and the hybrid approach are viable strategies for engine calibration and provide a strong foundation for the development of more specialized calibration methods.

Keywords: Optimization, Multi-objective optimization, Genetic Algorithms, NSGA-II, NSGA-III, Bayesian Optimization, Engine calibration

Acknowledgements

We extend our sincere gratitude to our academic supervisor Firooz Shahriari Mehr and company advisor Julien Ferri for their invaluable feedback, insightful discussions, and steadfast guidance. A special thank you to Ali Ghanaati and Mahboubeh Farid as well for their additional support during the thesis. We also thank Volvo Group for providing the opportunity to conduct this master's thesis. Our thanks also go to Ashkan Panahi for his dedicated service as our examiner.

Finally, we are deeply grateful to our friends and family for their unwavering support throughout our university studies and during this thesis.

Sara Borg, Victor Hui, Gothenburg, 2025-08-27

Contents

List of Figures

List of Tables

List of Abbreviations

BO *Bayesian Optimization.* 2, 5, 7, 16, 17, 19–22, 24, 26, 29, 37, 38, 40–42, 45

CC *CO_2 Concentration.* 29, 31, 32, 35

CES *Combustion EATS Simulation.* 1, 3, 21, 25

EHVI *Expected Hypervolume Improvement.* 18, 26

EI *Expected Improvement.* 17–20, 26

FC *Fuel Consumption.* 29–32, 34, 35, 37, 38

GA *Genetic Algorithm.* 2, 5, 7, 11–14, 20–22, 24, 26, 29, 32, 37–42, 45

GP *Gaussian Process.* 16, 17, 19, 20, 23

HV *Hypervolume.* 10, 18, 24, 25, 28, 32, 38–40

KP *Key Point.* 1–3, 26, 29, 31, 32, 38–43

MO *Multi-Objective.* 3, 7–9, 20, 21, 24, 28, 31, 37, 38, 40, 41, 45

MOBO *Multi-Objective Bayesian Optimization.* 18, 24–26, 28, 34, 38–40, 42, 43, 45

NE *NOx Emissions.* 29, 31, 32, 34, 35, 38

NSGA-II *Fast Nondominated Sorting Genetic Algorithm.* 13–15, 24–27, 32, 34, 38–40, 43, 45

RWS *Roulette-Wheel Selection.* 12

SO *Single-Objective.* 7, 25, 28, 29, 37, 38, 45

SOBO *Single-Objective Bayesian Optimization.* 17, 23, 25, 26, 28, 29, 37, 38

SOGA *Single-Objective Genetic Algorithm.* 22, 24, 28–30, 37, 38, 40

TS *Tournament Selection.* 12, 22, 24, 25

1

Introduction

Modern engines contain hundreds of parameters that can be tuned to optimize towards multiple objectives simultaneously. These objectives include fuel efficiency, reduced energy consumption, and lower emissions.

In order to test these different parameter configurations virtually, Volvo Group uses in-house simulation models. In some instances, the tuning of parameters towards target objectives is guided by human intuition, and typically the focus is on single-objective optimization. This leads to some considerations:

1. Optimization guided by human intuition may result in an unstructured search process, increasing likelihood of overlooking better parameter configurations.
2. Furthermore, human-guided search typically depends on prior domain knowledge in order to be conducted effectively. This reliance introduces challenges when domain conditions shift or new factors are introduced.
3. The focus on single-objective optimization can lead to oversimplified solutions, as it ignores the need to balance multiple, potentially conflicting objectives.

Taking these considerations into account, this thesis explores different multi-objective approaches capable of operating without human oversight while minimizing the reliance on prior domain knowledge. The approaches will be implemented and evaluated with the objective of optimizing outputs derived from an engine cycle simulation. This is known as engine calibration.

1.1 Background

The engine cycle simulation is performed in Simulink using Volvo's internal model, the *Combustion EATS Simulation* (CES) platform. In this thesis, the simulation inputs will be limited to five parameters which are the actuators for the specific combustion engine model used for the simulations. Besides these five actuators, the engine speed and engine torque can also be varied. An engine *Key Point* (KP) is a specific operating condition or state of an engine. In this thesis a KP refers to a specific combination of engine speed and engine torque. Each KP introduces its own calibration search space that may feature narrower or wider constraints. As a result, the calibrations are KP-specific and the optimization difficulty may vary from one KP to another.

Beyond considerations specific to the setup used in this thesis, engine calibration introduces several challenges from an optimization perspective. Thus, it is important to select optimization techniques that addresses these challenges. The following are the challenges that have been identified:

Black-box system: The simulation functions as a black-box system. This is a system where the internal workings are either unknown, too complex, or inaccessible. As a result, only input-output relationships are observable and used for the optimization. This renders many classical optimization methods unusable, as they rely on knowing the exact correlation between inputs and outputs. Consequently, an optimization technique suitable for use on black-box systems is required.

Non-convex problem: Local optima refers to suboptimal solutions that appear to be optimal within limited regions of the search space, potentially trapping optimization algorithms and causing premature convergence. Problems with many such local optima are referred to as *non-convex*. In the context of combustion engine calibration, parameters like injection timing and EGR[1] rate interact nonlinearly with objectives like fuel consumption and NOx emissions. Thus, small parameter changes can cause abrupt shifts in combustion efficiency or emissions, creating a non-convex problem. This issue is further exacerbated when moving from single-objective optimization to multi-objective optimization due to the increased number of objectives. Since the goal of this thesis is to approximate the global optima as closely as possible, the chosen optimization techniques needs to be able to handle non-convex problems.

Input/Output constraints: In real-world scenarios there are constraints to the engine parameters and objectives (inputs and outputs of the simulation). These constraints need to be adhered to in order for a parameter configuration to be considered valid. There is little interest in invalid configuration, thus the optimization technique must incorporate constraint handling to ensure the generation of valid configurations.

To address these challenges, this thesis investigates optimization techniques that can operate without prior knowledge of the objective function, are effective for non-convex problems, support multi-objective optimization and incorporate constraint handling. There are two methods that fulfill these criteria, which this thesis will focus on: *Genetic Algorithm* (GA) [1], which are metaheuristic-based, and *Bayesian Optimization* (BO) [2], a probabilistic approach. In addition, a hybrid model that combines elements of GAs and BO will also be explored.

1.2 Aim

The aim of this thesis is to implement a framework for the multi-objective optimization of a limited set of input parameters for engine calibration at individual KPs. To demonstrate the feasibility of the approach, the work begins with a single-objective

[1] Exhaust Gas Recirculation.

optimization model. Building upon this foundation, a multi-objective optimization framework is developed, within which various methods are explored and compared in order to evaluate their respective strengths, limitations, and suitability for engine calibration tasks.

1.3 Limitations

While this thesis explores promising approaches for multi-objective optimization, several limitations affect the scope, general applicability, and depth of the findings. This section outlines key limitations and the measures taken to address or simplify them.

Multi-objective optimization: In *Multi-Objective* (MO) optimization, finding a single optimal solution is typically impossible because the objectives often conflict, meaning improving one objective may worsen another. As a result, the goal is to identify a set of trade-off solutions known as the Pareto front (see Section 3.1.3), rather than a single "best" solution. However, obtaining all Pareto optimal solutions is computationally infeasible because continuous-variable problems like engine calibration have infinitely many such solutions, particularly when each function evaluation is costly. Instead, MO algorithms typically aim to approximate the Pareto front by producing a diverse and representative subset of near-optimal solutions that the decision maker can choose from.

Key point-specific calibration: The calibration process is KP-specific, meaning that near-optimal calibrations are determined separately for each KP. Given that there are over 40 such KPs in this context, analyzing all of them would be unmanageable. Therefore, the focus is limited to highlight two types of KPs:

1. Internal KPs: Key points corresponding to mid-range torque levels relative to engine speed.
2. High-torque KPs: Key points characterized by higher torque demands, which tend to result in less stable behavior.

Simulation model choice: For development, the focus will be on the CES platform because of its lighter computations and faster run time. There are larger simulation platforms that simulate the entire vehicle's performance, but for developing the optimization methods, the CES is sufficient. With future work in mind, the implemented optimization methods should be model-agnostic. As a result, it should be able to be applied on any other black-box function with minor modifications. This was taken into consideration throughout the development of the methods.

Search space dimensionality: In the context of optimization, the complexity increases rapidly with the number of decision variables (inputs). Throughout this thesis, the specific engine model used for the CES simulations contains hundreds of

parameters. However, only five actuator parameters are considered in the optimization, as this restriction is dictated by the specific engine model itself. This keeps the search space to a manageable five dimensions. Therefore, when working with other simulation models or black-box functions, some form of parameter reduction may be necessary if the dimension increases. This more complex consideration is left for future work.

Confidentiality of simulation details: Specific details regarding the inputs, outputs, as well as details about the simulation models and input parameters are obfuscated or generalized as part of the intellectual property agreement with Volvo.

1.4 Thesis Outline

Chapter 2 presents previous relevant findings and contributions that form the foundation of this thesis. In Chapter 3, the theoretical background necessary for understanding the thesis is outlined, covering key concepts in optimization and the algorithmic methods used. Chapter 4 details the implementation of the models and the steps taken, along with a description of the experimental setup and evaluation procedures. Chapter 5 then presents the results gathered from the experiments of this project. It will display the performance of the implemented models in multiple problem formulations. Chapter 6 follows with a discussion of the algorithmic choices made throughout the project and a review of the results. Finally, Chapter 7 summarizes the key insights and concludes the thesis.

2

Related Works

This chapter focuses on research contributions relevant to this thesis, demonstrating a variety of methods and their potential applications to solving similar challenges in engine calibration and optimization tasks.

The application of both GAs and BOs in engine calibration is explored in [3]. Zhu *et al.* state that both GA and BO find solutions very close to the single-objective global optimum in their experimental study done on their high-fidelity engine model. The results demonstrate that both methods can succeed in an engine calibration context. This thesis aims to replicate the success of their work with GAs and BO in the context of engine calibration. Furthermore, it extends their approach to address multi-objective optimization.

Hybrid optimization, the combination of two or more optimization strategies, is an interesting area of research in optimization. It is an approach suggested by Yu *et al.*, in [4]. The authors state that while optimization methods such as GAs have been validated in engine calibration, novel optimization approaches are preferred as they produce better optimization results. A suggested novel approach is the combination of metaheuristic optimization approaches with machine learning techniques. This thesis investigates this approach through the combination of the metaheuristic GA and the machine learning-based BO.

Moving away from engine calibration specifically, sequential hybrid approaches have been proposed for the parameter calibration in sand models [5] and hyperparameter optimization of deep neural networks in demand forecasting [6]. The hybrid method proposed in [5] uses Bayesian optimization to quickly locate a promising region over a large multi-parameter search space. A GA is then used to refine the solution within this BO-identified region. However, this approach only focuses on minimizing an error function, making it a single-objective problem. Similarly, [6], proposes a sequential approach to optimize hyperparameters for minimizing forecasting error in a long short-term memory (LSTM) network. This approach is also limited to single-objective optimization. This thesis builds upon the sequential hybrid approach proposed in these two papers by extending it to support multi-objective optimization and adapting it to the context of engine calibration.

To summarize, our work builds on the foundation laid by previous works by attempting to replicate their success and adapting their methods to the context of engine calibration. In addition we will be investigating whether their approach remains

effective in a multi-objective setting.

3

Theory

This chapter presents the theoretical background of various optimization methods necessary to understand the developed optimization models. It begins with an overview of optimization, introducing both single-objective and multi-objective cases. The chapter then introduces GAs and BO, outlining their basic concepts and roles in optimization. Lastly, the concept of hybrid optimization is presented, highlighting how combining GAs and BO can lead to more efficient and effective exploration of the objective space in multi-objective problems.

3.1 Optimization

In short, *optimization* is the process of finding the best solution from a set of feasible solutions by maximizing or minimizing one or more objective functions. When a single objective function is considered, it is referred to as *Single-Objective* (SO) *optimization*, whereas when there are two or more objective functions, it is called *Multi-Objective* (MO) *optimization*.

3.1.1 Single-objective optimization

A single-objective optimization problem can be written in the following form:

$$\min_{\mathbf{x}\in\Omega} \quad f(\mathbf{x}). \tag{3.1}$$

The function $f(\mathbf{x})$ is called the *objective function*. The vector $\mathbf{x}$ is a n-dimensional vector composed of n independent variables: $\mathbf{x} = [x_1, x_2, ...x_n]^T$. The set Ω is called the *feasible set*, representing the set of all feasible solutions relative to potential constraints [7]. The optimization problem is then to find the vector $\mathbf{x}$ over all possible vectors in Ω that results in the minimum objective value of $f(\mathbf{x})$.

The classic methods to solve optimization problems are analytical techniques that usually involve differential calculus based on the objective function(s), knowing that they are continuous and differentiable. Most of the common methods are gradient-based, which rely on the gradient of the function to find optima.

3.1.2 Constrained optimization

Most real-world problems have *constraints* that all solutions need to account for. In the context of engine calibration, an example of this would be if the exhaust temper-

ature of the engine exceeds some predefined threshold. In that case, the calibration would be infeasible and hold no practical value. Constrained optimization algorithms thus require some constraint-handling process so that feasible calibrations are produced more often.

A common approach is to either discard solutions that violate constraints or penalize them by modifying their objective values. These techniques, known as penalty methods, can apply fixed penalties or adaptive penalties that reflect the degree of constraint violation [8]. Using penalty methods inherently transforms the constrained problem into an unconstrained one but this comes with some issues. With too aggressive penalties, the likelihood of premature convergence increases, while with too modest penalties, the likelihood that the solution violates a constraint increases [8].

3.1.3 Multi-objective optimization

In single-objective optimization, the goal is to find the best solution that corresponds to a minimum or maximum value of the objective function [9]. However, there are scenarios where optimization of multiple targets is desired, and thus MO optimization approaches have been developed [9].

A general multi-objective problem with N objectives can be formulated as follows [10]:

$$\begin{aligned} \min_{\mathbf{x}\in\Omega} \quad & \mathbf{y} = [f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_N(\mathbf{x})]^T \\ \text{subject to} \quad & g_j(\mathbf{x}) \leq 0, \quad j = 1, 2, \ldots, M, \end{aligned} \tag{3.2}$$

where $\mathbf{y}$ is the objective vector, g_j are the potential constraints and $\mathbf{x}$ is the P-dimensional vector representing the decision variables within the parameter space Ω. When $N = 1$, the problem is reduced to the single-objective problem as in Equation (3.1). In cases where $N > 1$, the overall best solution generally does not exist since the individual objective functions are typically conflicting. Hence, a typical multi-objective algorithm tries to balance these contradicting objectives and returns a set of solutions, so called nondominated, that each represent different compromises or trade-offs between the objectives [10].

The classical approach to solving multi-objective problems is to assign weights to each (normalized) objective function and then minimize/maximize the aggregated sum, converting it to a single-objective problem. The following problem definition shows the weighted aggregation approach using linear scalarization with N objective functions [11]:

$$\min_{\mathbf{x}\in\Omega} \quad \sum_{i=1}^{N} w_i * f_i(\mathbf{x}),$$

where $w_i \geq 0$ and $\sum_{i=1}^{N} w_i = 1$, which are assumed to be user-defined constants. This approach rely on the user to define a weighted "importance" of each objective.

The solution will yield a single solution that is as good as the selection of the weights. However relying on intuition is not something that is desired in this thesis.

Pareto optimality: Pareto optimality is the most commonly used term for comparing solutions against each other in multi-dimensional objective spaces. A solution is Pareto optimal if no other solution can improve at least one objective without worsening another. This introduces the concept of *dominance*. Let f_i denote the i-th objective function, a solution x is said to dominate y ($x \succ y$) by the following definition:

$$\begin{aligned} x \succ y \;\Leftrightarrow\; & \forall i,\ f_i(x) \leq f_i(y) \text{ and} \\ & \exists j,\ f_j(x) < f_j(y), \end{aligned} \tag{3.3}$$

where $i, j \in \{1, ..., m\}$ index the set of m objective functions.

The solutions that are Pareto optimal are then the solutions that are *nondominated*, meaning it is not dominated by any other solution. The ability to rank the solutions is very important for any MO optimization model to distinguish the quality between any solution in a solution set.

The *Pareto front* is the set of all Pareto optimal solutions to any given problem. In continuous problems, the Pareto front is then infinitely large. The goal of most multi-objective algorithms is to generate a Pareto set that covers the Pareto front as much as possible. However, when a problem has more than three objectives, it forms a *many-objective optimization problem*. The issue with Pareto dominance in these problems is that with an increasing number of objectives, the nondominated set of solutions also increases. Therefore, pure Pareto-based multi-objective algorithms have been shown to be less useful to discriminate among solutions in many-objective problems [12].

Multi-objective performance metrics: The performance metrics used to measure multi-objective optimization algorithms considers three aspects [13]:

- *Cardinality* reflects the amount of solutions provided. Intuitively, a larger number of solutions is preferred because it increases the decision-maker's flexibility to choose a trade-off that best fits a specific application.
- *Accuracy* measures the convergence of the solution set. It indicates the distance from the true Pareto front. High accuracy is important to ensure that the returned solutions are near-optimal, rather than suboptimal results that might mislead decision-making.
- *Diversity* refers to the distribution and spread of the solutions. Distribution refers to the relative spacing between solutions, while spread refers to the range of values covered by the solutions. Good diversity ensures that the solutions are well-distributed across the entire Pareto front, rather than clustered in one region. Without sufficient diversity, important trade-off regions may be missed, limiting the usefulness of the optimization results.

Most performance metrics that measure accuracy require either prior knowledge about the true Pareto front or a reference point or set. The most commonly used metric is the *Hypervolume* (HV) [13], which measures the volume of the objective space that is dominated by a set of solutions and bounded by a predefined reference point. This reference point, often chosen to represent the worst-case objective values (also known as the *nadir point*), is essential for computing the size of the dominated region. The HV metric is particularly powerful as it captures both the accuracy and the diversity of the solution set. As illustrated in Figure 3.1, the HV corresponds to the gray area enclosed between the reference point r and the set of Pareto-optimal solutions $P = \{p^{(1)}, p^{(2)}, \cdots, p^{(n)}\}$. Each point in the set contributes to expanding the boundary of this dominated space, and the total HV is the union of all such dominated regions.

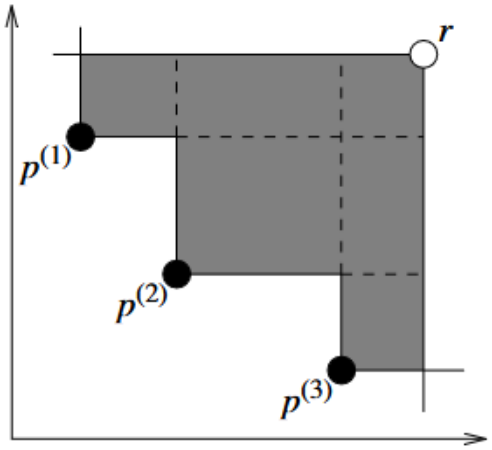


Figure 3.1: Hypervolume measure of the region dominated by the Pareto set $P = \{p^{(1)}, p^{(2)}, \cdots, p^{(n)}\}$ and bounded by the reference point r. Figure taken from [14].

The calculation of the HV can be seen in Equation (3.4). Here, $\mathcal{L}(\cdot)$ represents the Lebesgue measure, a generalization of length, area, and volume for higher dimensional spaces [15]. The expression defines the HV as the measure of the union of all points b in the objective space dominated by some solution $a \in A$ and that themselves dominate the reference point r:

$$HV(A, r) = \mathcal{L}(\cup_{a \in A}\{b | a \succ b \succ r\}). \tag{3.4}$$

3.1.4 Black-box optimization

Black-box optimization refers to a class of problems where optimal solutions are found without using any derivative information of the objective function(s). Typically, this is the case when the objective function is unknown, and only the inputs and outputs can be observed. In the literature it is also known as derivative-free optimization, gradient-free optimization and simulation-based optimization [16].

3.2 Genetic algorithms

Genetic Algorithms (GAs) are optimization techniques inspired by the principles of natural selection and genetics. They are a part of the broader class of *evolutionary algorithms* used to solve complex optimization and search problems [17]. The three main characteristics of an evolutionary algorithm, and thus a GA, are that they are population-based, fitness-oriented, and variation-driven [17]. These characteristics allow GAs to explore a wide solution space while contiguously refining solutions toward better performance.

Evolutionary algorithms maintain a group of candidate solutions and iteratively tune them using biologically inspired operations such as selection, crossover, and mutation. Since they do not follow a deterministic search pattern, they are classified as *metaheuristic* optimization techniques [17]. Metaheuristics, including GAs, do not guarantee finding the global optimum but are designed to efficiently explore complex search spaces and provide sufficiently good solutions [18]. Their performance is highly dependent on the specific problem at hand.

GAs are particularly effective for non-convex, high-dimensional black-box optimization problems due to their unique search mechanisms and flexibility. They are well suited for black-box functions as they require only objective function evaluations. Additionally, the population-based search allows for parallel evaluation of solutions, accelerating exploration in high-dimensional spaces.

In optimization, a GA consists of several key components and terminology that define its process [17]:

- *Individual*: A potential solution to the problem, typically represented as a vector of parameters (also referred to as *genes*).
- *Population*: The group of candidate individuals that evolves over multiple *generations*.
- *Fitness*: An individual's performance based on its objective function value.
- *Selection*: A mechanism for selecting individuals with better fitness values, that is, those with lower objective function values in minimization problems. The selected individuals contribute to the next generation.
- *Variation (crossover & mutation)*: Genetic operators that introduce changes in individuals to explore new solutions and maintain diversity within the population.
- *Convergence*: Over successive generations, the population ideally moves toward an optimal or near-optimal solution.

3.2.1 Stages of a simple genetic algorithm

A simple GA follows an iterative process consisting of the following steps:

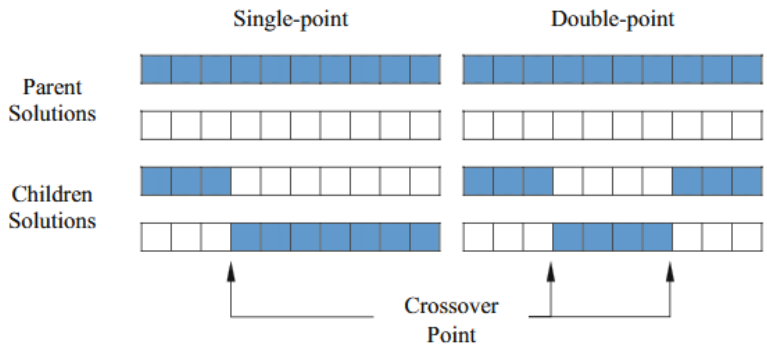


Figure 3.2: Single-point crossover to the left, and double point crossover to the right. In the single-point cross over, the chromosomes of two parent solutions are swapped before and after a single point. In the double-point crossover, there are two cross over points and the chromosomes between the points are swapped only. Figure taken from [1].

1. **Initialization**: The algorithm begins by generating an initial population of individuals within the solution space, either randomly or based on heuristics.

2. **Mating selection**: A subset of individuals are chosen from the population to form a *mating pool*. Selection is typically biased toward individuals with higher fitness, ensuring that better solutions have a higher probability of contributing to the next generation. Common selection methods include *Tournament Selection* (TS) and *Roulette-Wheel Selection* (RWS) [1], [19].

   In TS, a small group of individuals is randomly selected from the population, and the one with the highest fitness is chosen as the parent [19]. In RWS, each individual is assigned a probability of being selected based on its fitness relative to the total fitness of the population [1].

3. **Crossover**: Pairs of selected individuals, or *parents*, then undergo a crossover operation to create new offspring. This combines the genetic information of two parents to generate one or more child solutions. In the common GA, the crossover operation is done with a predefined crossover probability. This means that if crossover is not performed, the offspring generated will be the copies of the original parent solutions.

   Figure 3.2 illustrates two common crossover techniques: single-point and double-point. These can be generalized to any arbitrary number of crossover points, as k-point crossover.

4. **Mutation**: To maintain diversity and prevent premature convergence, mutation is applied to offspring by making small random alternations to their genetic representation. These random alternations can be applied in different ways. One of the most common is *uniform mutation*, where a gene or parameter is replaced with a completely new value sampled uniformly within its bounds. Alternatively, the mutation can involve a small variation, an increase or decrease, relative to the current value. Similar to crossover, mutation is

only performed with a predefined probability.

5. **Survival selection**: The newly generated offspring replace some or all individuals in the existing population, forming the next generation. This process repeats for multiple generations. *Elitism* is an operator that can be implemented in this step, in which one or multiple best solutions are kept and transferred without alternation to the next generation [1].
6. **Stopping criteria**: The algorithm terminates when a predefined condition is met. The stopping criteria is commonly implemented as a predefined number of generations, a lack of significant improvement in fitness, or reaching a known optimal solution.

The flow of this algorithm is illustrated in Figure 3.3.

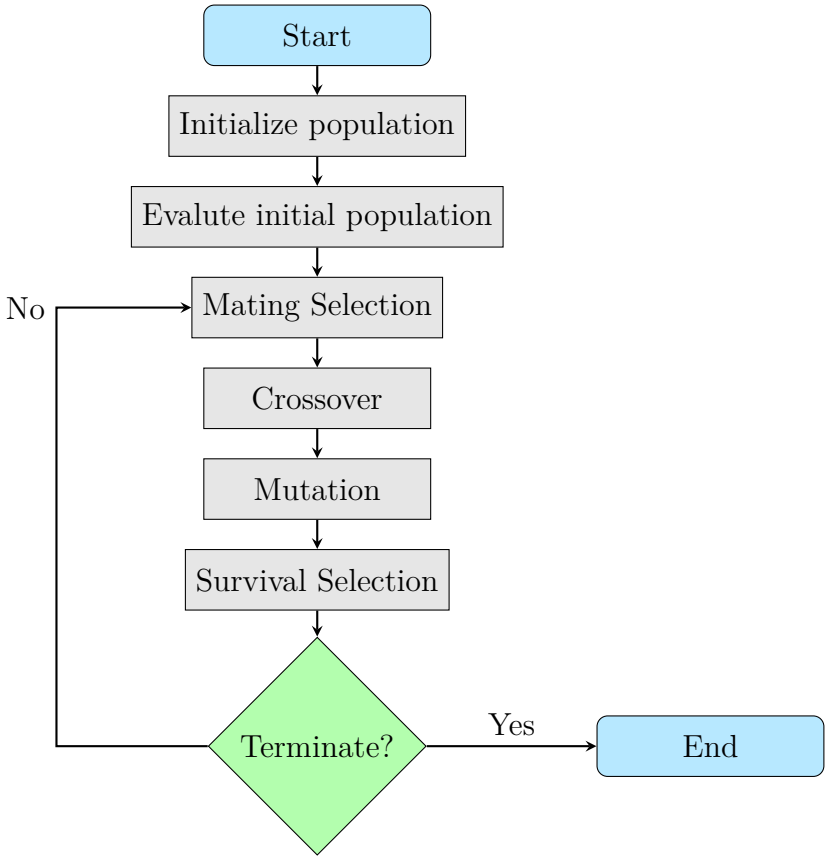


Figure 3.3: Flowchart describing the GA.

3.2.2 Multi-objective genetic algorithms

Since GAs are population-based approaches and finds multiple solutions in each run, they are well-suited for solving multi-objective optimization problems [20]. A multi-objective genetic algorithm alters the GA to find a set of multiple nondominated solutions in each generation. This is the core aspect of any multi-objective GA.

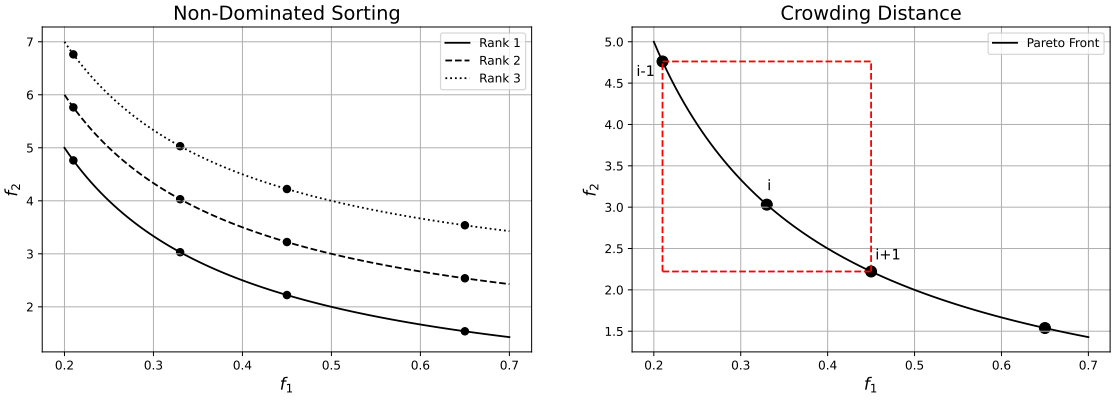
For problems with 2 or 3 objective functions, the *Fast Nondominated Sorting Genetic Algorithm* (NSGA-II) proposed by Deb *et al.* [21] is widely used. It follows

the general outline of a GA with modified mating and survival selection steps. It does this by introducing a *nondominated sorting* technique and measures *crowding distance* to maintain diversity.

The nondominated sorting is a technique to classify and rank solutions based on Pareto dominance. Firstly, each individual p is compared to the others to see if it is dominated, keeping track of a domination count n_p that shows how many individuals dominate p as well as a domination set S_p with all individuals that p dominate. If the domination count n_p is zero, this means that individual p is nondominated and is included in the first front $\mathcal{F}_1$. The rest of the population is then divided into nondominated fronts, where the second front consists of individuals dominated only by the ones in the first, the third only by the ones in the second and first, and so on. Figure 3.4a illustrates how the individuals are ranked, and pseudo-code of the procedure can be found in Algorithm 1 in Appendix A.1.

Then in the survival step, NSGA-II first selects individuals front-wise. At the first front that will not entirely fit all individuals to the population, they are instead selected by their crowding distance. The main aspect for this is diversity preservation, thus maintaining a good spread of solutions. This crowding metric is an estimate of the density of individuals surrounding a specific individual in the population [21]. The crowding distance of an individual, $i_{distance}$, is measured as the average distance of the two nearest points (in the same front) on either side of i along each objective. Figure 3.4b illustrates this. This requires a prior sorting on the population according to each objective function value in ascending order. Then, for M objectives, this is calculated as:

$$i_{\text{distance}} = \sum_{m=1}^{M} \frac{f_{i+1}^{(m)} - f_{i-1}^{(m)}}{f_{\max}^{(m)} - f_{\min}^{(m)}}. \tag{3.5}$$



(a) Non-dominated sorting into ranks based on dominance. In this example, the individuals are placed into three ranks, each illustrating a front.

(b) The crowding distance of individual i is calculated based on the cuboid formed by the nearest neighbors, $i-1$ and $i+1$, in each direction.

Figure 3.4: Illustrations of non-dominated sorting and crowding distance in multi-objective optimization.

For constraint-handling, NSGA-II can be modified to use a new domination definition in the sorting step [21]. An individual i is said to constraint-dominate j, if any

of the following is true.

- Individual i is feasible and j is not.
- Individuals i and j are both infeasible, but individual i has a smaller overall constraint violation.
- Individual i and j are feasible and individual i dominates individual j.

This ensures that feasible individuals are ranked higher than infeasible ones, and among the infeasible individuals, those with smaller constraint violations receive better ranks [21].

As mentioned in Section 3.1.3, the number of nondominated individuals increases significantly when introducing more objectives, forming the *many-objective optimization problems*, which are defined as those with four or more objectives. This has been shown to make the NSGA-II less efficient as there is not much room for creating new individuals in each generation, which slows down the search process [22]. Additionally, the evaluation of the diversity measure (crowding distance in NSGA-II) become computationally expensive for large-dimensional spaces and struggles to uniformly distribute individuals across the high-dimensional Pareto front. Thus, a new modification of NSGA was developed using reference-points, known as NSGA-III [22].

NSGA-III replaces NSGA-II's crowding distance with a reference-point system to guide diversity and convergence. It keeps the nondominated sorting to rank the population but uses these reference points for final selection in the survival step. When no preference information is available, the reference points can be of any structured and uniform distribution across the objective space that is generated prior to the algorithm run [22]. In the survival selection, each individual is associated with the nearest reference point and selection prioritizes points with fewer associated individuals, ensuring uniform Pareto front coverage [22]. The overall procedure is presented in Algorithm 2 in Appendix A.2.

In order to account for constraints, NSGA-III uses the same constraint-domination principles as described for NSGA-II [23]. In the original unconstrained implementation of NSGA-III, there was no need to perform a mating selection operator, such as tournament selection, on the parent population to create the offspring population [22]. However, in the presence of infeasible solutions, it is crucial to consider the constraints to emphasize feasible solutions over infeasible ones, and smaller constraint violations over larger ones [23]. The resulting binary tournament selection operator between two individuals i and j is defined as follows:

- If i is feasible and j is not, select i. If j is feasible and i is not, select j.
- If i and j are infeasible and i has a smaller constraint violation select i, else if j has a smaller constraint violation, select j.
- If both i and j are feasible then i or j is chosen at random.

3.3 Bayesian optimization

Bayesian optimization (BO) is an optimization strategy for optimizing functions that are expensive to evaluate [24]. The strategy does not rely on internal knowledge of a function, such as gradients or closed-form expressions. Instead it uses a surrogate model to approximate the objective function which enables it to optimize black-box functions [2]. One of BO's major strengths is its sample efficiency [2], meaning it can achieve good optimization results with limited function evaluations. Some areas where BO is applied include tuning hyperparameters in machine learning algorithms [25], designing engineering systems [26], and the selection of laboratory experiments in materials and drug design [27].

The main components of BO are:

- *Surrogate model*: A Bayesian statistical model which acts as a surrogate to the objective function. Typically this is a *Gaussian Process* (GP) [24]
- *Acquisition function*: A function that uses information gained from the surrogate model in order to determine which point(s) to evaluate next.

3.3.1 Surrogate model

To understand the Bayesian nature of the model, it is necessary to first explain *Bayesian inference* which is based on Bayes' Theorem.

Bayes' Theorem uses conditional probabilities to describe the probability of an event occurring based on prior knowledge about that event. Denote A and B as events and $P(A \mid B)$ as the probability of A occurring given that B is true. Bayes' theorem states that

$$P(A \mid B) = \frac{P(A)P(B \mid A)}{P(B)}, \tag{3.6}$$

where $P(A)$ and $P(B)$ are the probabilities of observing A and B respectively without any prior conditions [28].

Bayesian inference allows the deduction of uncertain characteristics of a system based on observed data, by leveraging probability theory. In the context of optimization, the unknown characteristic could be the value of the objective function at a given location. For example, consider the point x and the value of the objective function at x being $\phi = f(x)$. The prior distribution (or simply prior) $P(\phi \mid x)$ represents the plausible values for ϕ before observing any data. The likelihood function (or simply likelihood) $P(y \mid x, \phi)$ describes the likelihood of getting the measurement y as a function of $\phi = f(x)$ at the point x. An updated posterior distribution (or simply posterior) can be derived by using the prior and the likelihood as follows [2]:

$$P(\phi \mid x, y) = \frac{P(\phi \mid x)P(y \mid x, \phi)}{P(y \mid x)}. \tag{3.7}$$

The denominator $P(y \mid x)$ is known as the marginal likelihood. It is a constant with

respect to $\phi = f(x)$ that ensures normalization [2]:

$$P(y \mid x) = \int P(y \mid x, \phi) P(\phi \mid x)\, d\phi. \tag{3.8}$$

When new information is made available, the previous posterior serves as the new prior and multiplying it with the likelihood and re-normalizing results in a new posterior. In BO, the objective function is approximated using a statistical model based on Bayesian inference. As previously mentioned, typically a GP is used.

The topic of GPs is quite complex and therefore this section will not provide an comprehensive overview. Instead a simplified description relevant for understanding its role in Bayesian optimization will be presented.

GPs or GP regression is a tool that models unknown functions [29]. Instead of trying to fit a single function to the data, a GP gives a distribution of functions that could fit the data and gives a measure of its uncertainty at each point. GPs utilize the Bayesian posterior distribution to predict potential values of $f(x)$ at point x, updating this distribution after each evaluation of $f(x)$ at a new point x.

3.3.2 Acquisition function

An acquisition function uses the surrogate model to select the next evaluation point(s), balancing the trade-off between exploration of uncertain regions and exploitation of promising areas. The most commonly used acquisition function in *Single-Objective Bayesian Optimization* (SOBO) is *Expected Improvement* (EI), as it is easy to use and performs well [24].

Expected Improvement: As introduced in Section 3.3.3, x_n denotes the point selected in iteration n, and y_n the corresponding function value. Let $f_n^* = \max_{m \leq n} f(x_m)$[1] represent the largest function value in dataset $\mathcal{D}$ after n iterations. Suppose that a new point should be selected for iteration $n+1$. If the new point selected is x, then a new function value $f(x)$ will be added to $\mathcal{D}$. Consequently, the largest function value in $\mathcal{D}$ will either be $f(x)$ (if $f(x) \geq f_n^*$) or f_n^* (if $f(x) \leq f_n^*$). Thus the improvement in the value of the largest observed value is $f(x) - f_n^*$ if this results in a positive value, and 0 otherwise. The improvement can be written more compactly as $[f(x) - f_n^*]^+$, where $a^+ = \max(a, 0)$. The idea behind EI is to select an x such that this improvement is as large as possible. Since $f(x)$ is unknown until after the observation, the expected value of this improvement is used instead, selecting the x that maximizes it. The EI is calculated as:

$$\mathrm{EI}_n(x) := \mathbb{E}_n \left[[f(x) - f_n^*]^+ \right], \tag{3.9}$$

where $\mathbb{E}$ is the expected value. $\mathbb{E}_n$ is calculated using the posterior distribution obtained from the statistical model. At iteration $n+1$, the EI algorithm selects the point x with the largest expected improvement,

$$x_{n+1} = \mathrm{argmax}_n \mathrm{EI}_n(x), \tag{3.10}$$

[1] This example assumes the objective is to maximize.

where argmax returns the input point at which a function is maximized [30]. There is an extended version of EI that solves constrained optimization problems of the form

$$\max_{c(\mathbf{x})\leq\lambda} f(x), \tag{3.11}$$

where $c(x)$ is a constraint function, sometimes it is observed together with the objective function f [31]. The constraint $c(x) \leq \lambda$ has to be satisfied in order for the observation at x to be considered feasible. In order to derive the expected constrained improvement acquisition function EI_C, two modifications have to be made to the standard EI acquisition function [31], [32].

1. f^* is now the value for the best **feasible** observation.
2. The improvement $[f(x) - f_n^*]^+$ is now 0 if x is infeasible, and otherwise the usual improvement.

EI_C can now be defined as,

$$\mathrm{EI}_C(x) = \mathrm{EI}(x)\mathrm{PF}(x), \tag{3.12}$$

where $\mathrm{PF}(x)$ is the probability that x is feasible.

An alternative version of EI used in *Multi-Objective Bayesian Optimization* (MOBO) is *Expected Hypervolume Improvement* (EHVI) [33].

Expected Hypervolume Improvement: As mentioned in Section 3.1.3, HV is a metric that can be used to measure the quality of a Pareto front. The *Hypervolume Improvement* (HVI) can be written as $\mathrm{HVI}(\mathcal{Y}, \mathcal{P}, r) = \mathrm{HV}(\mathcal{P} \cup \{\mathcal{Y}\}, r) - \mathrm{HV}(\mathcal{Y}, r)$, where $\mathcal{P}$ is the Pareto set, r is the reference point and $\mathcal{Y}$ is a set of points [33]. The EHVI is defined as,

$$\mathrm{EHVI}(x) = \mathbb{E}[\mathrm{HVI}(f(x))]^2. \tag{3.13}$$

Similarly to Equation (3.12), the constrained expected hypervolume improvement EHVI_C can be defined as

$$\mathrm{EHVI}_C(x) = \mathrm{HVI}(f(x))\mathrm{PF}(x). \tag{3.14}$$

That is the HV improvement weighted by the probability of it being feasible, $\mathrm{PF}(x)$ [33].

3.3.3 Overview of Bayesian optimization

Bayesian optimization is an iterative process, outlined as follows [24]:

1. **Initialization:** The algorithm starts by generating some initial point(s) and evaluating them with the objective function. The pair(s) of initial point(s) and their corresponding function value (evaluation) is stored in a dataset, $\mathcal{D}$.
2. **Initializing the surrogate model:** The surrogate model is initialized and fitted with $\mathcal{D}$.

[2]Arguments $\mathcal{P}$ and r removed from HVI for brevity.

3. **Optimizing the acquisition function:** The next point(s) to evaluate are chosen by optimizing the acquisition function, either maximizing or minimizing based on the objective.

4. **Evaluating the suggested point(s):** The newly chosen point(s) are evaluated. They are stored together with their function values in $\mathcal{D}$.

5. **Fitting the model:** The model is fitted with the updated $\mathcal{D}$. This optimizes the hyperparameters of the model to best represent the relationship between the inputs and the outputs.

6. **Stopping Criteria:** Steps 3-5, referred to as an iteration, are repeated until a predefined condition is met, typically a fixed number of iterations.

3.3.4 BoTorch

BoTorch [34] is a python library for BO research built on top of PyTorch. In this thesis, BoTorch is used for the BO implementations. This section will cover the relevant BoTorch specific implementations of Bayesian optimization components (models, acquisition functions etc) used in the BO implementations later introduced in Chapter 4.

As previously mentioned in Section 3.3, BO uses a surrogate function in the form of a model in order to approximate the actual objective function. In BoTorch, a model takes a set of data points as input and produces a posterior probability distribution of their outputs [35]. Here are some BoTorch models used in this thesis' BO implementation:

- **SingleTaskGP**: A single output GP regression model that models a single function.
- **ModelListGP**: A model that combines a list of single output gaussian process regression models such as *SingleTaskGPs* into a single model.

BoTorch also has its own implementations of acquisition functions that utilize Monte Carlo sampling, a more flexible and efficient way of optimizing acquisition functions [34]. Furthermore, for Monte Carlo EI acquisition functions, BoTorch provides Log variants that offer better BO performance [36].

Some of these acquisition functions include:

- **qLogNoisyExpectedImprovement**: Single-objective Monte Carlo acquisition function based on value improvement.
- **qLogNoisyExpectedHypervolumeImprovement**: Multi-objective Monte Carlo acquisition function based on hypervolume improvement.

Along with the acquisition functions, BoTorch provides methods for optimizing the acquisition function. A general one provided is:

- **optimize_acqf**: Generates a set of new candidate points via multi-start optimization of acquisition function. It takes in several hyperparameters including

num restarts, and *raw samples*. Increasing these hyperparameters increases optimization quality at the expense of computation time.

3.4 Hybrid optimization methods

A hybrid optimization method for MO problems combine complementary algorithms to leverage their strengths in exploration, exploitation, and computational efficiency. In this thesis, we are interested in approaches that combine BO's sample efficiency with GA's global search capabilities.

A straightforward hybrid approach involves sequential integration of Bayesian and genetic optimization as in [5], [6] described in Chapter 2. This is where the output of the BO approach is used as an initial population to the GA. The goal is to supply the GA with an already high-quality population, thus reducing random search inefficiencies.

Another approach to combine them is to make surrogate-assisted genetic algorithms, where the BO's probabilistic models are used to enhance the GA's operators like selection and mutation. In [37], Jin *et al.* proposed a Bayesian-genetic model that does this. More specifically, the proposed model uses the BO acquisition function to guide the GA selection process, ensuring that the genetic search is not solely stochastic but also influenced by the probabilistic predictions produced by BO. However, only relying on the acquisition function, in this case EI, for selection would lead to a decrease in diversity. Instead, they combine EI with the crowding distance of the candidate solutions. Additionally, this approach introduces an adaptive mutation operator that adjusts the mutation rates based on the uncertainty estimates provided by BO. Thus, in areas with higher uncertainty, we see higher mutation rates to encourage exploration. At the end of each iteration, all results found in the GA are fed back into the BO process to update the GP model. This flow allows for a continuous refinement of the understanding of the objective functions. The full procedure can be found in Algorithm 3 in Appendix A.3.

4

Methods

This chapter presents the details of the optimization approaches implemented in this thesis. We begin by describing the simulation framework and how the optimization techniques are integrated into it. Then, we outline both single-objective and multi-objective optimization methods, including GA and BO approaches. This is followed by a presentation of a hybrid approach to MO optimization that uses qualities from both the GA and the BO. Finally, we discuss how performance and quality of results is evaluated and compared across these methods.

The code implementation for all optimization methods presented in this thesis can be found in a GitHub repository[1]. All code related to the simulation, inputs and outputs is removed for confidentiality reasons.

4.1 Simulation setup & integration

The CES platform used in this work is a Simulink model, with MATLAB serving as the command interface. Inputs are sent to the Simulink model through MATLAB, and outputs are received in the same environment. By utilizing the Parallel Computing Toolbox [38], multiple simulations can be executed in parallel.

We developed a function that runs the simulations in parallel, extracts relevant outputs, and evaluates constraint violations that solutions must satisfy. This MATLAB function serves as the objective function for all optimization models introduced in the following sections. It takes inputs in the form of a set of solutions to evaluate, output/constraint bounds, and other simulation relevant input. The output of the function is the evaluated objective values, feasibility and constraint violations of each solution. The idea is that with minor modifications of expected input and output, this can be replaced with any other minimization function provided that the replacement adheres to the same input-output format, i.e. it accepts a batch of candidate solutions and returns objective values along with feasibility and constraint information.

[1] https://github.com/vichui1/optimization_algorithms.

4.2 Single-objective optimization models

To test our integration with the simulation model, we began by implementing single-objective optimization models. Both a simple GA and a BO model was implemented.

4.2.1 Single-objective genetic algorithm

We implemented a simple *Single-Objective Genetic Algorithm* (SOGA) and connected this to the simulation function in order to see how it would perform with regards to optimizing the simulation outputs. The outline of the SOGA is presented in Section 3.2.1 and Figure 3.3. We used TS with tournament size of two, single-point crossover and a mutation probability over all parameters. For the survival step we incorporated elitism by combining the parent population with the offspring population before selecting the N best individuals for next generation's population.

Here is a short walkthrough of the algorithm with a population size of N individuals:

1. **Initialization:** Our algorithm begins by generating an initial population using Sobol sampling [39], which produces quasi-random points uniformly distributed within the specified lower and upper bounds of each parameter. Each individual is represented as a vector of the input parameters. This is then followed by objective evaluation of each individual.

2. **Mating selection:** Next, N individuals are selected from the population using binary TS to form the mating pool. In our implementation, individuals can be selected multiple times, fully prioritizing fitter solutions. This approach increases the likelihood that better solutions contribute to the next generation.

3. **Variation**: The selected parents then undergo crossover to produce offspring by single-point crossover with a predefined probability. After that, each offspring undergo mutation with a separate mutation probability. The mutation operator is isolated for each parameter in the individual, which means that any parameter has the same chance to undergo variation. The resulting offspring population is then evaluated using the simulation function.

4. **Survival selection:** The previous population (before TS) is combined with the offspring to form a temporary population of size $2N$. The individuals are then sorted based on their objective fitness, and the top N individuals are selected to form the next generation. This completes one generation of the algorithm.

5. **Repeat** steps 2-4 for a predefined number of generations.

The main difference in the implementation compared to that outlined in Section 3.2.1 is how we follow output constraints that are not necessarily the objective function. For example, if we minimize the fuel consumption, we still have to make sure that the exhaust temperature is not too high. In order to adapt the algorithm to follow these constraints, we implemented a penalty method to penalize solutions that violate any constraint. The implementation of this was done using a hard penalty, which means

that a fixed value was added to the solution's fitness (objective value) as to deprioritize it in sorting and selection steps.

4.2.2 Single-objective Bayesian optimization algorithm

The *Single-Objective Bayesian Optimization Algorithm* (SOBO) is implemented in python using the BoTorch library [34] and follows the outline presented in Section 3.3.3. The following is a walkthrough of the implemented SOBO algorithm:

1. **Initialization:** Sobol sampling is used to choose a number of data points within the input bounds as the initialization inputs. These inputs are evaluated by the simulation platform (our objective function). The input points together with their evaluations are added to dataset $\mathcal{D}$.

2. **Initializing the model:** Two *SingleTaskGP* models are initialized and fitted with dataset $\mathcal{D}$ and are then combined into a single *ModelListGP*. The *SingleTaskGPs* model the objective function and feasibility function respectively. The feasibility function can be seen as a binary function that returns if a given input produces a feasible output or an infeasible output.

   The five parameters of each input point use different units and scales. To account for this, the input points are normalized to values between 0 and 1 before being passed into the model. Similarly, the outputs are standardized to have zero mean and unit variance, which the GP is trained on. The GP inverts the outputs back to their original scale before returning the data.

3. **Optimizing the acquisition function:** The model is then passed to an acquisition function so that it can be used in the optimization of the acquisition function, which in this implementation is *qLogNoisyExpectedImprovement*. This produces suggested points for the next evaluations. Evaluations will be done in batches, hence multiple points are produced. In addition we specify which output of the *ModelListGP* model corresponds to the objective and feasibility function respectively so that it handles constraints correctly.

4. **Evaluating the suggested points:** The suggested points given by optimizing the acquisition functions are evaluated by feeding them into the simulation function. Input and output pair is then added to dataset $\mathcal{D}$. This completes one iteration of the algorithm.

5. **Fitting the model:** The *ModelListGP* model is then fitted to the updated dataset $\mathcal{D}$.

6. **Repeat** steps 2-5 for a predefined number of iterations.

7. **Result:** Return the largest output value in $\mathcal{D}$.

Constraints are handled by the feasibility function mentioned in the second step. It guides the optimization loop towards producing feasible solutions. It is worth noting that BoTorch maximizes by default which means we need to negate the objective function values before adding them to dataset $\mathcal{D}$.

4.3 Multi-objective optimization models

After establishing a working framework for using single-objective optimization models, four multi-objective models were implemented: NSGA-II, NSGA-III, MOBO, and MOBOGA.

4.3.1 Custom stopping criteria

A custom stopping criteria was implemented for all subsequent MO models, with only minor configuration differences between the GA and BO implementations. The stopping criteria is designed to detect convergence by monitoring improvements in HV across generations or iterations. After each generation or iteration, the HV of the current Pareto-optimal set is computed and compared to that of a previous generation or iteration a number of steps back. The number of steps back is known as the window size. The improvement is calculated as a percentage, and if it falls below a predefined threshold, the run is terminated. Additionally, termination will not occur unless a minimum number of Pareto-optimal solutions have been identified, serving as a lower bound on the cardinality of the final solution set (see Section 3.1.3). This ensures the resulting Pareto set is sufficiently rich to support decision-making.

This approach approximates the algorithm's convergence by identifying when further improvements in both the quality and diversity of solutions become negligible. In order for this stopping criteria to work, a reference point is predefined. This reference point remains the same throughout all generations/iterations to ensure consistent calculation of HV improvement.

4.3.2 Multi-objective genetic algorithms

After observing the effect and convergence of the SOGA, we decided to move onto multi-objective versions of the genetic algorithms including the two NSGA versions NSGA-II and NSGA-III.

NSGA-II: The implementation of the NSGA-II follow the general outline of the original unconstrained description in [21] with added constraint-handling adaptations. Below is a short description of each step in the algorithm with a population size of N individuals:

1. **Initialization:** A random parent population P_0 is created using the same Sobol sampling method within the parameter ranges as in the SOGA. Then, it is evaluated using the simulation function to gather its objective outputs and any constraint violations. They are then sorted based on nondomination and so each individual is given a fitness, or rank. Then, the same binary TS, single-point crossover and mutation operator as in the SOGA, is used to create the offspring population Q_0 of size N.
2. **Elitist merging and sorting:** At generation t, a combined population $R_t = P_t \cup Q_t$ is formed followed by the nondominated sorting. Since all previous and current population members are included in R_t, elitism is ensured.

3. **Survival selection:** The next population P_{t+1} is selected in the same way as described in Section 3.2.2 using the crowding distance operator to prioritize diversity when choosing from the last fitting front.
4. **Offspring generation:** The new population P_{t+1} of size N is now used for selection, crossover, and mutation to create a new population Q_{t+1} of size N.
5. **Offspring evaluation:** The offspring is then evaluated. To cut down on time, parents that are copied as offspring, i.e., did not go through crossover or mutation, are not evaluated again as their fitness is already known[2].
6. **Repeat** steps 2-4 for a defined number of generations or until a stopping criteria is triggered.

Penalty methods prove inefficient for our constrained problem because a solutions fitness is determined by its Pareto dominance rank rather than directly from the objective values, as is typical in SO problems. Instead, we implemented the constraint-domination definition described in Section 3.2.2. This new domination scheme was integrated into the nondominated sorting algorithm as well as the TS. This implementation required normalization of all the constraint violations in order to compare the total violation between separate individuals.

In this implementation of the stopping criteria, after each generation, the HV of the current population is used to measure improvement, and the window size determines which previous generation to compare it to. Because each generation involves a large number of evaluations, the window size is kept relatively small to avoid letting the algorithm run too long without making progress.

NSGA-III: In order to account for the possibility of many-objective variations of engine calibrations, we then implemented the NSGA-III using the python library *pymoo* [40]. This implementation closely follows the description presented in Section 3.2.2 as well as the outline in [22]. The usage and structure can be found in pymoo's documentation [41]. For the reference directions we used Das and Dennis's approach that places points on a normalized hyperplane which is equally inclined to all objective axes and has an intercept of one on each axis [22], [42].

As mentioned in Section 3.2.2, the NSGA-III replaces the selection criterion in the survival step and TS when considering constrained problems. Our survival selection procedure follows the one presented in Algorithm 2 and the TS incorporates the constraint-domination scheme as described in Section 3.2.2. Additionally, binary TS and the same single-point crossover and uniform mutation operators were used as in the previous model implementations. The stopping criteria implementation is exactly the same as the one for NSGA-II.

4.3.3 Multi-objective Bayesian Optimization

The *Multi-objective Bayesian Optimization* (MOBO) algorithm is very similar to the SOBO algorithm. The main differences are the surrogate model as well as the

[2]The CES model is deterministic. If it were not, a new evaluation would still be necessary.

acquisition function used. Below is an outline of the algorithm:

1. **Initialization:** Sobol sampling is used in order to gather initial points that are within the input bounds. The initial points are evaluated and added to dataset $\mathcal{D}$ together with their evaluations.
2. **Initializing the model:** As in the SOBO algorithm, a *ModelListGP* is also used for the MOBO algorithm. It combines a list of *SingleTaskGPs*, one for each target objective and then an additional one for the feasibility function. In the same manner as in SOBO, inputs are normalized before being passed to the model and outputs are standardized in the model, but inverted to the original scale before being returned. The model is initialized and fitted with $\mathcal{D}$.
3. **Optimizing the acquisition function:** We have to use EHVI instead of EI since we are dealing with multiple objectives instead of a single objective.
4. **Evaluating the suggested points:** The suggested points given by optimizing the acquisition functions are evaluated by feeding them into the simulation function. The resulting input-output pair is added to dataset $\mathcal{D}$, marking the completion of one algorithm iteration.
5. **Fitting the model:** The *ModelListGP* is fitted to the updated dataset $\mathcal{D}$ in the same manner as in the SOBO algorithm.
6. **Repeat** steps 2-5 for a predefined number of generations or until stopping criteria is reached.
7. **Result:** Return the Pareto set from $\mathcal{D}$.

As with SOBO, constraint handling is modeled using a feasibility function.

4.3.4 Hybrid optimization model

To implement the hybrid algorithm model (MOBOGA), a sequential framework was developed that integrates the strengths of both MOBO and a GA, as described in Section 3.4. The framework initially runs the MOBO algorithm for a specified number of iterations, which produces a number of observations. Out of these observations, the first three fronts (after nondominated sorting) are chosen to be included in the initial population in the GA, either NSGA-II or NSGA-III. The decision to select the first three fronts helps balance the goal between obtaining high-quality solutions and ensuring a sufficiently large and diverse initial population. For instance, at certain KPs, BO may struggle to find feasible solutions, reducing its ability to identify many Pareto optimal solutions. Therefore, including solutions from the second and third fronts helps maintain population diversity and quality.

In order for this sequential approach to work, the GA implementations were modified to be able to take in an initial population along with their corresponding fitness and feasibility. If the size of the initial population is less than the desired population size, the missing number of individuals are generated in the same way as described

for when no initial population is given. If the given population exceeds the population size, the input is truncated to match the desired population size. Only the newly generated individuals are evaluated by the objective function to reduce the re-evaluation of the given initial population. The hybrid framework is illustrated in Figure 4.1, and the same stopping criteria as in both NSGA-II and NSGA-III is used.

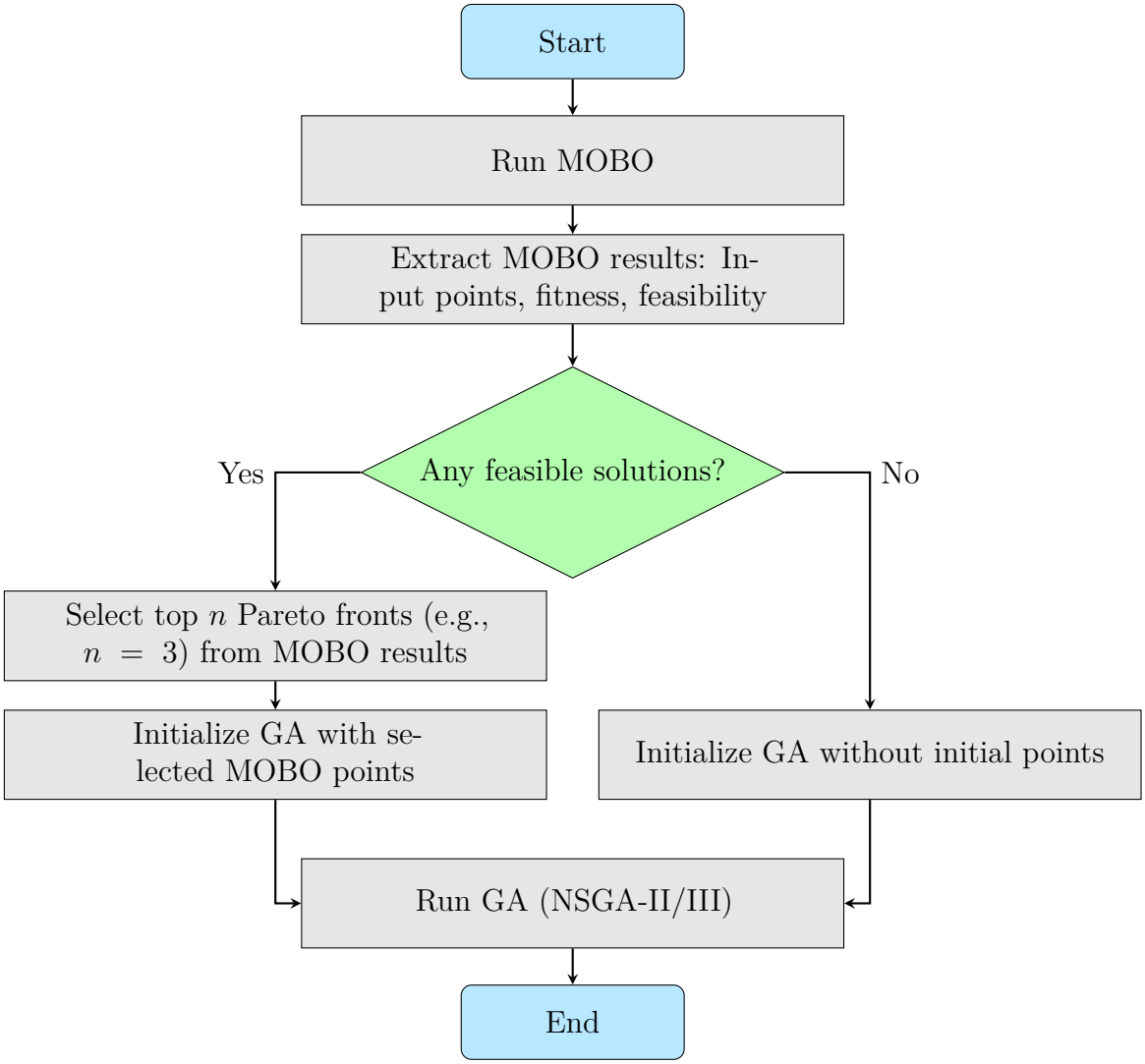


Figure 4.1: Sequential hybrid framework that uses the MOBO as a starting point to feed the GA.

4.4 Performance evaluation & comparison

To evaluate and compare the optimization methods, we use benchmark values derived from real engine test data provided by Volvo. This dataset contains parameter calibrations across multiple key operating points, and the benchmark values correspond to the minimum observed objective values. As this is our only frame of reference, it serves as the basis for all performance comparisons in the thesis.

For the SO problems, the optimal solution found by the SOGA and SOBO algorithms is compared directly against the benchmark values. Additionally, the parameter values of the optimal solutions are analyzed to asses their alignment with expected results. To ensure a fair comparison between the two methods, we allocate an equal number of evaluations to each. For instance, if the evaluation budget is n, the SOGA must satisfy `population size` $\times$ `number of generations` $= n$ [3], while the SOBO must satisfy `inits` $+$ `batch size` $\times$ `number of iterations` $= n$.

For the MO problems, the benchmark values are used to validate the solution ranges given by our NSGA-II/III, MOBO and MOBOGA implementations. The quality of these Pareto sets is assessed using the HV performance indicator. The indicator will be used to evaluate the average and standard deviation of HV across runs and between models, to assess the reliability and stability of the optimization algorithms. For these runs, we will use the stopping criteria described in Section 4.3.1 to assess both the number of evaluations and runtime each algorithm requires until no further improvements in quality and diversity are observed.

[3]The initial population is counted as one generation.

5

Results

In this chapter, we will present the results of the experiments done in this thesis. We begin in Section 5.1 by evaluating the performance of the two single-objective methods, focusing on their ability to minimize *Fuel Consumption* (FC) across one selected KP. In Section 5.2, we turn to the multi-objective methods and assess their performance in balancing multiple objectives, including FC, *NOx Emissions* (NE) and CO_2 *Concentration* (CC). In addition to presenting the results, we also define all hyperparameter values used for each model to provide context for the experimental setups. Given the notably poor scaling of the BO algorithms with an increasing number of objectives, we investigated the time required to optimize the acquisition function per iteration for all BO algorithms, as this was identified as the primary computational bottleneck. The results are presented in Appendix C.2.

For all subsequent experiments, each model was run multiple times to show what the performance looks like between several runs. This is particularly important for stochastic algorithms like GAs, which may not produce the same high-quality results in every execution. Therefore, average performance metrics are presented alongside standard deviations to provide a more robust comparison between benchmarks and between models.

Additionally, in accordance with our agreement with Volvo, the units for all objective values have been obfuscated.

5.1 Single-objective optimization results

For the SO problem, the objective FC was minimized. Both the SOGA and SOBO algorithms were executed 10 times for the KP with engine speed 1000 rpm and torque at 1400 Nm. The initial evaluation budget for these runs were 360 objective evaluations. To assess the sample efficiency of BO, two additional SOBO experiments were conducted using smaller evaluation budgets of 180 and 90 evaluations. The benchmark value is the current recorded minimum FC found for this KP. The hyperparameters used for both SOGA and SOBO can be seen in Table 5.1. For SOGA, since the first generation consists of a double-set of evaluations, initial population and their offspring, the number of evaluations adds up to 360. The results are shown in Table 5.2, displaying the mean best FC found, the standard deviation of the best FC in each run as well as the average runtime of the 10 optimization runs.

Evaluation budget	360
Hyperparameter	Value
population size	30
generations	11
tournament size	2
crossover rate	0.8
mutation rate	0.1

(a) SOGA hyperparameters.

Evaluation budget	360	180	90
Hyperparameter	Value		
batch size	8	8	8
iterations	44	21	10
num restarts[1]	10	10	10
raw samples[2]	512	512	512
inits	8	12	10

(b) SOBO hyperparameters.

Table 5.1: Hyperparameter settings for the two single-objective optimization methods.

	Benchmark	SOGA	SOBO	SOBO	SOBO
Mean best FC (N.A.)	7.29	7.1972	7.1789	**7.1785**	7.1849
Standard deviation (FC)	-	0.0110	**0.0036**	0.0041	0.0079
Mean runtime (minutes)	-	65.03	95.47	41.74	19.52
Evaluations	-	360	360	180	90

Table 5.2: Single-objective optimization results over 10 runs.

Figure 5.1 illustrates a SOGA run, showing both the minimum observed FC in each generation and the average FC across the entire population, highlighting the overall improvement over time. Due to the penalty implementation as described in Section 4.2.1, all observed infeasible solutions are discarded right away, ensuring that they are not present in the population and do not distort the observed average or best objective values. In Appendix B.1, the evolution of the parameter configurations are illustrated in two figures of the same run. The first showing the distribution of the initial population, highlighting the space-filling effect of Sobol sampling, and the second illustrates the parameter configurations in the last generation.

[1]Hyperparameter for optimizing the acquisition function. Refer to Section 3.3.4.
[2]Hyperparameter for optimizing the acquisition function. Refer to Section 3.3.4.

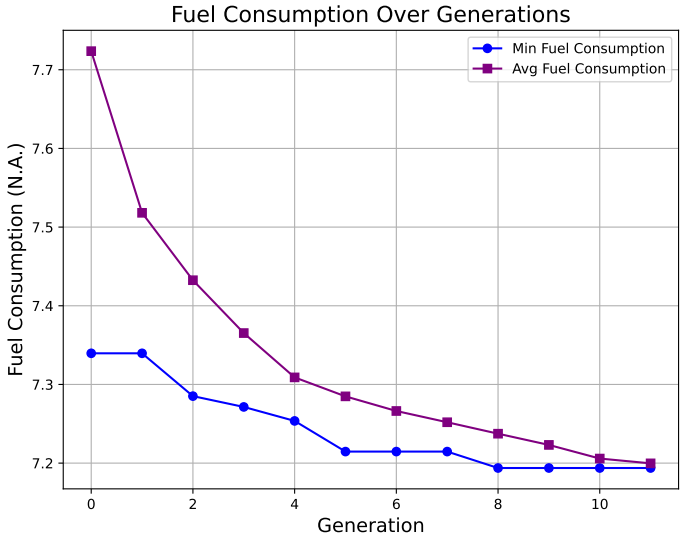


Figure 5.1: SOGA run minimizing fuel consumption at KP=(1000 rpm, 1400 Nm).

5.2 Multi-objective optimization results

For the multi-objective optimization, we define two different problem formulations: one with two objectives and one with three. For the bi-objective problem, FC and NE were minimized. For the tri-objective problem, CC was also minimized. To evaluate the generality and robustness of the approaches, runs were done on two KPs of the engine model:

- KP1 = (1000 rpm, 1400 Nm), representing a stable internal key point.
- KP2 = (1900 rpm, 2005 Nm), representing a high-torque condition.

These setups allow for assessing algorithm performance under varying engine conditions and complexities.

The benchmark values referred to in the following sections are previously identified minimum values of FC, NE, and CC for a specific KP. The benchmark values for the two KPs used in this experiments are presented in Table 5.3. The minimum CC is omitted for KP2 since it is not used for comparison in the tri-objective experiments.

	KP1	KP2
Min FC (N.A.)	7.29	21.31
Min NE (N.A.)	589.6	595.4
Min CC (N.A.)	5.74291	-

Table 5.3: Benchmarks values of the objectives for each key point used in the experiments.

As mentioned in Section 4.4, all MO experiments uses the stopping criteria outlined in Section 4.3.1, with varying window sizes depending on the model. The reference

point used for all algorithm runs in this section is:

- reference FC: 27.5,
- reference NE: 2640,
- reference CC: 15.4,

where each value is an approximation of the worst observed value (in all KPs) multiplied by a small factor, ensuring that solutions found throughout the optimization runs do not exceed this limit.

5.2.1 Bi-objective results

The experiments on the bi-objective problem formulation were conducted on each model, with each model being run five times using the hyperparameters defined in Table 5.4. The results gathered for KP1 are seen in Table 5.5. Similarly, the results for KP2 are shown in Table 5.6. The results presented in the table include the average minimum objective value across five model executions to enable comparison with the benchmark values. Additionally, the average HV achieved and its standard deviation are reported. Finally, the average runtime and number of evaluations are also included. For the MOBOGA experiments, the NSGA-II implementation was used for the GA step.

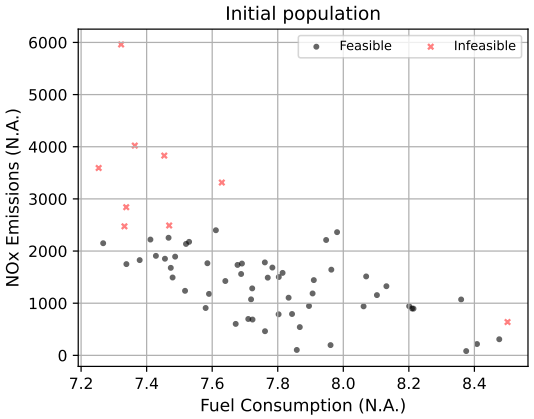
	NSGA-II	NSGA-III	MOBO	MOBOGA
Hyperparameter	Value			
population size	64	64	-	64
max generations	40	40	-	30
tournament size	2	2	-	2
crossover rate	0.8	0.8	-	0.8
mutation rate	0.1	0.1	-	0.1
inits	-	-	8	8
batch size	-	-	8	8
max iterations	-	-	80	10
num restarts	-	-	10	10
raw samples	-	-	512	512
window size	5	5	10	5

Table 5.4: Hyperparameters used for the bi-objective experiments on KP1 and KP2.

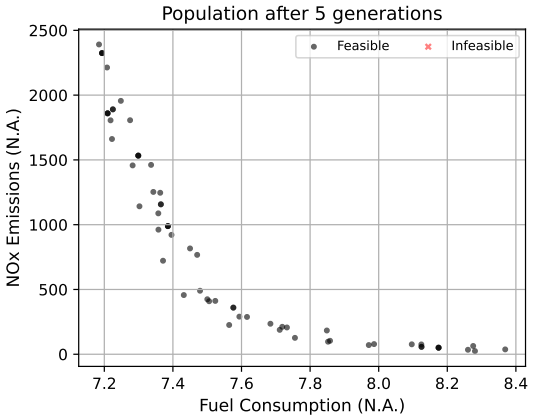
Figure 5.2 shows an example of a NSGA-II run of how the population evolves in the objective space from initialization to convergence for KP1.

In Appendix C.1, Figures C.1 and C.2 show Pareto fronts generated by each optimization method for KP1 and KP2 respectively. Each subfigure represents the run with the highest achieved HV over the five runs of that optimization model. The

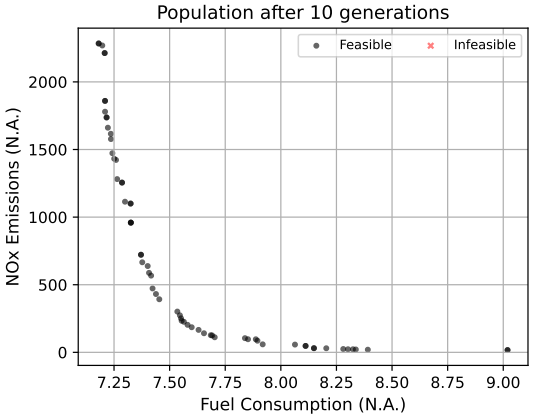
[3]One out of the five runs reached maximum number of generations, which was set at 40.
[4]Two out of the five runs reached maximum number of iterations, which was set at 80.
[5]Two out of the five runs reached maximum number of generations, which was set at 30.



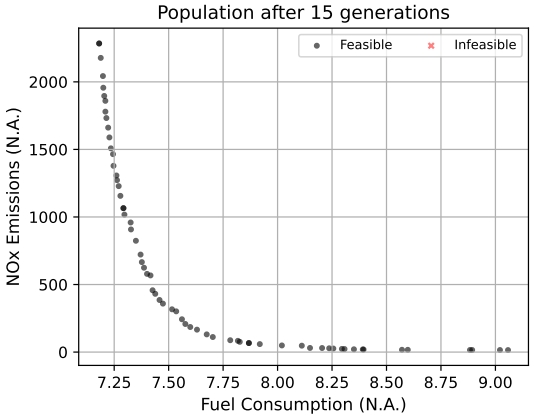
(a) Objective values of the initial population, generated by Sobol sampling.



(b) Objective values of the population after 5 generations.



(c) Objective values of the population after 10 generations.



(d) Objective values of the population in the final, in this case 15th generation.

Figure 5.2: Evolution of the population over generations in the NSGA-II run with the highest achieved hypervolume. Subfigures (a)−(d) show the progression of objective values from the initial population to the final generation. The population gradually moves toward better trade-offs between objectives, illustrating convergence and improved spread along the Pareto front.

	NSGA-II	NSGA-III	MOBO	MOBOGA
Mean Best FC (N.A.)	**7.18521**	7.19462	7.19013	7.20188
Mean Best NE (N.A.)	17.96735	23.66855	**11.472296**	14.3539
Mean HV	0.72860	0.72691	**0.73006**	0.729243
Standard deviation of HV	0.000686	0.002165	**0.000288**	0.00054
Mean runtime (minutes)	227.17	230	**77.31**	178.8
Mean number of evals	1036	1036.8	**233.6**	922.4

Table 5.5: Multi-objective optimization results over five runs for KP1 and two objectives.

	NSGA-II[3]	NSGA-III	MOBO[4]	MOBOGA[5]
Mean Best FC (N.A.)	20.242727	20.2867	**20.198954**	20.2672
Mean Best NE (N.A.)	78.1098	97.3741	**50.967812**	80.34315
Mean HV	0.249998	0.247014	**0.253025**	0.249462
Standard deviation of HV	0.00094	0.0021	**0.000137**	0.00123
Mean runtime (minutes)	293.65	291.6	**285.58**	306
Mean number of evals	1557	1331	**486.4**	1558.8

Table 5.6: Multi-objective optimization results over five runs for KP2 and two objectives.

corresponding parameter configurations of the five engine actuators, taken from the Pareto sets, are provided in Appendix B.3, with each figure showing one of the five runs for each optimization setup. Initial points evaluated in the first generation or iteration of the NSGA-II and MOBO algorithms are shown in Appendix B.2, specifically Figures B.3 and B.4, also for KP1. Additionally, the initial populations passed to the MOBOGA step are shown for both key points in Figures B.5 and B.6. These populations include both the clustered solutions from the MOBO phase and the additional Sobol-sampled points used to fill the population.

5.2.2 Tri-objective results

In line with the bi-objective experiments, the tri-objective formulation was evaluated across all models, with each run conducted five times. Note that the experiments were only conducted on KP1, as the main interest was observing the scalability of each model for higher objective problems. The hyperparameters used in each model are detailed in Table 5.7 and the corresponding results are presented in Table 5.8. The structure of the results gathered is the same as for the bi-objective experiments. As before, the NSGA-II was used for the GA step in the MOBOGA.

	NSGA-II	NSGA-III	MOBO	MOBOGA
Hyperparameter	Value			
population size	80	80	-	80
max generations	30	30	-	30
tournament size	2	2	-	2
crossover rate	0.8	0.8	-	0.8
mutation rate	0.1	0.1	-	0.1
inits	-	-	8	8
batch size	-	-	8	8
max iterations	-	-	40	10
num restarts	-	-	10	10
raw samples	-	-	512	512
window size	5	5	10	5

Table 5.7: Hyperparameters used for the tri-objective experiments on KP1.

	NSGA-II	NSGA-III	MOBO[6]	MOBOGA
Mean Best FC (N.A.)	7.20448	7.2076065	**7.194636**	7.20319
Mean Best NE (N.A.)	19.17228	41.193549	**15.407844**	17.87998
Mean Best CC (N.A.)	5.63534	5.637608	**5.383833**	5.4248
Mean HV	0.4342143	0.4375	**0.453283**	0.4484
Standard deviation of HV	0.00662	0.00275	**0.00093**	0.003979
Mean runtime (minutes)	226.8	234.7	306.9	**173.9**
Mean number of evals	1019.4	1248	**328**	900.2

Table 5.8: Multi-objective optimization results over 5 runs for KP1 and three objectives.

Examples of the parameter configurations of the resulting Pareto optimal solutions are displayed in Appendix B.4, where each figure shows one of the algorithms.

[6]All five runs reached the maximum number of iterations, which was set at 40.

6

Discussion

This chapter presents a discussion of the optimization model results and the methodological choices made throughout this project. We begin by detailing the outcomes of the SO experiments, followed by the MO experiments. In addition to assessing model performance, we investigate the convergence range of Pareto optimal solutions achieved by each method.

The discussion then shifts to the algorithm configurations, where we elaborate on choices regarding initialization, stopping criteria, and aspects unique to the GA, BO, and hybrid implementations. The chapter concludes by outlining potential avenues for future research, focusing on both enhancements to the current implementations and extensions to the problem's scope.

6.1 Discussion of results

This section is dedicated to discussion of results presented in Chapter 5. We begin by discussing the SO results, followed by the MO results.

6.1.1 Single-objective results

From the results observed in Section 5.1, both SOGA and SOBO consistently achieved better minimum FC compared to the benchmark value. For KP1, SOBO consistently demonstrated superior minimum FC, even with a reduced number of evaluations. An interesting observation is that SOBO results from the 180-evaluation runs were marginally better than those from 360 evaluations. This suggests that minimal improvement is gained by increasing evaluations from 180 to 360. Within this range, variance likely plays a more significant role than the number of evaluations in determining the results. This effect is even more pronounced for SOGA's 360-evaluation runs, which exhibited a larger standard deviation compared to SOBO. This indicates that SOGA is more susceptible to variance, aligning with its inherent stochasticity. When the number of SOBO evaluations was reduced to 90, we observed slightly worse results compared to the 180 and 360 evaluation runs, suggesting that reducing evaluations below 180 does negatively impact performance.

A reduction in evaluations, while leading to a slightly higher minimum FC, provides the benefit of lower computational runtime. Therefore, determining the appropriate number of evaluations involves a trade-off between prioritizing computational

runtime and achieving acceptable minimum FC.

For the 360-evaluation runs of SOGA and SOBO, SOBO exhibited an average runtime approximately 30 minutes longer. This suggests an inherent overhead in each SOBO iteration absent in SOGA. As indicated by acquisition time data showcased in Appendix C.2, this overhead appears to stem primarily from the optimization of the acquisition function. In contrast, the offspring generation and selection steps in the GA are computationally less demanding, resulting in lower overhead.

6.1.2 Multi-objective results

This section discusses the algorithms' performance when solving multi-objective problems. We first examine the bi-objective case, then extend the analysis to the tri-objective formulation.

Bi-objective results: Mirroring the trends observed in the SO results, all algorithms surpassed the benchmarks in simultaneously minimizing both FC and NE within the bi-objective problem. The BO-based method, MOBO, achieved a higher HV compared to both GA-based implementations. This superior performance is likely attributable to BOs inherent ability to more effectively balance exploration and exploitation than GAs. The overhead associated with MOBO becomes even more pronounced in MO optimization, as its number of evaluations is significantly lower than both GA methods, even when accounting for overall runtime. Regarding the GA implementations, NSGA-II outperformed NSGA-III, as expected in a low-dimensional objective space.

The algorithms' performance varied across the KPs. For KP2, all algorithms required more iterations/generations and longer runtime to reach their stopping criteria compared to KP1, with MOBO notably reaching its maximum iterations in two out of five runs. This increased difficulty is attributed to the high-torque operating condition at KP2, which introduces greater instability and thus increases the complexity of finding feasible solutions.

Interestingly, the increase in evaluations was most significant for MOBO, which more than doubled its evaluations for KP2 compared to KP1. However, the runtime did not increase proportionally, instead, it more than tripled. This discrepancy can be attributed to the iterative growth in the computational cost of optimizing the acquisition function within each iteration. This suggests that our MOBO implementation may encounter issues when applied to problems requiring a high number of evaluations to meet the stopping criteria. This runtime challenge is further exacerbated when an additional model (objective or constraint) is incorporated into the *ModelListGP*, as will be discussed in the tri-objective results.

Despite these challenges for KP2, MOBO still achieved a superior HV compared to all other implementations, likely due to its inherent strength in effectively balancing the trade-off between exploration and exploitation. NSGA-II continued to outperform NSGA-III, as anticipated in low-dimensional objective spaces.

As for the hybrid MOBOGA, it outperformed both GA implementations in all regards for KP1, while only surpassing NSGA-III for KP2. In the latter case, NSGA-III converged faster at the cost of achieving a worse HV. In both scenarios, the HV achieved by MOBOGA surpassed both GAs, closely approaching the HV attained by MOBO. This indicates that the hybrid combination successfully enhanced the GAs convergence rate, although for this particular problem, the standalone MOBO outperformed MOBOGA in all aspects.

Figures B.5 and B.6 in Appendix B.2 illustrate the effect of the sequential hybrid approach. The size of the initial population generated by MOBO varies depending on the selected KP. For instance, at KP1, the clustered initial population is relatively large, which means the GA relies less on Sobol sampling for exploration. In contrast, at KP2, the initial population is smaller, causing MOBOGA to behave more like a traditional GA with a greater dependence on exploration through sampling. This observation is supported by the results in Table 5.6, where MOBOGA and NSGA-II achieve similar HV values, runtime, and number of evaluations. The key takeaway is that MOBOGA's efficiency is influenced by the KP selected. When more data is available upfront to the GA, the algorithm becomes more exploitation-driven and efficient. Conversely, with less initial data, the algorithm leans more toward exploration, resembling a standard GA.

As summarized in Figures C.1 and C.2 of Appendix C.1, all four described methods generated solution sets that cover broad regions of the Pareto fronts for the bi-objective formulation. Notably, MOBO consistently extended the range of coverage in both objective directions, indicating a strong ability to explore the trade-off space more thoroughly than the other methods.

Tri-objective results: When extending from two objectives to three, we first observe that NSGA-III achieved a higher HV than NSGA-II, with only a modest increase in evaluations and runtime. The improved HV performance of NSGA-III can be attributed to its design, which is specifically tailored for many-objective problems. MOBO continues to outperform the other algorithms. However, with three objectives, its runtime increases significantly compared to the bi-objective case, demonstrating poor scalability. This is further supported by observing the increase in acquisition function optimization time from bi to tri objectives, shown in Appendix C.2. Unlike with constraints, objectives cannot be merged, limiting the scalability of MOBO. Therefore it is not recommended to use *ModelListGP* as the surrogate model when handling many-objective problems.

Examining the results for MOBOGA, it achieved a higher HV than both GAs while also requiring fewer evaluations and achieving faster runtimes. As MOBO slows down in later iterations, MOBOGA effectively circumvents this issue, which explains its overall efficiency. Since MOBO generally generates high-quality solutions in relatively few iterations (in this problem setting), MOBOGA avoids the early-stage exploration of low-importance regions typically seen in GA runs. Although it did not quite match the HV performance of MOBO, MOBOGA demonstrates strong potential for both bi- and tri-objective optimization.

Parameter configuration results: Despite achieving slightly differing HVs, all optimization methods converged to similar parameter ranges, as shown in Appendix B.3 and B.4. This was especially evident for the bi-objective problem, providing valuable insight into the optimal ranges of each parameter.

6.2 Algorithm configurations

Given that this project focused on providing a proof-of-concept for the optimization models' feasibility in engine calibration, many hyperparameter decisions and algorithm configurations were not explored as thoroughly as they could have been. This includes investigating alternative hyperparameter values and diverse implementations and configurations of the algorithms themselves.

6.2.1 Initialization

For both the GA and BO approaches, the initial population or batch was generated using Sobol sampling. The decision to generate these without relying on heuristics or expert knowledge was based on the goal of ensuring these optimization models could perform effectively on novel engine models or operating points. While leveraging previously identified "optimal" calibrations could potentially accelerate optimization runs, it also risks introducing bias that might limit the algorithms' ability to discover superior calibrations.

Appendix B.2 displays the parameter initializations for some of the methods. Figures B.3 and B.4 show the coverage of the parameter space using Sobol sampling. A notable difference is that MOBO initializes with substantially less spatial coverage compared to NSGA-II, which highlights its sample efficiency, as reflected by the higher HV achieved in the bi-objective problems.

6.2.2 Stopping criteria

Since our primary focus was on solution quality, we implemented a stopping criterion based on HV improvement. HV calculations necessitated a fixed reference point to enable meaningful comparison of HV values across generations or iterations. We opted for a static reference point, defined by the upper bounds on the objectives across all KPs, thereby eliminating the need for a separate reference point for each KP. This approach effectively balances practicality with the preservation of high solution quality.

6.2.3 Genetic algorithms

In both the SOGA and MO variants of the genetic algorithms, we retained common components such as elitism, selection, and variation operators. These choices were made based on their suitability for the specific problem and their compatibility with the various optimization goals.

Elitism was incorporated because any high-quality solution found throughout the optimization run is valuable for the final solution set, mitigating the risk of its removal by variation operators. This is crucial given that varying KPs introduce different levels of constraints, which can limit the probability of re-discovering these high-quality solutions.

One-point crossover is a simple yet effective recombination method that encourages the GA to explore the search space. While two-point or k-point crossover could be implemented, further exploration in this area was not a primary focus for this thesis. A crossover probability of 0.8 was consistently used across all GA implementations, aligning with common starting points typically falling between $0.7 - 0.8$ [43]. Similarly, uniform mutation (see Section 3.2.1) was chosen as the mutation operator for all implementations, with its probability set at 0.1. This value, on the higher end of common ranges, was selected because we prioritize exploration, given the incorporation of elitism and the feeding of previously found solutions into the survival selection step.

The population size used during result gathering was limited due to computational capacity and memory constraints. Consequently, for future work or when implementing these methods, a larger population size is more ideal, particularly for MO cases. This is because MO problems necessitate greater diversity as the number of objectives increases and the theoretical Pareto front expands accordingly. For NSGA-III, Das and Dennis's approach was chosen for generating the reference sets (see Section 3.2.2), consistent with the methodology outlined in [22]. The paper also states that any predefined structured placement of reference points can be utilized.

6.2.4 Bayesian Optimization

Unlike GAs, there are no particular benefits to having a larger batch size for BO other than reducing the time required to complete a given number of evaluations. Thus, the batch size was set to the maximum number of evaluations that could be run in parallel.

For constraint handling, we opted for a single feasibility function encompassing all constraints to avoid unmanageable runtime. While an alternative approach would involve modeling each constraint separately using its own *SingleTaskGP* model, each additional model results in increased runtime. As observed in Appendices C.3, C.4 and C.5, this increase in runtime can be attributed to the optimization of the acquisition function, which appears to grow exponentially. Theoretically, handling each constraint separately should increase the number of feasible candidates generated. Exploring whether this benefit justifies the computational cost could be an interesting area for future work, but it falls beyond the scope of this thesis.

In a similar manner to constraint handling, multiple objectives are managed by modeling each objective as a separate *SingleTaskGP*. As noted previously, each additional model increases runtime, and this growth becomes more significant with more iterations. Therefore, an alternative model choice might be more suitable when working with many-objective problems.

The hyperparameters for the optimization of the acquisition function were chosen based on the values presented in the BoTorch documentation [44]. No attempts were made to tune these parameters in this thesis, although it presents an intriguing avenue for future exploration. As mentioned in Section 3.3.4, reducing *num restarts* and *raw samples* could lead to faster candidate generation, likely at the cost of reduced candidate quality. Identifying the optimal balance between time efficiency and candidate quality could be particularly beneficial in real-world use cases.

6.2.5 Hybrid model

For the hybrid MOBOGA implementation, we adopted a sequential approach. Further attempts to introduce surrogate-based GA operators, such as selection and mutation, are reserved for future research. Nevertheless, this sequential model effectively combines the strengths of each approach, demonstrating that the two present an interesting combination for engine calibration.

The decision to feed the first three non-dominated fronts generated by BO into the GA was an approximate attempt to ensure the GA received a substantial number of solutions. This is particularly relevant when shifting KPs, such as from a stable internal one to a high-torque one. In such cases, the feasible parameter space is more restricted, which minimizes the number of feasible candidate solutions found within a limited time.

The number of iterations for the BO runs was consistently set to 10 across all experiments. This decision stems from observations during development, where MOBO typically required 8-10 iterations to generate a substantial set of high-quality solutions for all KPs. Furthermore, the overhead incurred when increasing the number of objectives was less significant at 10 iterations compared to later iterations.

6.3 Future Work

There are numerous ways to build on and enhance the current work, both in terms of model improvements and extending their application.

6.3.1 Model improvements

Regarding model improvements, several avenues exist to enhance the current models. One direction involves further exploring various parameters and configurations, as discussed in Section 6.2. This could include investigating population sizes, mutation rates, batch sizes, or even entirely alternative strategies, such as different mutation operators or acquisition functions. Exploring these choices could yield improved performance on the same problem formulations used in this thesis.

Another identified bottleneck of the GA models is their slow convergence rate, partly attributable to the global exploration performed by the uniform mutation operator. Introducing a surrogate-based mutation operator, inspired by BO, could focus search

efforts on more promising regions of the search space and significantly reduce computational time. This approach would be valuable to employ in both NSGA-II/III and MOBOGA, further enhancing its hybrid nature. However, sufficient global exploration must remain possible to avoid converging to local optima.

Thus far, the optimization models have only been applied to bi- and tri-objective problems. Future scenarios may involve more objectives, which would increase optimization complexity and likely require adaptations to maintain model performance. For instance, NSGA-IIs performance would deteriorate (see Section 3.2.2), whereas NSGA-III is better suited for many-objective problems and would therefore perform more effectively. Additionally, as observed from the MOBO results, increasing the number of objectives significantly raises its time complexity due to the use of a separate *SingleTaskGP* model for each objective. Therefore, exploring new MOBO configurations would be necessary.

Throughout the experiments presented in this thesis, solution quality was prioritized, and thus, the time efficiency of the optimizations was overlooked. This choice depends on the specific use case and represents an avenue for further exploration. For example, future work could involve evaluating the results by allowing each algorithm a certain time budget.

6.3.2 Extending the scope

An important next step involves extending these models from engine calibration to new use cases such as full vehicle calibration using a GSP[1] simulation platform. This would enable optimizing the engine within a more realistic operational context. This step would also test the models' generality and scalability, as it entails a complete exchange of the objective function.

In this thesis, each calibration was performed at isolated KPs. A future goal is to generate a complete engine map by interpolating between KPs, which requires smoothing techniques to ensure consistent transitions across the operating range. In addition, analyzing the resulting parameter sets could provide valuable insights beyond the optimality of solutions. For example, identifying parameter importance or sensitivity could guide future optimization runs and aid in model simplification when dealing with numerous parameters. If the application is later extended to more complex simulations, such as the GSP, parameter reduction techniques may become essential for these models to remain functional.

[1] Global simulation platform

7

Conclusion

This thesis presented a framework for Multi-Objective (MO) optimization of engine calibration parameters at individual operating points. The work began with Single-Objective (SO) optimization models and progressed into a full MO framework. Both metaheuristic and probabilistic methods were implemented and compared, these being Genetic Algorithms (GAs) and Bayesian Optimization (BO), as well as a hybrid model combining the elements of both.

All models demonstrated their suitability for engine calibration in both SO and MO contexts by outperforming the objective benchmarks. However, their effectiveness varies depending on the specific use case and optimization priorities. For generating a wide spread of the Pareto fronts, Multi-Objective Bayesian Optimization (MOBO) proved to yield superior results with fewer evaluations, highlighting its sample efficiency. However, when moving to higher objective-spaces, the proposed implementation shows poor scalability. In contrast, the GA-based implementations, NSGA-II and NSGA-II, and the hybrid model (MOBOGA) scaled appropriately.

The MOBOGA showed potential, especially in higher-dimensional problems, but requires further tuning to fully leverage the best qualities of GAs and BO. Overall, the developed framework lays a strong foundation for future work, including efforts to improve hybridization strategies and adapt the approach for more complex engine calibration tasks.

Bibliography

[1] S. Mirjalili and S. Mirjalili, "Genetic algorithm," *Evolutionary algorithms and neural networks: Theory and applications*, pp. 43–55, 2019.

[2] R. Garnett, *Bayesian optimization*. Cambridge University Press, 2023.

[3] L. Zhu, Y. Wang, A. Pal, and G. Zhu, "Engine calibration using global optimization methods with customization," SAE Technical Paper, Tech. Rep., 2020.

[4] X. Yu, L. Zhu, Y. Wang, D. Filev, and X. Yao, "Internal combustion engine calibration using optimization algorithms," *Applied Energy*, vol. 305, p. 117 894, 2022.

[5] M. Liu, P. Zhuang, and F. Lai, "A bayesian optimization-genetic algorithm-based approach for automatic parameter calibration of soil models: Application to clay and sand model," *Computers and Geotechnics*, vol. 176, p. 106 717, 2024.

[6] P. Suryawanshi, S. Gaikwad, A. Kumar, A. Patlolla, and S. K. Jayakumar, "A hybrid bayesian-genetic algorithm based hyperparameter optimization of a lstm network for demand forecasting of retail products.," in *IJCCI*, 2023, pp. 230–237.

[7] E. K. Chong and S. H. ak, *An introduction to optimization*. John Wiley & Sons, 2013, vol. 76.

[8] A. Homaifar, C. X. Qi, and S. H. Lai, "Constrained optimization via genetic algorithms," *Simulation*, vol. 62, no. 4, pp. 242–253, 1994.

[9] K. Deb *et al.*, "Evolutionary algorithms for multi-criterion optimization in engineering design," *Evolutionary algorithms in engineering and computer science*, vol. 2, pp. 135–161, 1999.

[10] P. Ngatchou, A. Zarei, and A. El-Sharkawi, "Pareto multi objective optimization," in *Proceedings of the 13th international conference on, intelligent systems application to power systems*, IEEE, 2005, pp. 84–91.

[11] V. Noghin, "Linear scalarization in multi-criterion optimization," *Scientific and Technical Information Processing*, vol. 42, pp. 463–469, 2015.

[12] C. von Lücken, B. Barán, and C. Brizuela, "A survey on multi-objective evolutionary algorithms for many-objective problems," en, *Computational Optimization and Applications*, vol. 58, no. 3, pp. 707–756, Jul. 2014, ISSN: 1573-2894. DOI: 10.1007/s10589-014-9644-1. [Online]. Available: https://doi.org/10.1007/s10589-014-9644-1 (visited on 03/19/2025).

[13] N. Riquelme, C. Von Lücken, and B. Baran, "Performance metrics in multi-objective optimization," in *2015 Latin American computing conference (CLEI)*, IEEE, 2015, pp. 1–11.

[14] C. M. Fonseca, L. Paquete, and M. López-Ibánez, "An improved dimension-sweep algorithm for the hypervolume indicator," in *2006 IEEE international conference on evolutionary computation*, IEEE, 2006, pp. 1157–1163.

[15] *Lebesgue measure*, en, Page Version ID: 1280128918, Mar. 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Lebesgue_measure&oldid=1280128918 (visited on 03/28/2025).

[16] I. Bajaj, A. Arora, and M. M. F. Hasan, "Black-Box Optimization: Methods and Applications," en, in *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*, P. M. Pardalos, V. Rasskazova, and M. N. Vrahatis, Eds., Cham: Springer International Publishing, 2021, pp. 35–37, ISBN: 9783030665159. DOI: 10.1007/978-3-030-66515-9_2. [Online]. Available: https://doi.org/10.1007/978-3-030-66515-9_2 (visited on 03/18/2025).

[17] X. Yu and M. Gen, *Introduction to Evolutionary Algorithms* (Decision Engineering), 1st ed., R. Roy, Ed. London: Springer London, 2010, vol. 0, ISBN: 9781849961288 9781849961295. DOI: 10.1007/978-1-84996-129-5. [Online]. Available: http://link.springer.com/10.1007/978-1-84996-129-5 (visited on 03/05/2025).

[18] *Metaheuristic*, en, Page Version ID: 1278583986, Mar. 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Metaheuristic&oldid=1278583986 (visited on 03/05/2025).

[19] B. L. Miller, D. E. Goldberg, *et al.*, "Genetic algorithms, tournament selection, and the effects of noise," *Complex systems*, vol. 9, no. 3, pp. 193–212, 1995.

[20] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability engineering & system safety*, vol. 91, no. 9, pp. 992–1007, 2006.

[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[22] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014. DOI: 10.1109/TEVC.2013.2281535.

[23] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach," *IEEE Transactions on evolutionary computation*, vol. 18, no. 4, pp. 602–622, 2013.

[24] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[25] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, vol. 25, 2012.

[26] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, pp. 455–492, 1998.

[27] P. I. Frazier and J. Wang, "Bayesian optimization for materials design," *Information science for materials discovery and design*, pp. 45–75, 2016.

[28] *Bayes' theorem*, en, Page Version ID: 1279989781, Mar. 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Bayes%27_theorem&oldid=1279989781 (visited on 03/18/2025).

[29] E. Schulz, M. Speekenbrink, and A. Krause, "A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions," *Journal of mathematical psychology*, vol. 85, pp. 1–16, 2018.

[30] *Arg max*, en, Page Version ID: 1225875746, May 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Arg_max&oldid=1225875746 (visited on 05/21/2025).

[31] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham, "Bayesian optimization with inequality constraints.," in *ICML*, vol. 2014, 2014, pp. 937–945.

[32] B. Letham, B. Karrer, G. Ottoni, and E. Bakshy, "Constrained bayesian optimization with noisy experiments," 2019.

[33] S. Daulton, M. Balandat, and E. Bakshy, "Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9851–9864, 2020.

[34] M. Balandat, B. Karrer, D. Jiang, *et al.*, "Botorch: A framework for efficient monte-carlo bayesian optimization," *Advances in neural information processing systems*, vol. 33, pp. 21 524–21 538, 2020.

[35] *Models | BoTorch*, en, Feb. 2025. [Online]. Available: https://botorch.org/docs/models (visited on 04/01/2025).

[36] S. Ament, S. Daulton, D. Eriksson, M. Balandat, and E. Bakshy, "Unexpected improvements to expected improvement for bayesian optimization," *Advances in Neural Information Processing Systems*, vol. 36, pp. 20 577–20 612, 2023.

[37] J. Jin, Y. Xu, J. He, *et al.*, "A hybrid bayesian-genetic approach for micro gear tolerance optimization," in *2024 29th International Conference on Automation and Computing (ICAC)*, 2024, pp. 1–6. DOI: 10.1109/ICAC61394.2024.10718775.

[38] MathWorks, *Parallel computing toolbox*, Version R2024a, The MathWorks, Inc., 2024. [Online]. Available: https://www.mathworks.com/products/parallel-computing.html.

[39] *Sobol sequence*, en, Page Version ID: 1293802726, Jun. 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Sobol_sequence&oldid=1293802726 (visited on 06/09/2025).

[40] J. Blank and K. Deb, "Pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.

[41] *Pymoo - NSGA-III*. [Online]. Available: https://pymoo.org/algorithms/moo/nsga3.html (visited on 03/27/2025).

[42] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems," *SIAM journal on optimization*, vol. 8, no. 3, pp. 631–657, 1998.

[43] J. Brownlee, *Chapter 5 - Crossover and Its Effects*, en-us. [Online]. Available: https://algorithmafternoon.com/books/genetic_algorithm/chapter05/ (visited on 05/21/2025).

[44] *Multi-objective optimization with qEHVI, qNEHVI, and qNParEGO | BoTorch*, en, Feb. 2025. [Online]. Available: https://botorch.org/docs/tutorials/multi_objective_bo/ (visited on 04/02/2025).

A

Algorithms

A.1 Fast Nondominated sorting algorithm

Algorithm 1 Fast Nondominated Sorting Algorithm [21]

Require: A population P.

Ensure: A set of fronts $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k\}$ such that individuals in $\mathcal{F}_1$ are nondominated by any other individual, and so on.

```
1:  for each p ∈ P
2:     S_p = ∅
3:     n_p = 0
4:     for each q ∈ P
5:        if (p ≻ q) then                          ▷ if p dominates q
6:           S_p = S_p ∪ {q}                       ▷ Add q to the set of solutions dominated by p
7:        else if (q ≻ p) then
8:           n_p = n_p + 1                         ▷ Increment the domination counter of p
9:     if n_p = 0 then                             ▷ p belongs to the first front
10:       p_rank = 1
11:       F_1 = F_1 ∪ {p}
12: i = 1                                          ▷ Initialize the front counter
13: while F_i ≠ ∅
14:    Q = ∅                                       ▷ Used to store the members of the next front
15:    for each p ∈ F_i
16:       for each q ∈ S_p
17:          n_q = n_q − 1                         ▷ q belongs to the next front
18:          if n_q = 0 then
19:             q_rank = i + 1
20:             Q = Q ∪ {q}
21:    i = i + 1
22:    F_i = Q
```

A.2 Survival selection in NSGA-III

Algorithm 2 Survival selection in NSGA-III [22]

Require: A combined population R_t of size $2N$, H structure reference points Z^s, Fronts $(\mathcal{F}_1, \mathcal{F}_2, ...)$

Ensure: Next population P_{t+1}

1: $S_t = \emptyset, i = 1$
2: **repeat**
3: $S_t = S_t \cup F_i$ and $i = i + 1$
4: **until** $|S_t| \geq N$
5: Last front to be included: $F_l = F_i$
6: **if** $|S_t| = N$ **then**
7: $P_{t+1} = S_t$, break
8: **else**
9: $P_{t+1} = \cup_{j=1}^{l-1} F_j$
10: Points to be chosen from $F_l : K = N - |P_{t+1}|$
11: Normalize objectives and create reference set: $\mathbf{f}^n, Z^r = \texttt{Normalize}(S_t, Z^s)$
12: Associate each member s of S_t with a reference point: $\pi(\mathbf{s}), d(\mathbf{s}) = \texttt{Associate}(S_t, Z^r) \triangleright \pi(\mathbf{s})$: closest reference point, $d(\mathbf{s})$: distance between $\mathbf{s}$ and $\pi(\mathbf{s})$
13: Compute niche count of reference point $j \in Z^r$: $\rho_j = \sum_{\mathbf{s} \in S_t / F_l} ((\pi(\mathbf{s}) = j) \; ? \; 1 : 0)$
14: Choose K members one at a time from F_l to construct $P_{t+1} = \texttt{Niching}(K, \rho_j, \pi, d, Z^r, F_l, P_{t+1})$
15: **end if**

A.3 Bayesian and Genetic Hybrid algorithm

Algorithm 3 Hybrid Optimization for Micro Gear Tolerance [37]

Require: Random Sampling particle, probabilistic model M, population size P, Crossover probability $CXPB$, mutation probability $MUTPB$, Tournament size, Mutation operator(μ, σ, indpb), search space.

Ensure: Best solution found.

repeat
  BO: for *a set of number of iterations* **do**
    Select next sample x_{BO} by maximizing EI acquisition function α_{EI} of M, considering diversity maintenance (crowding distance).
    Evaluate objective function f at x_{BO} to obtain y_{BO}
    Update model M with (x_{BO}, y_{BO}).
  GA: for *a set of number of generations* **do**
    Evaluate fitness of all individuals in P using M
    Select parents using EI, considering diversity maintenance (crowding distance)
    Generate offspring via crossover and mutations
    Update P with new offspring
  Feedback:
    Apply smoothing technique to the best solutions from GA (x_{GA}, y_{GA}) to obtain smoothed feedback solutions $(x_{GA}^{smoothed}, y_{GA}^{smoothed})$.
    Update BO dataset: $\mathcal{D}_{new} = \mathcal{D} \cup \{x_{GA}^{smoothed}, y_{GA}^{smoothed}\}$
    Update model M with new observations from GA
until converged

B

Parameter configurations

B.1 SOGA parameter evolution

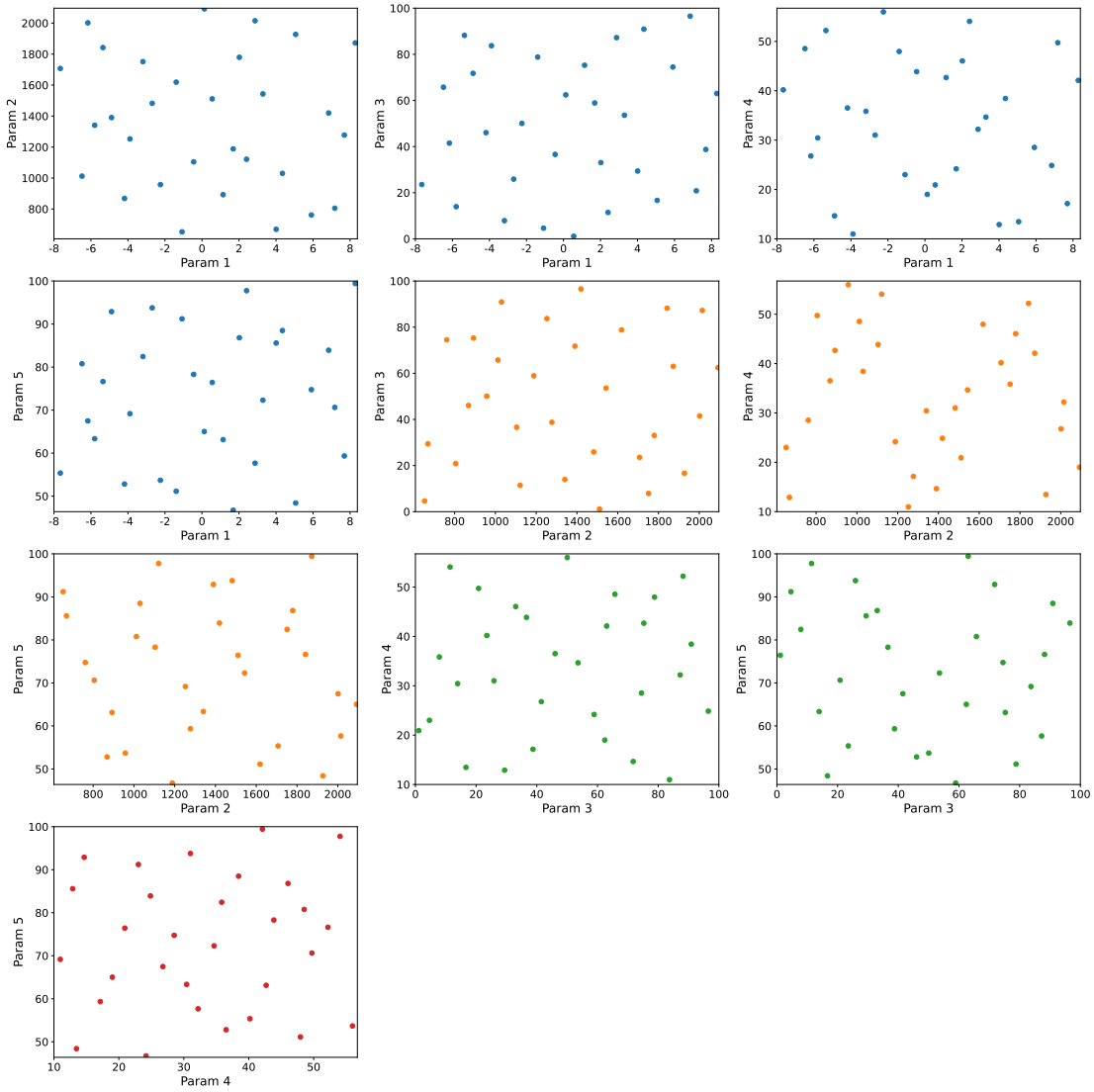


Figure B.1: Parameter initialization using Sobol sampling across the bounded parameter space for SOGA run on KP=(1000 rpm, 1400 Nm). Population size = 30.

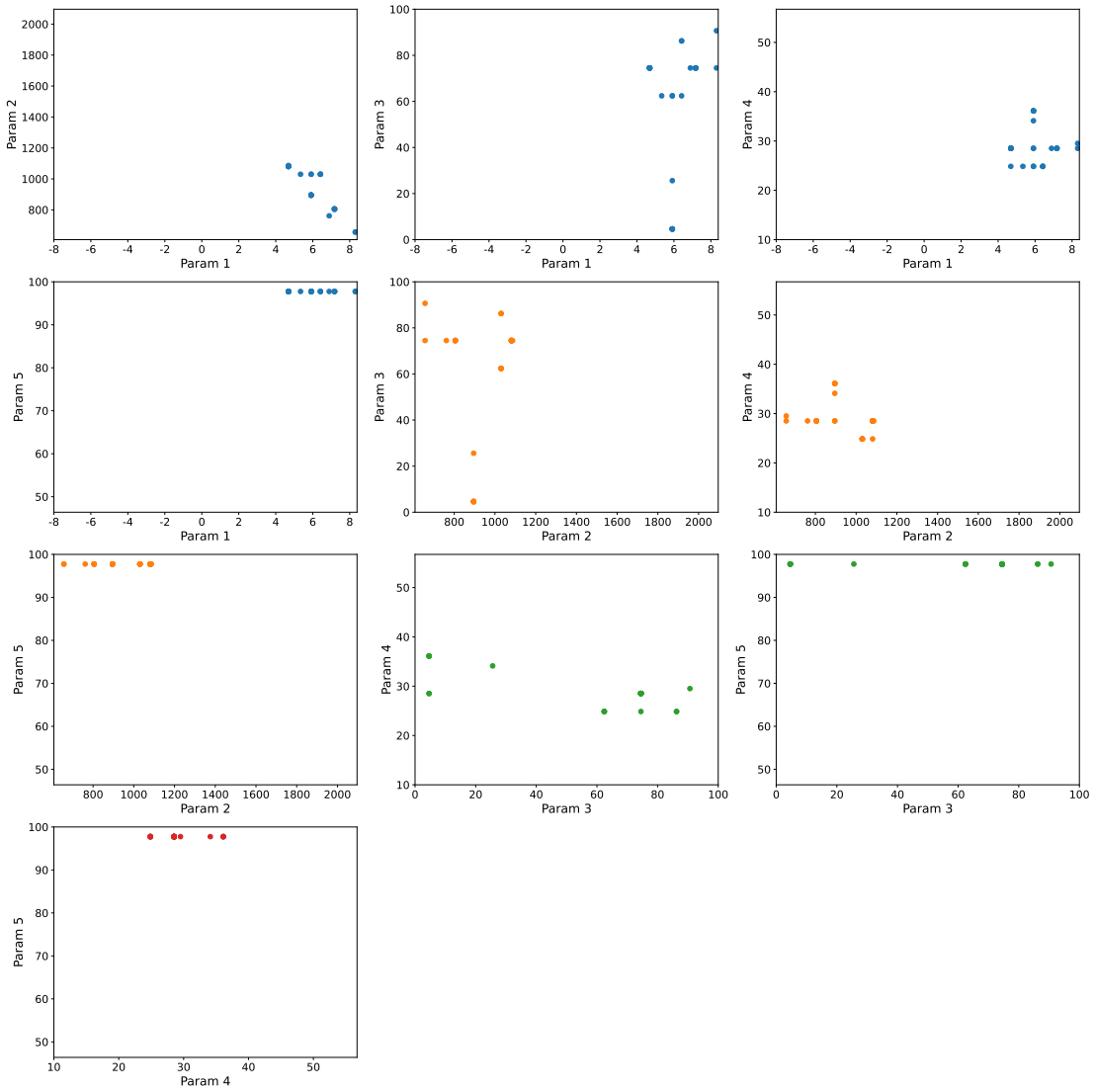


Figure B.2: Parameter configurations in the bounded parameter space for SOGA run on KP=(1000 rpm, 1400 Nm) at convergence. Population size = 30.

B.2 Parameter initialization

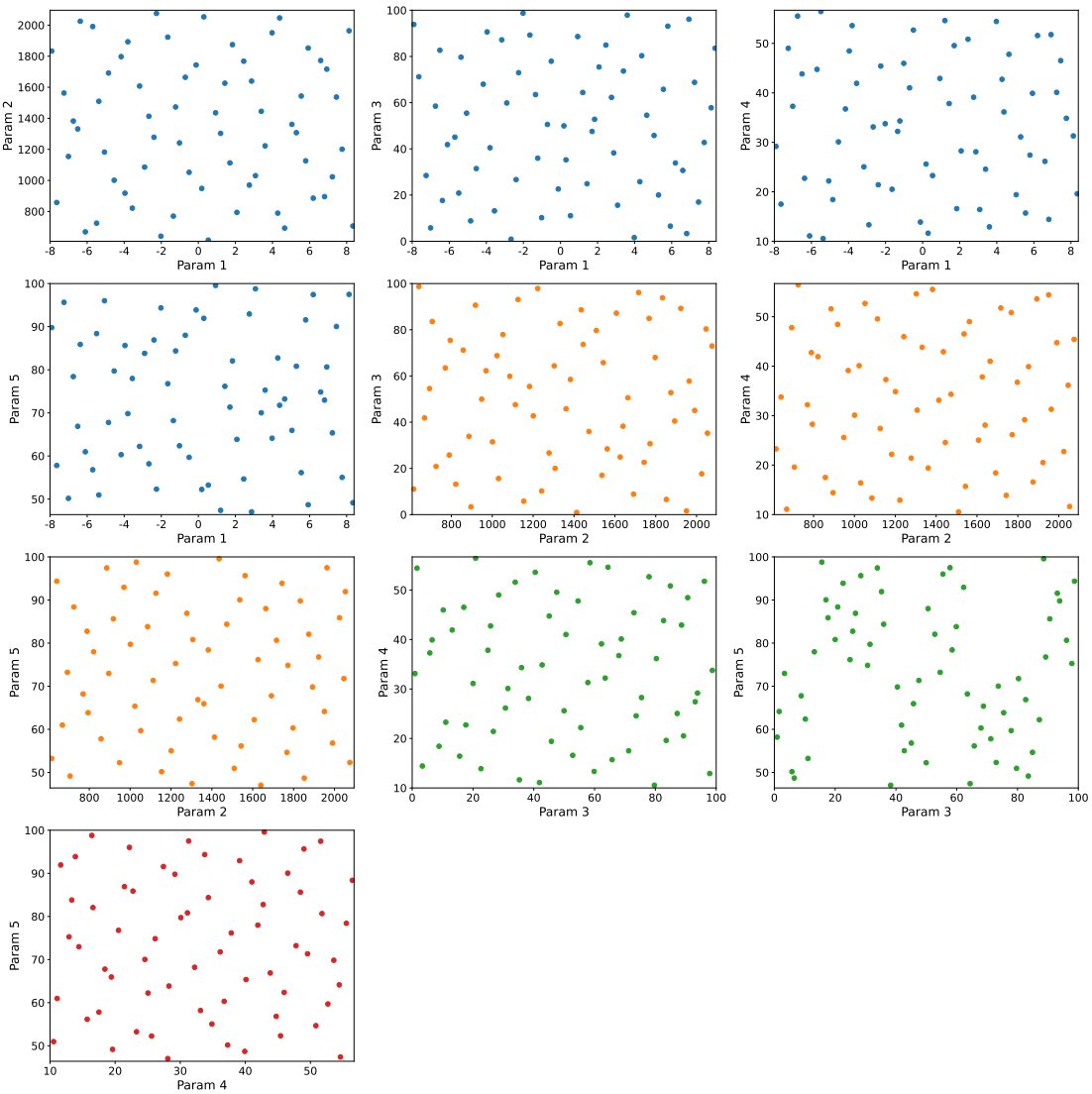


Figure B.3: Parameter initialization using Sobol sampling across the bounded parameter space for NSGA2 run on KP=(1000 rpm, 1400 Nm). Population size = 64.

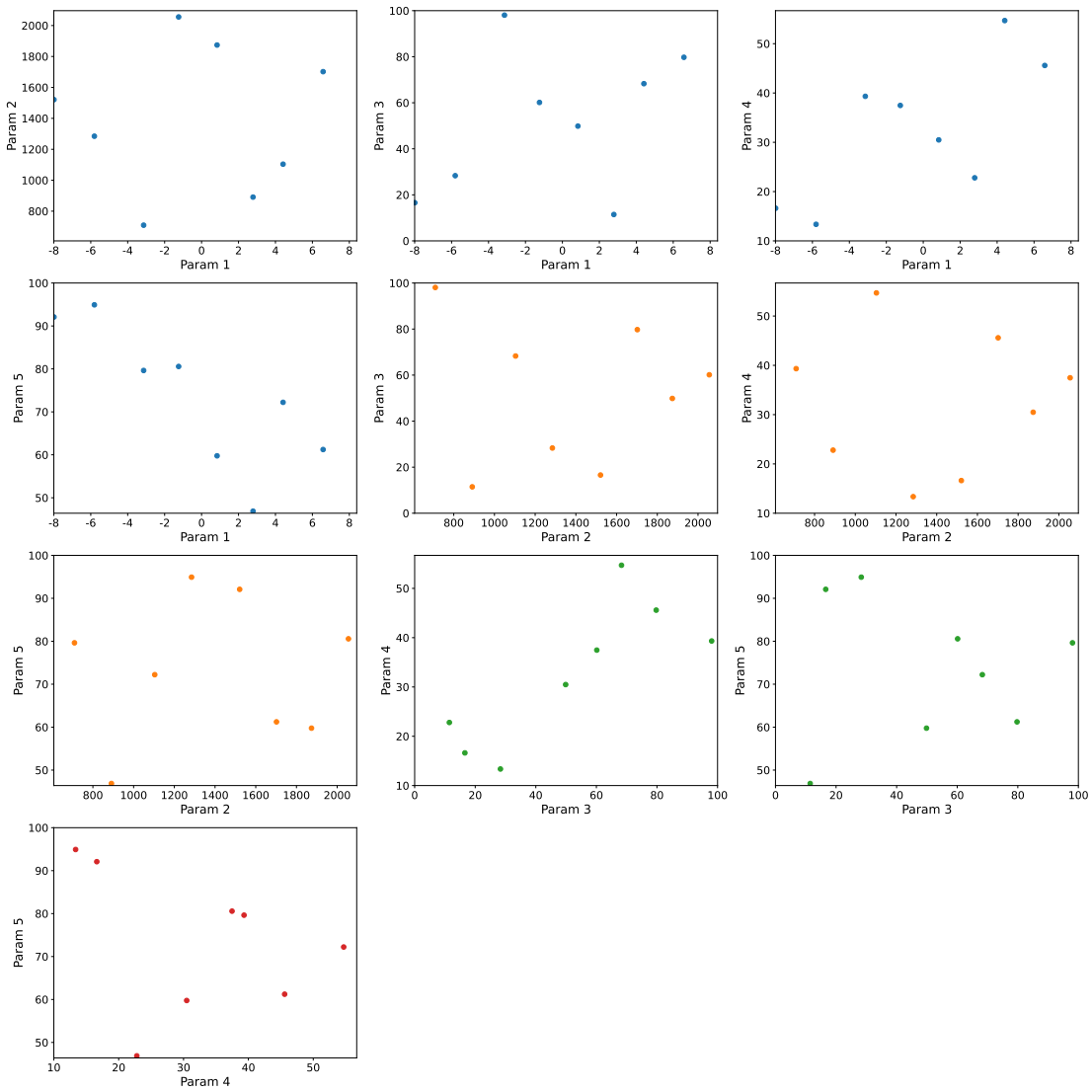


Figure B.4: Parameter initialization using Sobol sampling across the bounded parameter space for MOBO run on KP=(1000 rpm, 1400 Nm). Init size = 8.

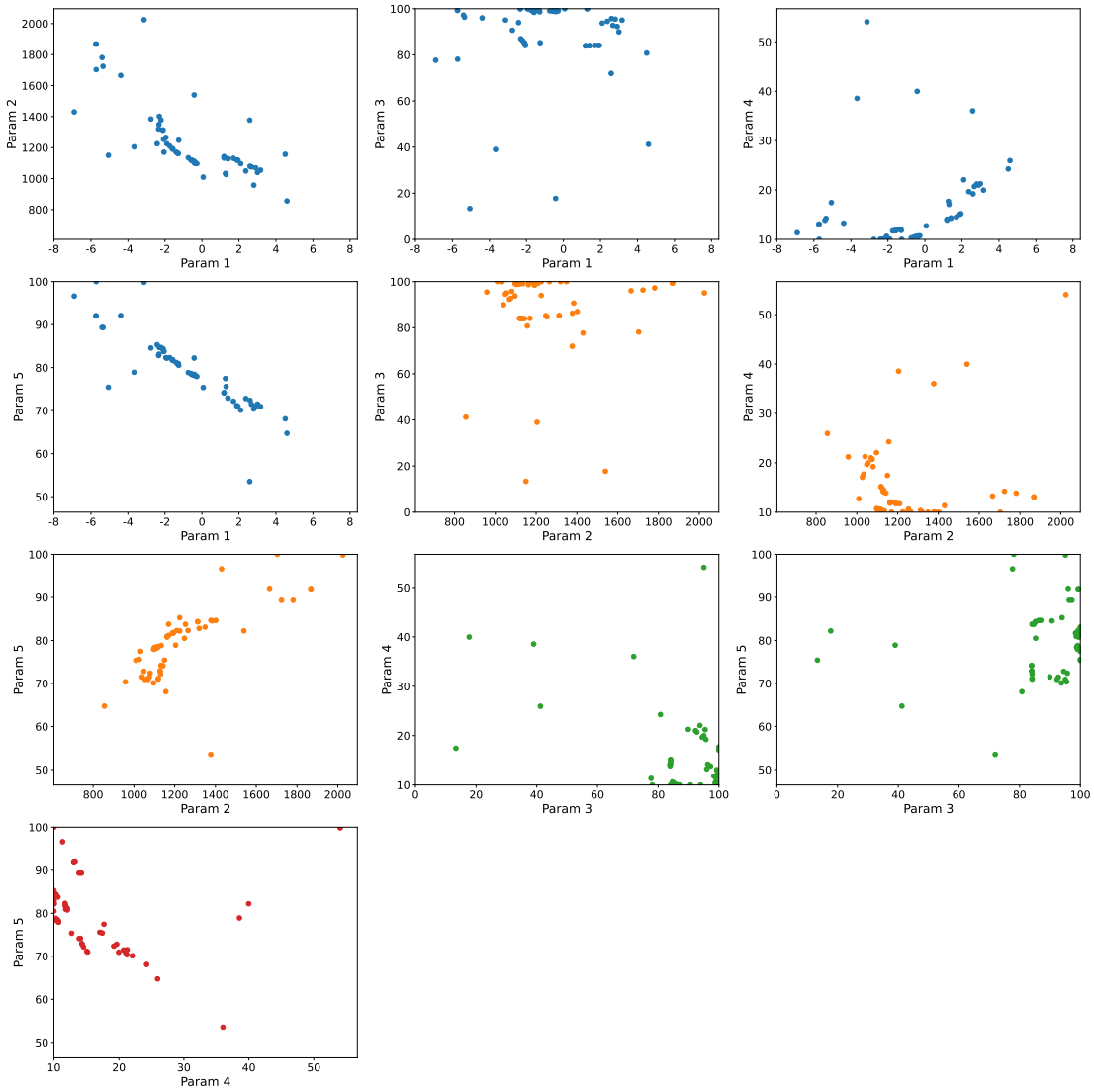


Figure B.5: Initial population used for the GA step in MOBOGA for KP=(1000 rpm, 1400 Nm).

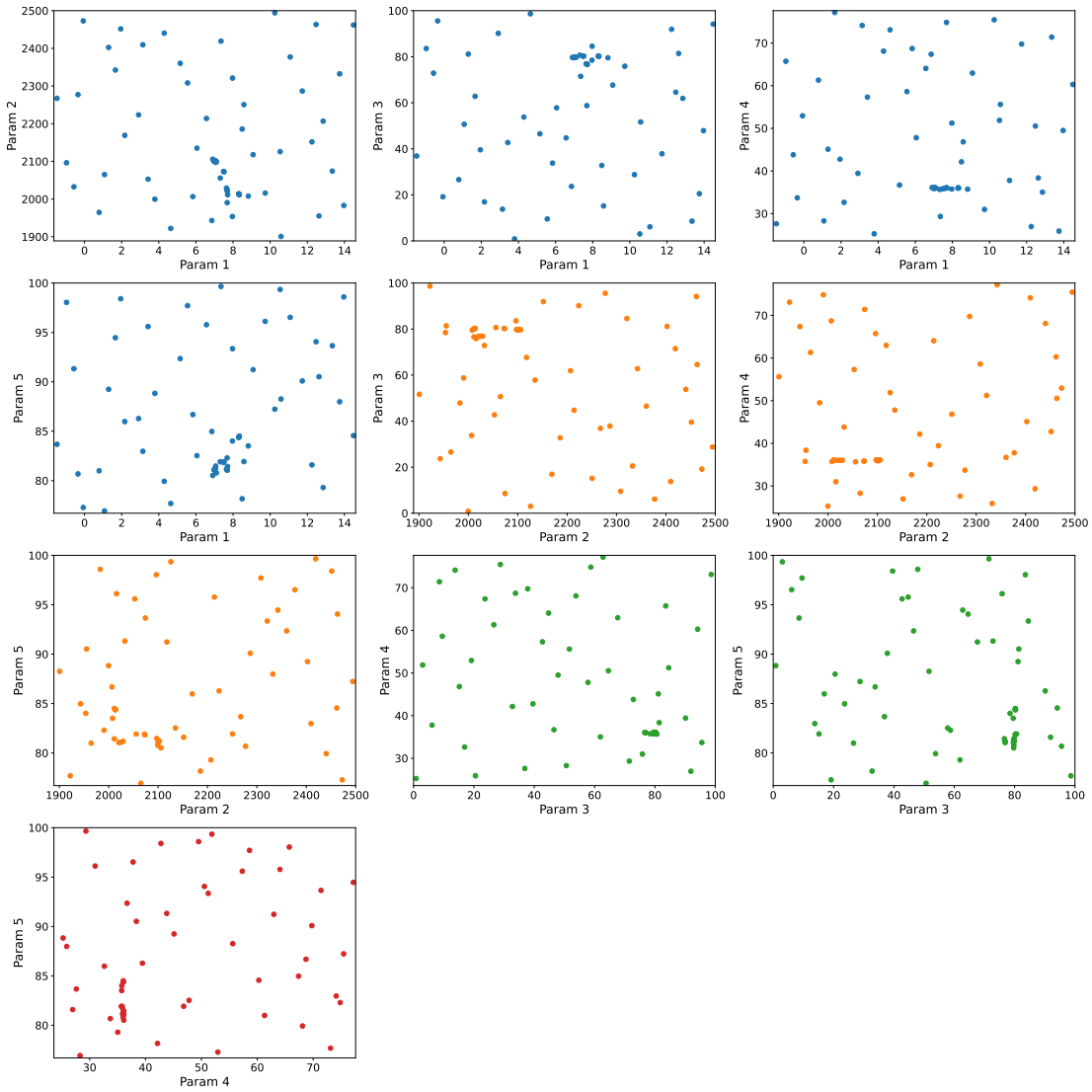


Figure B.6: Initial population used for the GA step in MOBOGA for KP=(1900 rpm, 2005 Nm).

B.3 Bi-objective parameter configurations

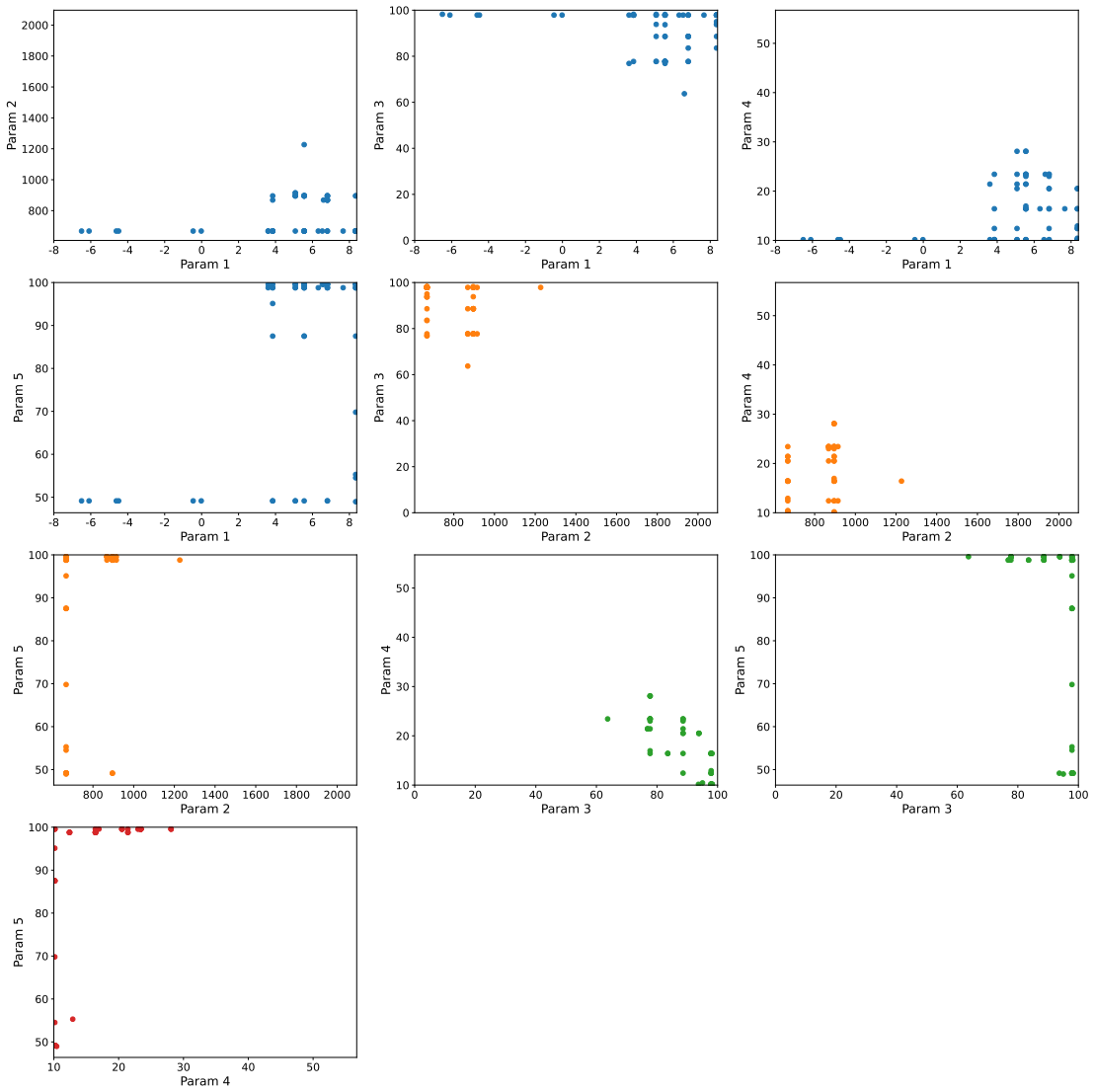


Figure B.7: Parameter configurations in the bounded parameter space for NSGA-II run on KP=(1000 rpm, 1400 Nm) at convergence in the bi-objective problem formulation.

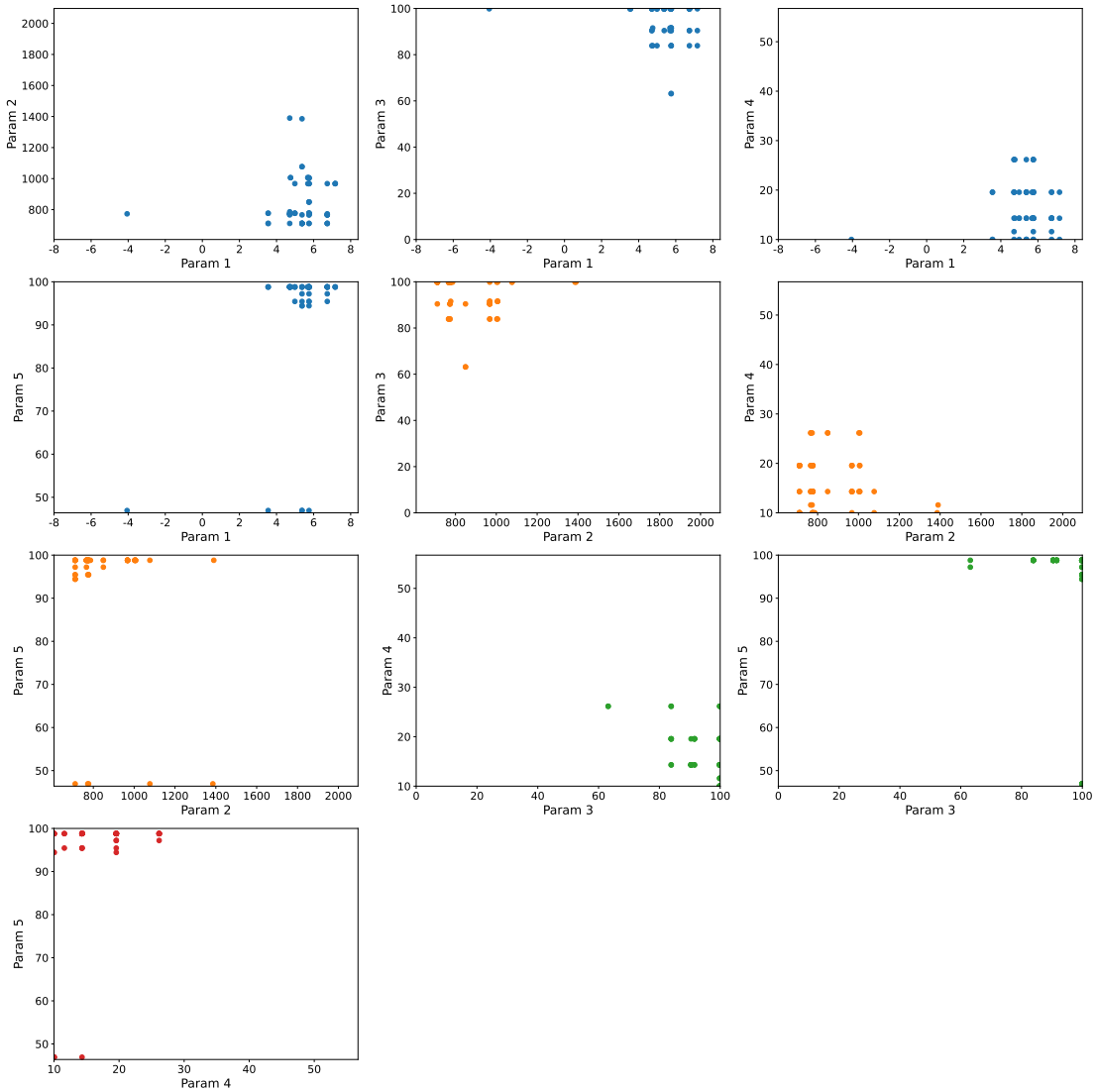


Figure B.8: Parameter configurations in the bounded parameter space for NSGA-III run on KP=(1000 rpm, 1400 Nm) at convergence in the bi-objective problem formulation.

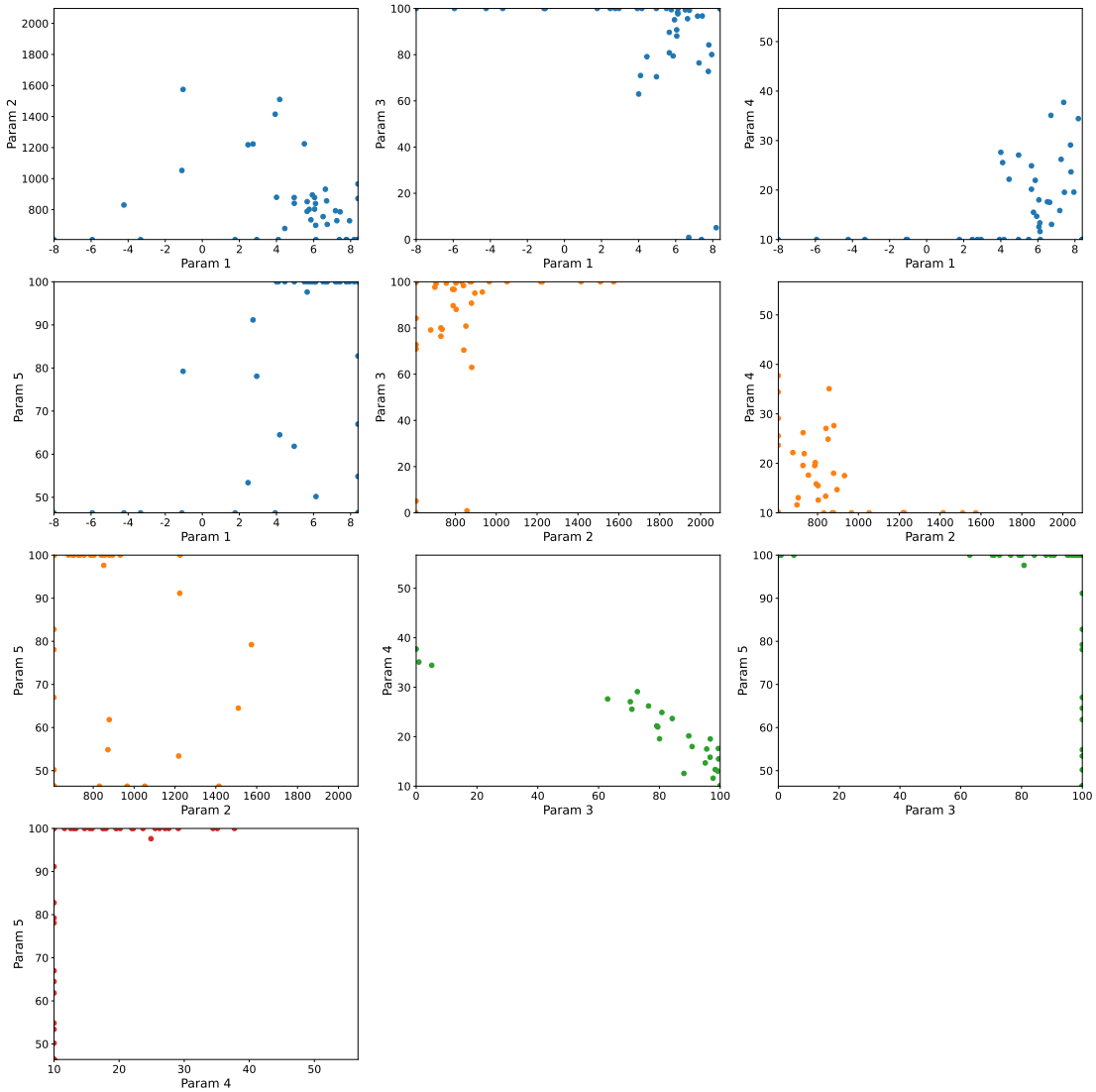


Figure B.9: Parameter configurations in the bounded parameter space for MOBO run on KP=(1000 rpm, 1400 Nm) at convergence in the bi-objective problem formulation.

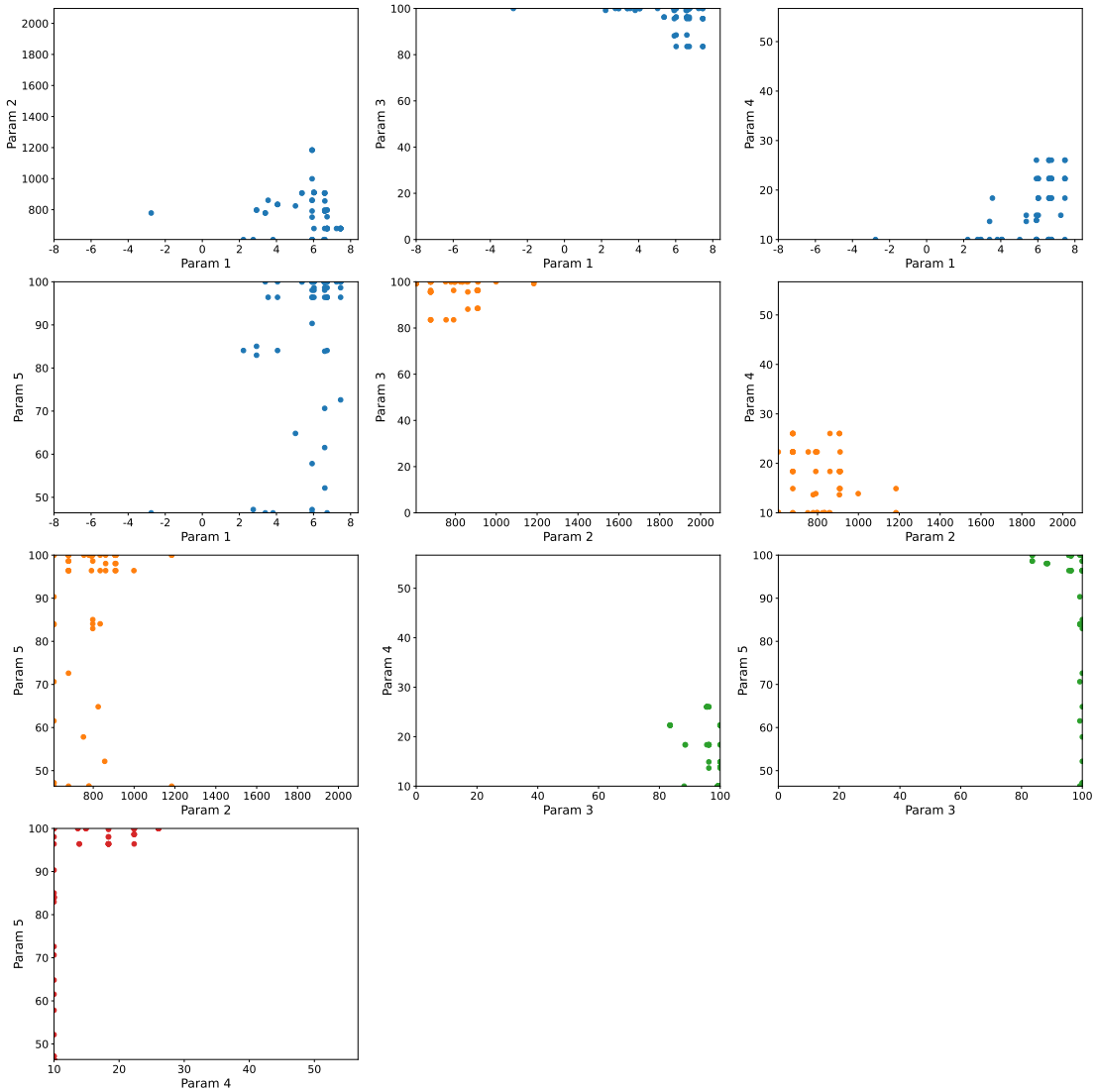


Figure B.10: Parameter configurations in the bounded parameter space for MOBOGA run on KP=(1000 rpm, 1400 Nm) at convergence in the bi-objective problem formulation.

B.4 Tri-objective parameter configurations

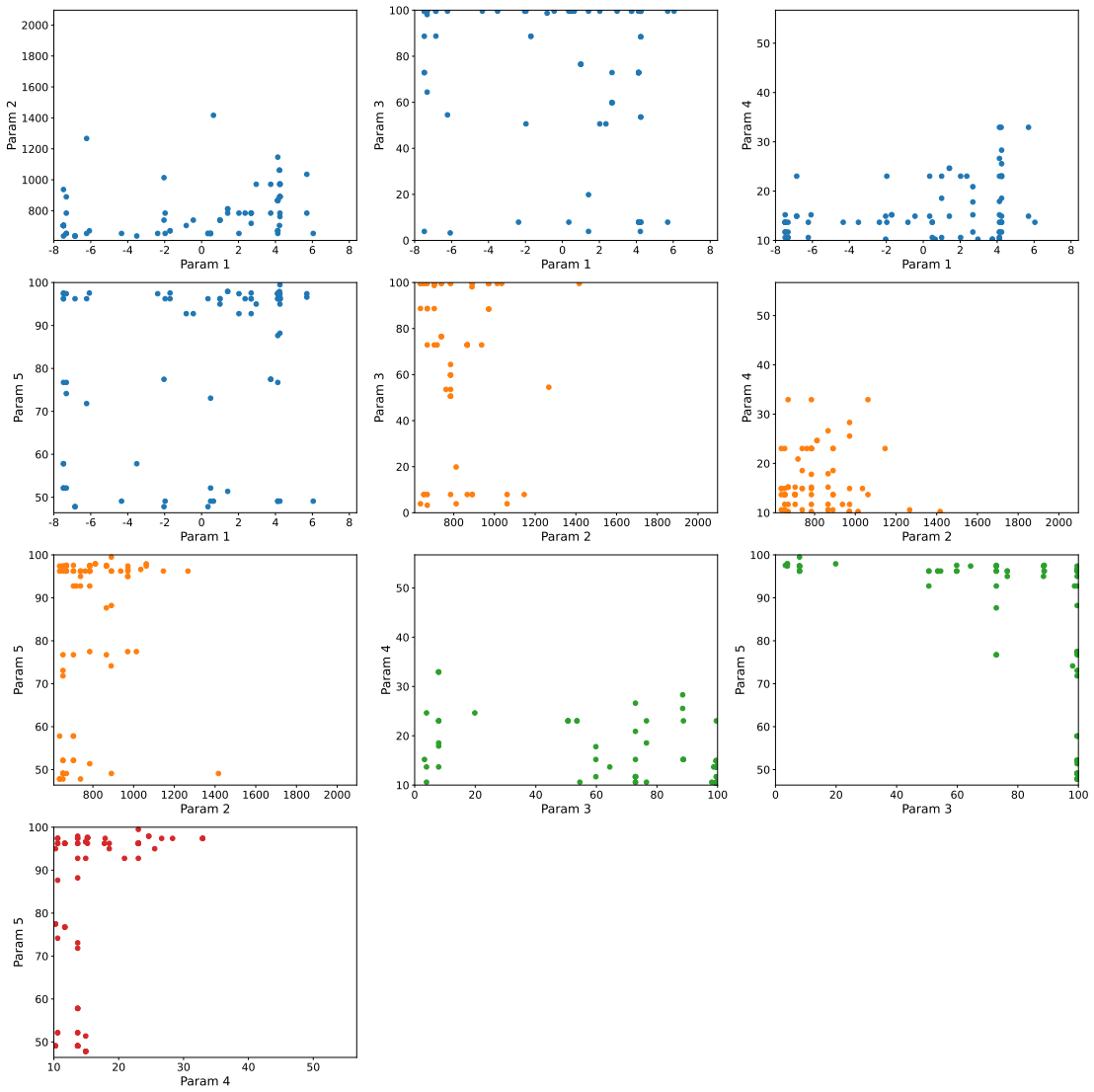


Figure B.11: Parameter configurations in the bounded parameter space for NSGA-II run on KP=(1000 rpm, 1400 Nm) at convergence in the tri-objective problem formulation.

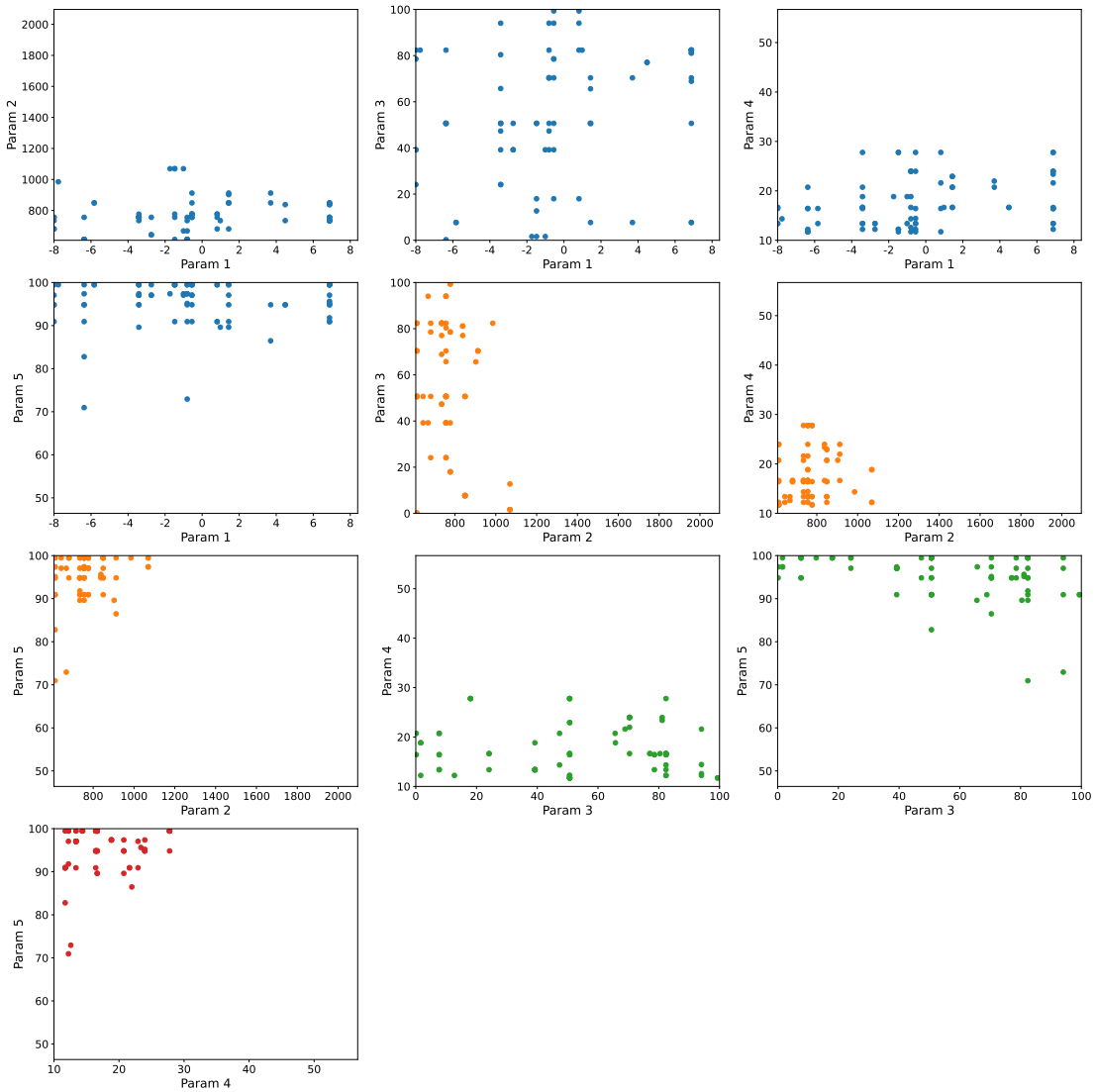


Figure B.12: Parameter configurations in the bounded parameter space for NSGA-III run on KP=(1000 rpm, 1400 Nm) at convergence in the tri-objective problem formulation.

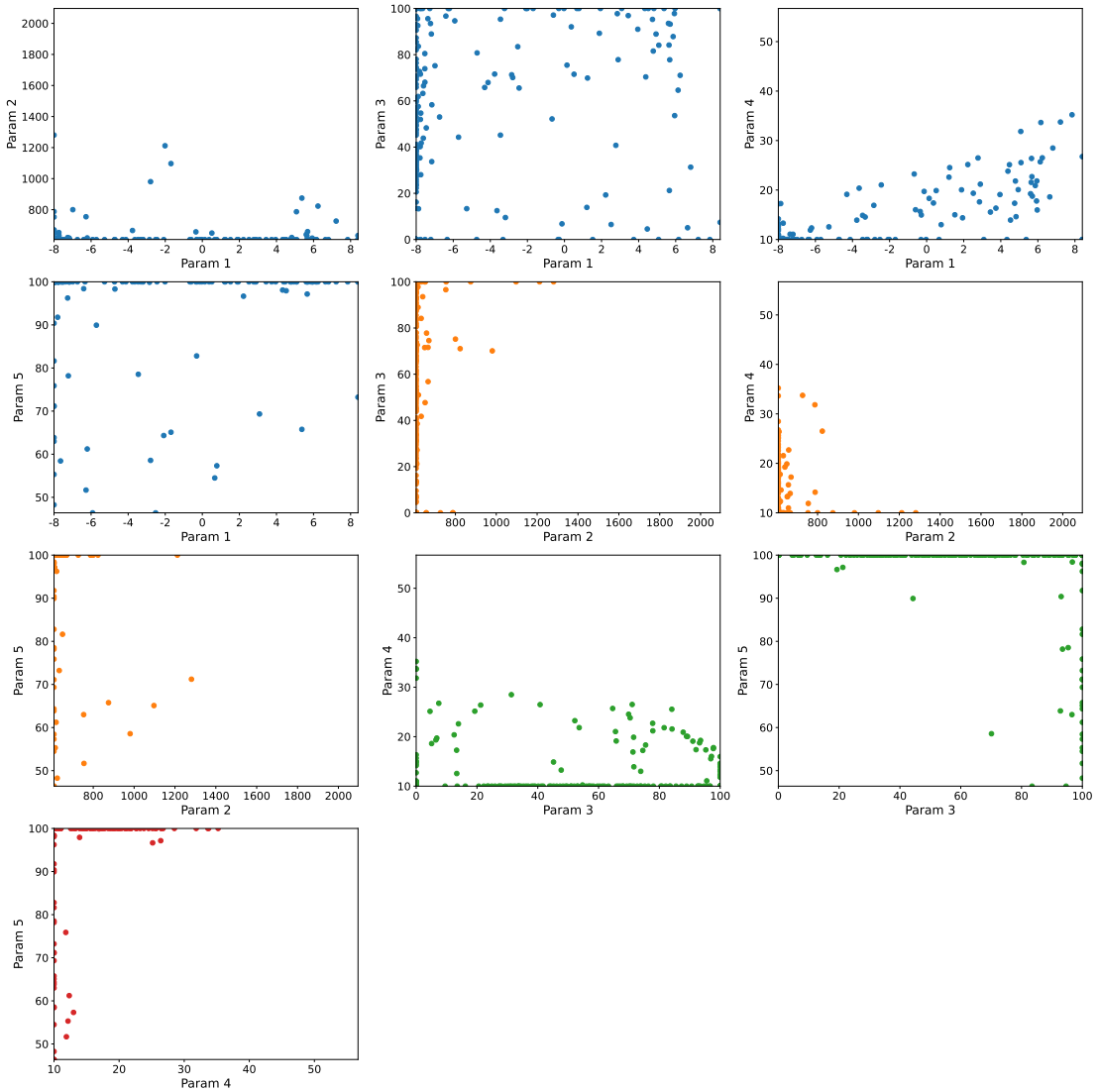


Figure B.13: Parameter configurations in the bounded parameter space for MOBO run on KP=(1000 rpm, 1400 Nm) at convergence in the tri-objective problem formulation.

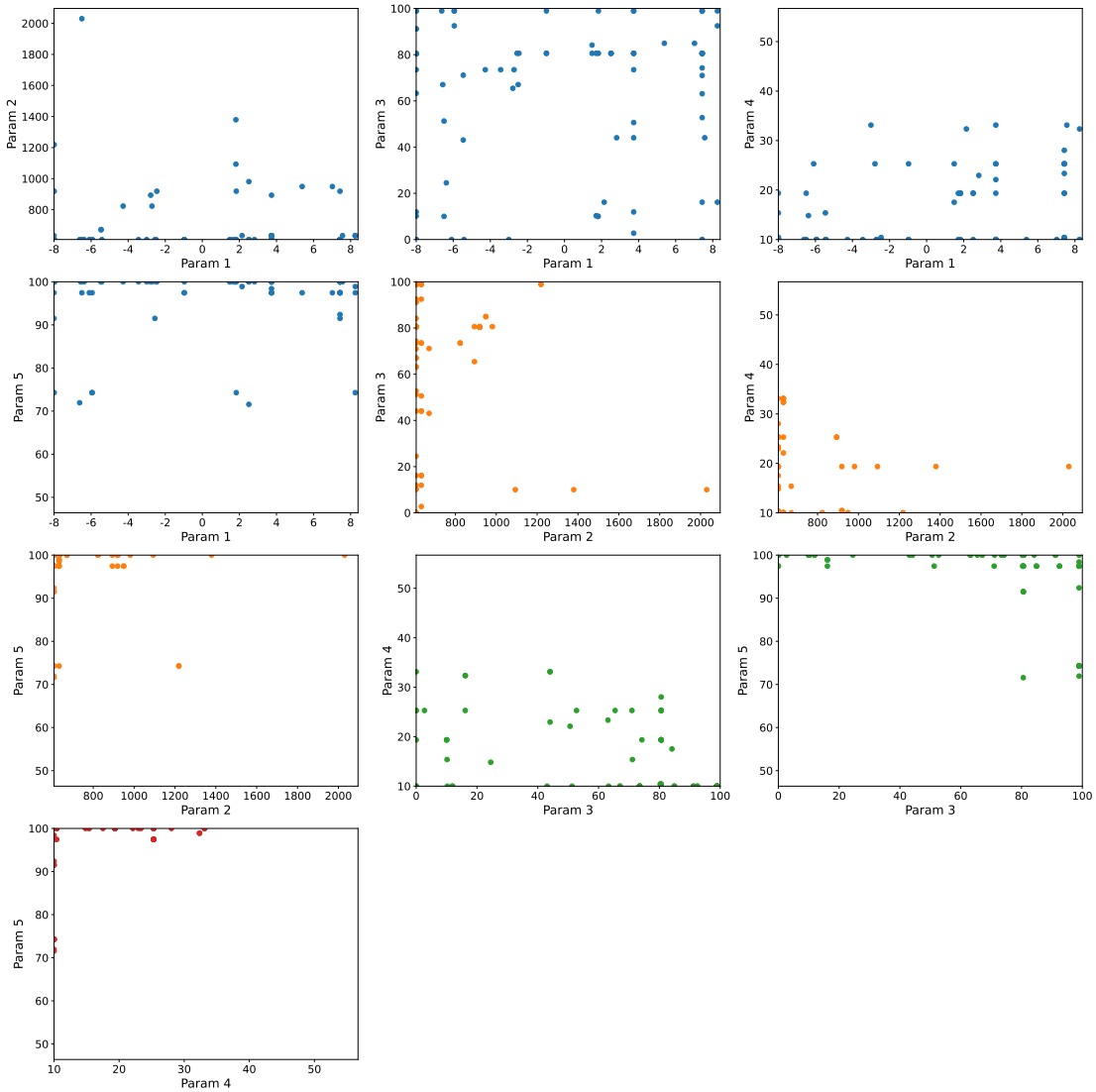
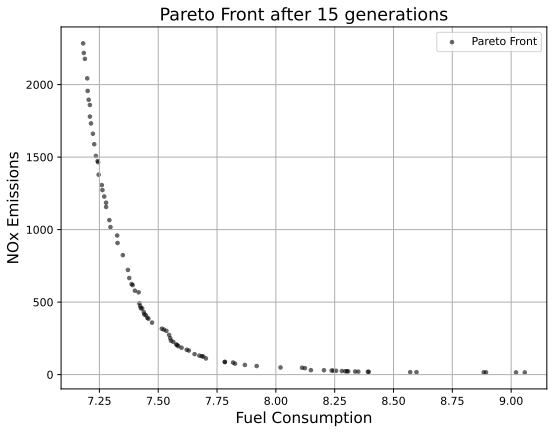


Figure B.14: Parameter configurations in the bounded parameter space for MOBOGA run on KP=(1000 rpm, 1400 Nm) at convergence in the tri-objective problem formulation.

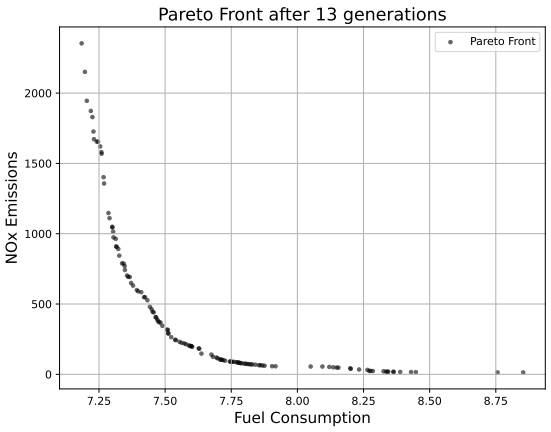
C

Plots

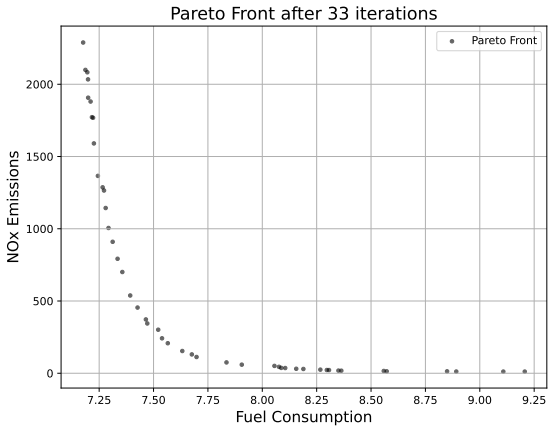
C.1 Pareto fronts



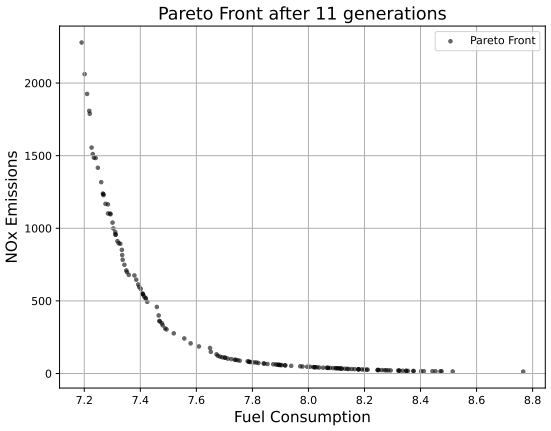
(a) Pareto front generated by NSGA-II.



(b) Pareto front generated by NSGA-III.

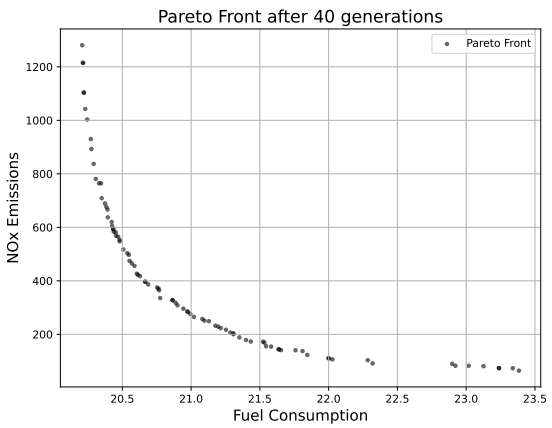


(c) Pareto front generated by MOBO.

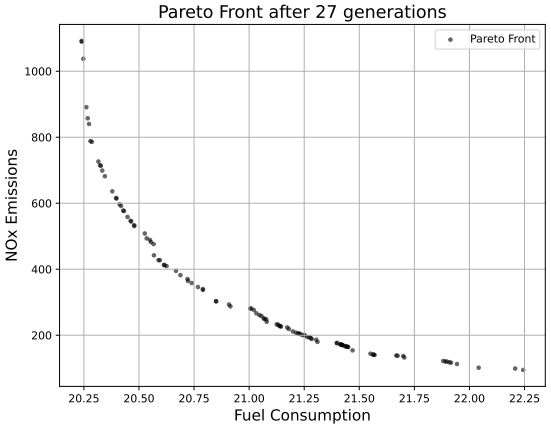


(d) Pareto front generated by MOBOGA

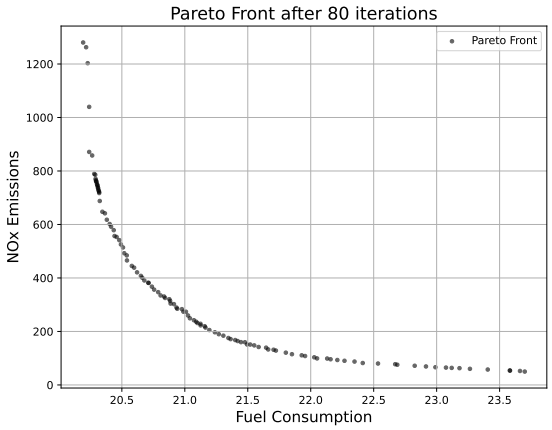
Figure C.1: Pareto fronts generated by each optimization method for KP=(1000 rpm, 1400 Nm). Each plot shows the best achieved hypervolume over the 5 runs for each method.



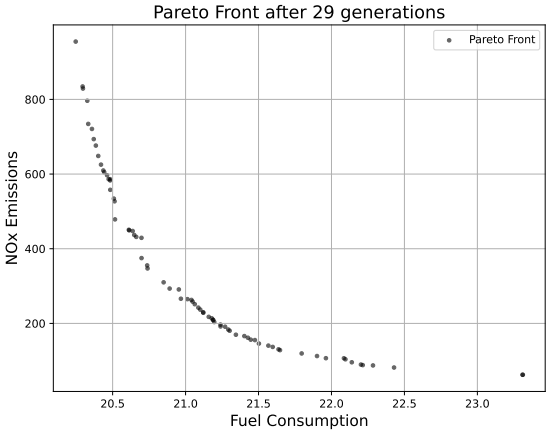
(a) Pareto front generated by NSGA-II.



(b) Pareto front generated by NSGA-III.



(c) Pareto front generated by MOBO.



(d) Pareto front generated by MOBOGA

Figure C.2: Pareto fronts generated by each optimization method for KP=(1900 rpm, 2005 Nm). Each plot shows the best achieved hypervolume over the 5 runs for each method.

C.2 BO: Acquisition Time

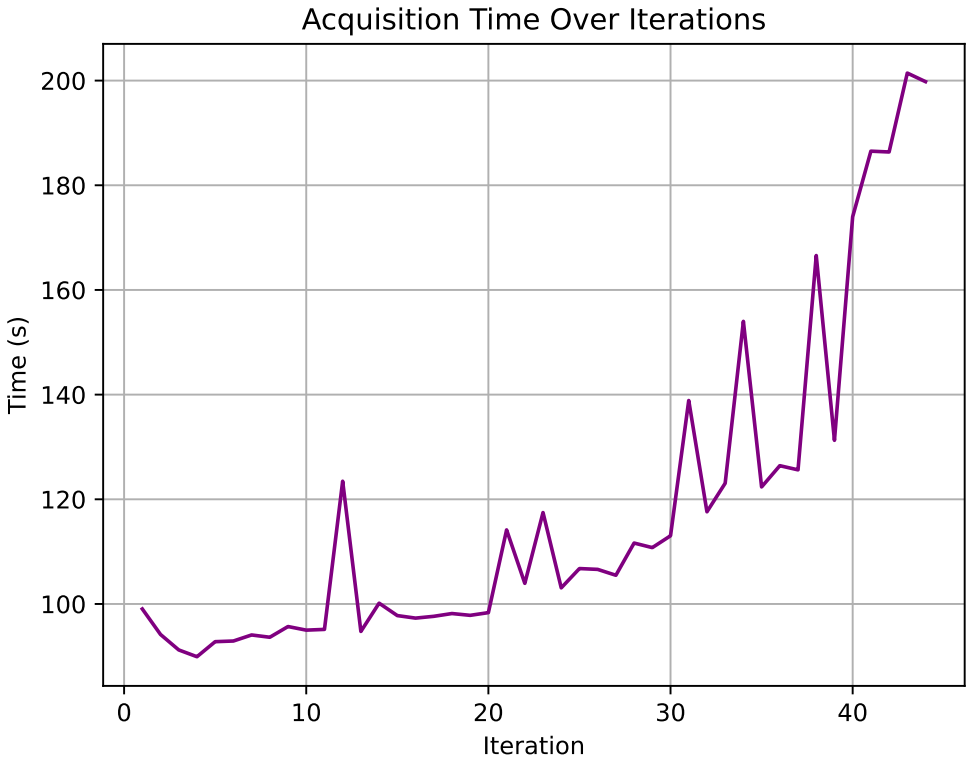


Figure C.3: Time spent optimizing the acquisition function at each iteration of a SOBO run on KP=(1000 rpm, 1400 Nm).

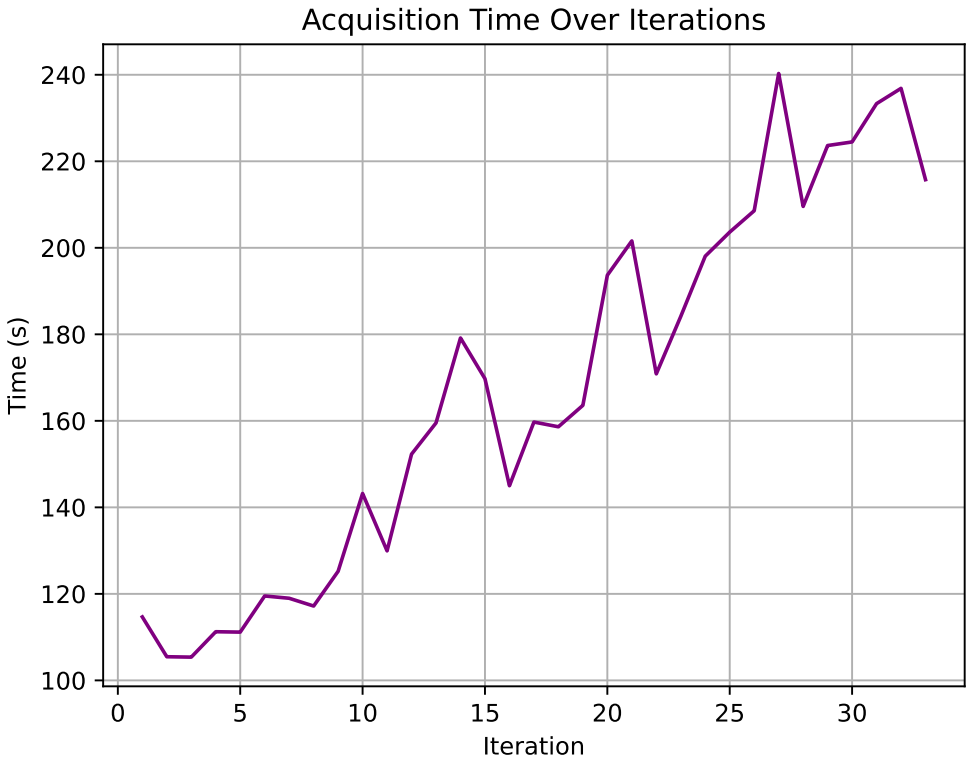


Figure C.4: Time spent optimizing the acquisition function at each iteration of a bi-objective MOBO run on KP=(1000 rpm, 1400 Nm).

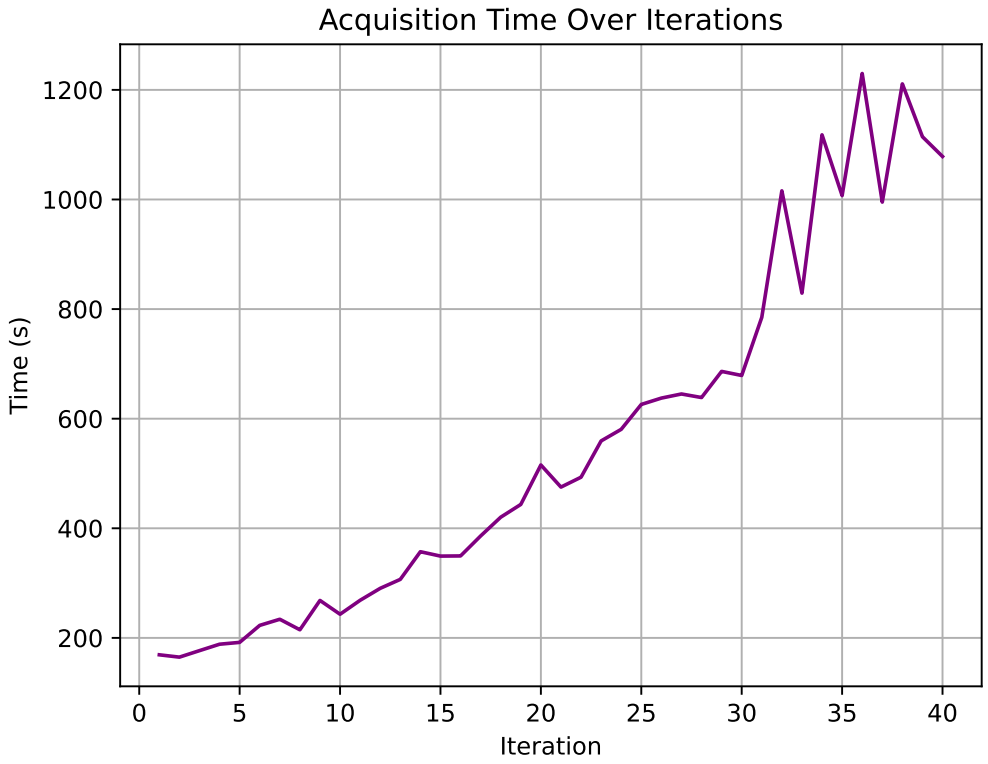


Figure C.5: Time spent optimizing the acquisition function at each iteration of a tri-objective MOBO run on KP=(1000 rpm, 1400 Nm).