Hans-Martin Wulfmeyer

# Offline Learning with a Sequence-based Selection Hyper-Heuristics for Evolutionary Multi-objective Optimization

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

Intelligent Cooperative Systems

Computational Intelligence

# Offline Learning with a Sequence-based Selection Hyper-Heuristics for Evolutionary Multi-objective Optimization

Master Thesis

Hans-Martin Wulfmeyer

December 31, 2021

Supervisor: Prof. Dr.-Ing. habil. Sanaz Mostaghim

Advisor: Dr.-Ing. Heiner Zille

**Hans-Martin Wulfmeyer:** *Offline Learning with a Sequence-based Selection Hyper-Heuristics for Evolutionary Multi-objective Optimization*
Otto-von-Guericke Universität
Intelligent Cooperative Systems
Computational Intelligence
Magdeburg, 2021.

# Abstract

Hyper-heuristics are methods that combine several different strategies from different optimization algorithms, which can improve the generality of the optimization process and searchability on a range of problems with different characteristics [1, 2]. So far, hyper-heuristics for solving continuous multi-objective problems have not received much attention in the literature, and even less attention has been paid to hyper-heuristics that use Multi-objective Evolutionary Algorithms (MOEAs) [1, 3–5].
In this thesis, an offline selection hyper-heuristic using MOEAs is developed and investigated. To the best of the author's knowledge, the Offline Learning Hyper-Heuristic collaborative Multi-objective Evolutionary Algorithm (OHHMOEA) as presented in this paper is the first case of an offline selection hyper-heuristic using MOEAs.
The offline hyper-heuristic is evaluated on nine different optimization problems. The performance of the hyper-heuristic is evaluated and compared with the MOEAs used, and the results of the offline hyper-heuristic are thoroughly analyzed. It is shown that OHHMOEA performs well and provides intriguing results. The results highlight the advantage of hyper-heuristics in combining several different strategies to solve optimization problems.

# Contents

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| **EA** | Evolutionary Algorithm |
| **MOEA** | Multi-objective Evolutionary Algorithm |
| **HHcMOEA** | Hyper-heuristic collaborative Multi-objective Evolutionary Algorithm |
| **OHHMOEA** | Offline Learning Hyper-Heuristic collaborative Multi-objective Evolutionary Algorithm |
| **HV** | hypervolume |
| **GD** | Generational Distance |
| **IGD** | Inverted Generational Distance |
| **NSGA-II** | Nondominated Sorting Genetic Algorithm II |
| **PBI** | penalty-based Boundary Intersection |
| **CD** | critical difference |

# 1 Introduction and Motivation

Evolutionary Algorithms (EAs) are search methods that utilize higher-level strategies, which aim to escape local optima, to perform robust search operations for a wide variety of optimization problems. EAs imitate principles of biological evolution and belong to the family of meta-heuristics. When applied to problems with multiple objectives, these algorithms are called Multi-objective Evolutionary Algorithms (MOEAs) [3, 6].
Optimization problems with enormously high computation power or time requirements benefit significantly from the usage of EAs. EAs have also shown promising results in many real-world applications such as cancer research [7], wind turbine design [8], and aerodynamic drag optimization for high-speed trains [9].
Currently, there is a vast amount of existing MOEAs to solve multi-objective optimization problems with a wide variety of strategies. The PlatEMO framework, which is utilized in this thesis, currently supports 162 algorithms [10]. The combination of several different strategies can help to improve the generality of the optimization process and may improve the searchability on a set of problems with diverse characteristics [1, 2].
Hyper-heuristics are high-level strategies, which aim to select the best heuristic at different decision points to solve an optimization problem, and are one method to combine several different strategies [1, 11].The literature to date has focused mainly on the application of hyper-heuristics to combinatorial (non-continuous) optimization problems, such as bin packing, scheduling, routing and timetabling problems [1, 4, 12]. Far less focus has been given to continuous multi-objective problems and even less attention was given to hyper-heuristics using MOEAs [1, 3–5].
In literature, selection hyper-heuristics utilizing MOEAs have only been used in combination with online learning to the best of the author's knowledge [2, 4, 13–18].
Online learning is faced with computation budget constraints and is likely based on sub-optimal decision-making since decisions are made ad hoc and

following assumptions about the applied algorithms. Offline learning can overcome these drawbacks by making decisions a posteriori based on the complete results. In this thesis, the Offline Learning Hyper-Heuristic collaborative Multi-objective Evolutionary Algorithm (OHHMOEA) is introduced. To the best of the author's knowledge, OHHMOEA is the first instance of an offline selection hyper-heuristic using MOEAs. It shows remarkable results in optimizing of multi-objective problems. Moreover, the analysis of the offline learning results gives important insights into the combination of different strategies in optimization and hyper-heuristics.

## 1.1 Goals

The main objective of this thesis is to outline the concept for an offline hyper-heuristic, named Offline Learning Hyper-Heuristic collaborative Multi-objective Evolutionary Algorithm (OHHMOEA). The hyper-heuristic contains a fixed set of MOEAs, which are optimized in a certain sequence to optimize a given problem. The central objective of this thesis is to evaluate the results of the hyper-heuristic based on the learned sequences by answering the following questions:

1. Do certain sub-sequences of MOEAs occur more often than others?
2. Are the resulting sequences problem-specific?
3. How does the OHHMOEA perform compared to the selected MOEAs?
4. Are the results of the sequences 'stable' if the population size, the function evaluations, the objective size, or decision variables are increased?
5. Are the learned sequences transferable to other problems?
6. Does the offline hyper-heuristic benefit from a reduced algorithm set?

## 1.2 Structure of Thesis

In Chapter 2, the fundamental knowledge, which is necessary for this thesis, regarding Evolutionary Algorithms is explained. It will also introduce several state-of-the-art MOEAs focusing on the main functionality and the differences. Afterward, an overview of the related work regarding hyper-heuristics is given in Chapter 3. Chapter 4 will explain the design and implementation of OHHMOEA. The results and the evaluation of the experiments on OHHMOEA are presented in Chapter 5, followed by Chapter 6, which contains the conclusion and future work to this thesis.

# 2 Evolutionary Multi-Objective Optimization

In this chapter, the main components of evolutionary algorithms as they relate to the topic of this thesis are explained. First, it will give a general introduction to Evolutionary Algorithms in Section 2.1. Afterward, quality indicators are presented in Section 2.2. The various MOEAs utilized in this thesis are introduced in Section 2.3. In Section 2.4, the chapter concludes with an introduction to the benchmark problems utilized for the evaluation.

## 2.1 Evolutionary Optimization

Evolutionary Algorithms are optimization techniques that imitate principles of biological evolution and belong to the family of meta-heuristics. Meta-heuristics, especially Evolutionary Algorithms, do not guarantee to find an optimal solution but usually find solutions with a sufficiently good quality [6]. Optimization problems are defined as displayed in Definition 2.1.

**Definition 2.1** (Optimization problem [6])**.** An *optimization problem* consists of the pair $(\Omega, f)$.
The search space of potential solutions $\Omega$ and an evaluation function $f : \Omega \to \mathbb{R}$ that assigns a quality assessment $f(\omega)$ to each solution $\omega \in \Omega$.
An element $\omega$ is an **exact solution** of $(\Omega, f)$ if it is a global optimum of $f$.
An element $\omega$ is a **global optimum** of $f$ if: $\forall \omega' \in \Omega \ : \ f(\omega) \succeq f(\omega')$
Where $f(x) \succeq f(z)$ means that $x$ has a quality $f$ better than or equal to $f(z)$

Evolutionary Algorithms usually contain four main parts. A fixed-size population of solutions, a mating pool selection operator, a genetic operator, and an environmental selection [6].
The mating pool selection operator selects two parent solutions from the population. The genetic operator then uses the two parents to produce a

child solution. The genetic operator usually combines the two parents via crossover and mutates the produced child. This child gets added to the population, and the environmental selection discards the worst solution. The main functionality of environmental selection is to keep the population at a constant size.

Algorithm 1 displays one basic Evolutionary Algorithm named Genetic Algorithm. In this thesis, all algorithms following a similar scheme to the displayed Genetic Algorithm are referred to as Evolutionary Algorithms. The *stopcondition* in Algorithm 1 is usually satisfied after a specific amount of iterations, generations, or function evaluations. Its truth value is evaluated each time it is called. Definition 2.1 and also the Algorithm 1 assume that the

**Algorithm 1** Genetic Algorithm

```
Input: Optimization Problem $\mathcal{Z}$
Output: Population $\mathcal{P}$
1: $\mathcal{P} \leftarrow$ Initialization($\mathcal{Z}$)
2: $\mathcal{P}$.CalculateFitness($\mathcal{Z}$)
3: while stopcondition $\neq$ True do
4:     MatingPool $\leftarrow$ FitnessSelection($\mathcal{P}$)
5:     Offspring $\leftarrow$ GeneticOperator(MatingPool)
6:     Offspring.CalculateFitness($\mathcal{Z}$)
7:     $\mathcal{P} \leftarrow$ EnvironmentalSelection($\mathcal{P}$, Offspring)
   return $\mathcal{P}$
```

problem is single-objective. With more than one objective, these problems are categorized as multi-objective problems. Optimization problems become more complex to solve when more objectives exist. The comparison of the solutions in the selection process has to be changed because these problems have different objectives, which usually compete with each other to some degree. A solution can not be perfect in one objective while simultaneously being very good in another objective if the objectives are conflicting. It is necessary to find a trade-off between objectives.

Therefore, solutions to multi-objective problems are usually compared by using Pareto-dominance, which is based on the definition of Pareto-optimality in Definition 2.2. The definition of Pareto-optimality is related to the definition of the *global optimum* in Definition 2.1.

**Definition 2.2** (Pareto-optimality [6])**.** A solution $\omega$ is called Pareto-optimal w.r.t. the objective functions $f_i$ with $i \in 1, \cdots, k$ if there does **not** exist any other $\omega'$ in the search space $\Omega$ for which these two properties are both true:

$$f_i(\omega') \succeq f_i(\omega) \qquad \forall i \ \ 1 \leq i \leq k \tag{2.1}$$

$$f_i(\omega') \succ f_i(\omega) \qquad \exists i \ \ 1 \leq i \leq k \tag{2.2}$$

Optimization algorithms do not consider the whole search space when comparing solutions to each other, so Pareto-dominance is used instead of Pareto-optimality. Pareto-dominance (or non-dominance) only considers the known solutions in the population. A solution $s$ is considered non-dominated if there does **not** exist any other solution $s'$ in the population for which the Equations 2.1 and 2.2 in Definition 2.2 are true.
The solutions in the population are usually mutually non-dominated to each other, which means they are theoretically of equal quality in the objective space. That is especially the case when methods based on Pareto-dominance are utilized. That raises the question how the quality of a specific solution in the population can be determined if not by Pareto-dominance. That is especially important when the goal is to compare different optimization methods on benchmark problems. Usual qualities compared are diversity and convergence. Diversity is meant to prevent premature convergence to specific suboptimal solutions and also ensures that the solutions are spread in the objective space [6]. Convergence is usually measured in comparison to the Pareto-optimal front. Convergence and diversity can be calculated by metrics called quality indicators.

## 2.2 Quality Indicators

An important topic in MOEA is quality indicators, also called evaluation metrics, which are also used for the calculation of the quality contribution of specific solutions to the population [6]. These metrics can then calculate which solutions to keep to minimize the quality loss. More importantly, these metrics are used to compare two Pareto-front approximations with each other. This comparison is usually performed with the Pareto-optimal front, which is a set of Pareto-optimal solutions. Normally, convergence does not consider diversity, which is why indicators that also measure diversity exist. Some indicators

also measure the diversity in the decision space.
The quality indicators mentioned in this thesis are the HV, R2, the GD and the IGD. For the evaluation of MOEAs, these quality indicators are used in conjunction with benchmark problems.

### Hypervolume

The hypervolume (HV) indicator is defined as the area or volume that is dominated by a Pareto front approximation, with the boundary of that volume defined by a reference point [6, 19, 20].
The hypervolume for a Pareto-set approximation $A$ and a reference point $\vec{r}$ is defined as follows:

$$HV(A, \vec{r}) = V\left(\bigcup_{\vec{a} \in A} H(\vec{a}, \vec{r})\right), \tag{2.3}$$

were $H$ returns the hyper-cubiod that is spanned between the objective vector $\vec{a}$ and the reference point $\vec{r}$ and $V$ returns the volume of the union of all hyper-cuboids.
One of the first methods proposed to calculate the HV and the most widely known is the Lebesgue measure [21]. However, its time complexity has been shown to be exponential in relation to the dimension of the optimization problem. Algorithms that calculate the HV with a lower time complexity have also been proposed. If the exact HV values are not of a particular interest the HV can also be approximated by Monte Carlo simulation, which is able to provide values within a 0.2% error margin and with a much lower time complexity in higher dimensions [22].
In Figure 2.1 an example for the hypervolume is shown. For the HV indicator, Pareto front approximations that are further from the reference point have larger hypervolume values. Furthermore, Pareto front approximations in which the solutions are further apart produce larger hypervolume values, than if the solutions are closer together. Hence the HV provides a qualitative measure of convergence as well as diversity (spread in the objective space) [23].

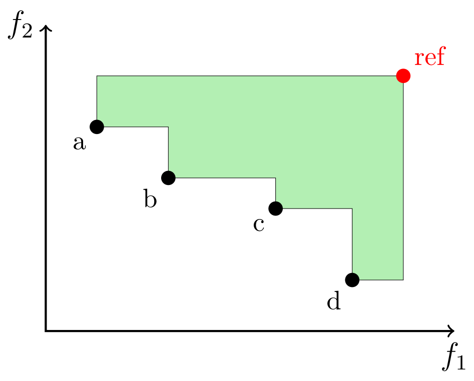


Figure 2.1: Example for the hypervolume indicator for two objectives. The pareto set approximation are the points a,b,c,d and the reference point is shown in red. The hypervolume is the green area.

### R2 Indicator

The R2 indicator uses utility functions, which map an objective vector to a scalar value. Utility functions may be based on the weighted sum and or weighted Tchebycheff functions.
The R2 indicator for a solution set $A$ with a set of utility functions $U$ is defined as [24]:

$$R2(A, U) = -\frac{1}{\mid U \mid} \sum_{u \in U} \max_{\vec{a} \in A} \{u(\vec{a})\}. \tag{2.4}$$

The R2 indicator summarizes the maximum utility function values in the solution set and then builds the negative of the average. The above definition is for the usage in minimization. In Figure 2.2 an example with weighted sum functions is shown.
The HV and the R2 indicator are often compared to each other because they have similar properties and have been shown to be highly correlated with each other [24]. The R2 indicator is often preferred over the HV because the runtime of the HV calculation is exponential, and using the HV results in populations that are biased towards the knee regions [25, 26]. However, the HV indicator fulfills the important property of strict monotonicity, i.e., the HV of a set that dominates another set has to have the better HV value. For the R2 indicator, the values for such sets may be equivalent in some specific cases, which means it is only weakly monotonic [24].
In Figure 2.2 an example for the R2 indicator is shown.

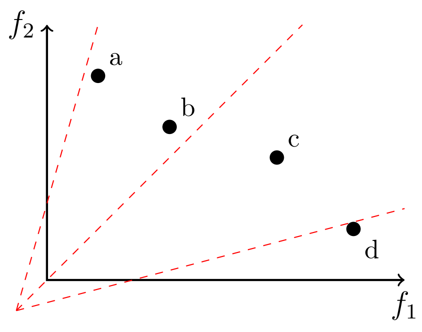


Figure 2.2: Example for the R2 indicator using three weighted sum functions as utility functions, which are displayed as dashed red lines. The solutions closest to a dashed line maximize the respective weighted sum function. The solutions that are selected are a, b and d.

### GD and IGD

The Generational Distance (GD) and Inverted Generational Distance (IGD) are related to each other. The equation to calculate both is similar with only a minor change. For a Pareto-optimal set $P$ and a Pareto-set approximation $A$ the GD and IGD are defined as follows [27]:

$$GD(P, A) = \frac{1}{\mid A \mid} \left( \sum_{\vec{a} \in A} d(\vec{a}, P)^q \right)^{1/q} \tag{2.5}$$

$$IGD(P, A) = \frac{1}{\mid P \mid} \left( \sum_{\vec{p} \in P} d(\vec{p}, S)^q \right)^{1/q} \tag{2.6}$$

where $d(\vec{g}, X)$ returns the Euclidean distance of the objective vector $\vec{g}$ with the closest member in the set $X$. The parameter $q$ is usually chosen as two. Figure 2.3 displays an example for the GD and IGD. The GD reflects how far the Pareto-set approximation is from the Pareto-optimal front and thus serves as an indicator for the convergence [28]. The IGD also serves as a measure for the diversity of the Pareto-set approximation by calculating the closest distance of the Pareto-optimal set to the Pareto set approximation [28].
For the application of both metrics, it is required that a sample of the Pareto-optimal set of an optimization problem is known. Problems for which this is the case are only test problems, and using these metrics for real-world problems proves to be rather difficult. However, methods to use the GD and IGD

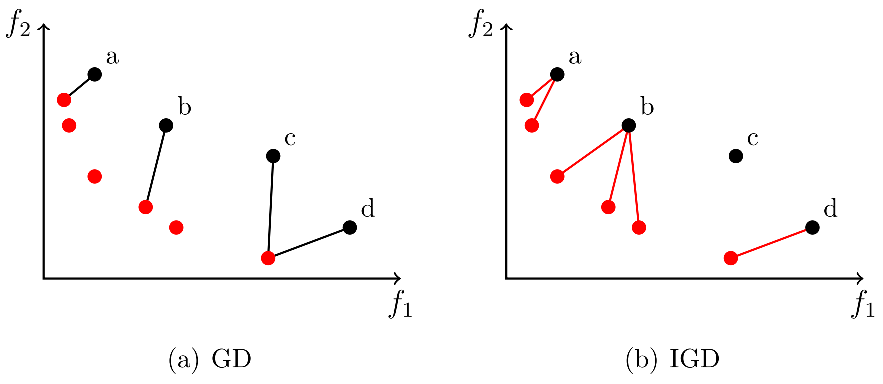


Figure 2.3: The respective Euclidean distances as black and red lines used by GD and IGD. The Pareto-set approximation is the set of points a,b,c,d, and the red points are samples of the Pareto-optimal front.

for problems without a known Pareto-optimal front have been proposed, which estimate the extreme points of the problem and then sample points on a hyperplane to generate a Utopian Pareto-optimal set [29].

## 2.3 Multi-Objective Evolutionary Algorithms

With multi-objective optimization and Pareto-dominance, the number of mutually non-dominated solutions may exceed the population size. Non-dominated solutions are theoretically of equal quality, and one problem that arises is what strategy is applied to select solutions optimally and how *optimal* is defined.
Since all solutions are theoretically equal, the naive solution is randomly chosen. However, several strategies and selection mechanisms for evolutionary algorithms have been contrived in the literature. MOEAs can also be categorized into their approach to handling an optimization problem, such as Pareto dominance-based, decomposition-based, and indicator-based [30].
Section 2.3 describes all MOEAs mentioned in this thesis. It will describe the main aspects and show the major differences to the other algorithms.
Some of the selected algorithms are specially designed for specific problem use-cases. One of that is many-objective optimization, which deals with problems that have four and up to 20 objectives [31, 32]. With many objectives, the problems get more complex to solve and also face the curse of dimensionality,

which means simple Pareto-based approaches do not work well anymore [33]. Another aspect is large-scale multi-objective optimization, which usually relates to problems with a large number of decision variables and is then referred to as many-variable optimization. Many-variable optimization usually handles problems with up to 5000 decision variables or more, with the lower end at about 50 variables [34].
When the term Multi-objective Evolutionary Algorithms (MOEAs) is used, it includes all the mentioned algorithms. In this thesis, the distinction between multi-objective, many-objective, and many-variable is more relevant for classifying optimization problems.

### SPEA2

The Strength Pareto Evolutionary Algorithm 2 (SPEA2) uses a regular population and a continuously updated archive, which is limited in size [35]. SPEA2 gives each individual in the population and the archive a strength value, which is the number of individuals it dominates. Then, a raw fitness value for each individual is calculated, which is the sum of the strength values of all the dominators of a respective individual. Afterward, a density estimation calculation is performed based on the distance to the k-th nearest individual. Finally, the resulting value of the density estimation and the raw fitness are summed to calculate an individual's fitness.
In each algorithm iteration, the non-dominated members (fitness lower than one) of the population and the archive are copied to the successor archive. A truncation operator is applied if the number of individuals in the archive exceeds the set size. The truncation operator iteratively removes the individual with the smallest k-th distance until the set archive size is reached.
Finally, for the mating selection to create the next population, only archive members are considered.

### SPEA2SDE

Li, Yang, and Liu combined dominance and decomposition-based approaches to develop SPEA2SDE, an extension to SPEA2 for many-objective optimization. It uses a shift-based density estimation and focuses not only on the diversity but also the convergence of the population [32].
The density estimation mechanism in SPEA2SDE shifts the position of other

individuals in the population. For a specific individual for which the density is calculated, every other individual that performs better than that individual on an objective will be shifted to the same position on this objective, and then the density is calculated. That means individuals, which initially had a low density and a poor convergence, are assigned a high-density value through the shifting method. That results in the algorithm preferring solutions with low density (good diversity) and good convergence over low density and poor convergence individuals.

### NSGA-II

The Nondominated Sorting Genetic Algorithm II (NSGA-II) was first described by Deb et al. in 2002 [36]. Its selection process creates different groups for the solutions called fronts. The fronts are based on the number of times a solution is dominated by other solutions, called the domination counter. That means the first front contains all non-dominated solutions. The next front contains all solutions that are non-dominated, not considering the solutions in the first front.
The fronts are ranked, and the first $n$ fronts that fit precisely into the population are taken. If the population still has remaining space left, the next front is partially taken. The solutions from this front are selected based on crowding distance. Crowding distance is a density estimation of the solutions surrounding a particular solution. It is calculated based on the absolute distance in the specific objectives of the surrounding solutions. Solutions with greater crowding distances are preferred, which maximizes the diversity of the selection of solutions. Boundary points in the front are always selected first because they get the highest crowding distance assignment.

### NSGA-III

As an extension of NSGA-II, Deb and Jain developed a reference-point-based many-objective algorithm named NSGA-III. NSGA-III emphasizes population members who are non-dominated yet close to a set of supplied reference points. The base algorithm of NSGA-III is the same as in NSGA-II. However, the crowding distance operator is replaced with another approach [31].
The algorithm uses reference points, which are systematically determined and evenly spread on a normalized hyper-plane with (M-1)-dimensions for an M-objective problem. Then, reference lines are constructed by joining the points

with the origin.
The selection mechanism normalizes the objective space, and each population member is associated with the reference point to which reference line they are the closest.
Each reference point gets a niche counter, which is the number of population members from the already selected population associated with this point. The last front members, which do not fit into the population, are then iteratively selected based on the reference points with the smallest niche counts. The niche counts are updated after every newly selected population member.

### MOEA/D

The idea of MOEA/D is to decompose the multi-objective problem into several single-objective problems, which are optimized simultaneously [37]. MOEA/D aims to find different parts of the Pareto-front by solving these single-objective problems. The creation of the single-objective problems is done via scalarization functions, which can be designed in various types. The simplest one is using a weighted sum over the objectives of each individual, for which several different weight vectors are utilized. Other more complex approaches are the Tchebycheff and penalty-based Boundary Intersection approach. Both approaches also use weight vectors and a reference point to calculate the scalarization functions.
The Tchebycheff approach minimizes the maximum value in the resulting vector of the weight vector multiplied with the distance of the reference point and an individual.
The penalty-based Boundary Intersection approach uses two vectors, an ideal vector, and a direction vector, and the method minimizes the distance to both, with the distance to the direction vector penalized. The ideal and direction vectors are both based on the definition of the weight vector.
MOEA/D also defines a neighborhood structure to each weight vector and uses the solutions in this neighborhood to optimize each single-objective problem.

### MOEA/DD

MOEA/DD is an extension of MOEA/D for many-objective optimization, which is also based on decomposition as MOEA/D and additionally also includes a dominance-based mechanism, which is related to NSGA-III [38]. The

basic algorithm is similar to MOEA/D. It also uses weight vectors and scalarization functions (penalty-based Boundary Intersection (PBI)) to divide the problem into subregions and to measure the fitness value of a solution.
The mating selection also uses a neighborhood structure, with the added possibility to select from the whole population for possible parent solutions. Each offspring is then iteratively added to the parent population, and then a solution is deleted based on non-dominated sorting, inspired by NSGA-III. If the fronts do not fit exactly into the population, the algorithm employs density estimation and scalarization functions to select from the last front. The most crowded subregion is chosen, and the worst value based on the scalarization function in that region is deleted. However, to preserve diversity, solutions in an isolated subregion are not deleted, and instead, a solution from the whole population is chosen, based on the most crowded subregion, to be deleted.

### IBEA

The Indicator-Based Evolutionary Algorithm (IBEA) was developed by Zitzler and Künzli to allow the flexible integration of preference information [39]. It is a relatively simple algorithm that assigns each population member a fitness rank using a binary function quality indicator. Based on this indicator, the algorithm calculates the loss in quality if a specific solution is removed from the population. The loss in quality is then used as the fitness of a solution. The solution with the smallest fitness value is removed from the population for the environmental selection, and the fitness ranks are recalculated. That is repeated until the set population size is reached.

### HypE

The HypE algorithm was developed by Bader and Zitzler and is similar to IBEA. Both use an indicator function to calculate the fitness value for the members of the population. However, HypE specifically uses the HV indicator for this task [22].
The HV fitness is calculated by using the aggregate value of the HV contributions of each individual. The calculation also does not require that the members of the population are mutually non-dominating.
The algorithm applies a non-dominated sorting algorithm and ranks solutions into fronts for environmental selection. In the last front, the selection is performed by calculating the expected loss in HV if a solution is removed. The

mating selection is made based on the scalar HV fitness values.
A major contribution of HypE is that it uses Monte Carlo simulations based on sampling points to obtain an estimate of the exact HV values because using the exact calculation is too computationally expensive for problems with more than three objectives. The algorithm uses the exact calculation for less than or exactly three objectives.

### MOMBI-II

MOMBI-II is an indicator algorithm developed by Hernández Gómez and Coello Coello using the R2 indicator, which is described in Section 2.2. It is based on its predecessor, the MOMBI algorithm [40].
MOMBI-II uses an R2 ranking scheme, which works similarly to non-dominated sorting. A specific solution optimizes each utility function, and the solutions are grouped according to which optimize the set of utility functions. This set of solutions gets the best rank, are removed from the population, and the second rank is determined similarly. That is repeated until all the solutions are assigned a rank. When two solutions have the same quality, the Euclidean norm is used instead [40, 41].
For the R2 indicator, MOMBI-II uses the achievement scalarizing function (ASF) metric, which uses weight vectors and reference points, the ideal and nadir point. Compared to MOMBI, MOMBI-II contains an improvement in the calculation of the reference points.

### ThetaDEA

ThetaDEA aims to improve the convergence of the NSGA-III algorithm by using the fitness assignment scheme used in MOEA/D. The algorithm aims to ensure not only convergence but also diversity [42]. ThetaDEA has parallels to the way MOEA/DD works.
At first, the algorithm assigns the solutions to different subregions (called clusters) represented by well-distributed reference points. The algorithm defines a new dominance relation called $\theta$-dominance, which uses the information about the clusters and the PBI values of the solutions. Solutions can only $\theta$-dominate each other when they are located in the same subregion, and for the relation inside the subregion, the PBI is utilized. For the environmental selection, $\theta$-non-dominated sorting is applied. The solutions from the last front are selected randomly. For the mating selection, random selection is applied as well.

### GLMO

GLMO is the only large-scale multi-objective optimization algorithm mentioned in this thesis. It uses the Grouped Linked Polynomial mutation operator. The Grouped Linked Polynomial mutation operator combines the Grouped Polynomial mutation with the Linked Polynomial mutation. The linked Polynomial mutation alters the decision vector based on a particular distribution around the current vector, and which decision variables are mutated is chosen uniformly random. The amount of change is the same, i.e., linked for all decision variables [43].
Grouped Polynomial mutation separates the variables into a predefined number of groups based on decomposition strategies. A group is randomly chosen, and the mutation is then applied to the whole group. Contrary to Linked Polynomial mutation, the amount of change for each variable in a group is not linked. In Grouped Linked Polynomial mutation, the change for all variables in a group is the same.
This new mutation operator can be used in an arbitrary MOEA. Zille et al. used the operator in SMPSO, NSGA-II and NSGA-III. In this thesis GLMO is utilized in combination with NSGA-III.

## 2.4 Benchmark Problems

There are several problem sets available to benchmark MOEAs. Two classical scalable test suites in multi-objective optimization are the DTLZ [44] and WFG [45] optimization problem test suites [46, 47]. The ZDT test suite is another popular test suite [48].
Test problems can be classified in multiple ways, and one vital interest is in the fitness landscape. Interesting characteristics here are the modality and the geometry of the front. The modality can be either unimodal or multimodal. Multimodal problems have several local optima and frequently can also be deceptive when the majority of the search space favors a deceptive optimum. Since multimodal problems have multiple local optima, they are more challenging to solve, and multimodal problems that are also deceptive make this even more difficult by making the global optimum harder to reach [47].
The geometry can be described in various ways. The used problems in this thesis are either linear, concave, degenerate, convex, mixed, or disconnected. Mixed fronts have connected subsets that are convex, concave, or linear.

Table 2.1: Properties of the DTLZ, WFG, and ZDT test problems utilized in this thesis for the experiments.

| Problem | Geometry | Modality |
|---|---|---|
| DTLZ1 | linear | multimodal |
| DTLZ2 | concave | unimodal |
| DTLZ3 | concave | multimodal |
| DTLZ4 | concave | unimodal |
| DTLZ7 | disconnected | multimodal |
| WFG1 | convex | unimodal |
| WFG2 | convex, disconnected | unimodal |
| WFG3 | linear, degenerate | unimodal |
| WFG4 | concave | multimodal |
| WFG5 | concave | unimodal, deceptive |
| WFG6 | concave | unimodal |
| WFG7 | concave | unimodal |
| WFG8 | concave | unimodal |
| WFG9 | concave | multimodal, deceptive |
| ZDT1 | convex | unimodal |
| ZDT2 | concave | unimodal |
| ZDT3 | disconnected | multimodal |
| ZDT4 | convex | multimodal |
| ZDT6 | concave | multimodal |

M-dimensional optimization problems generally have (M-1)-dimensional Pareto-fronts and if the "dimension" of the Pareto-front is smaller than M-1, it is referred to as degenerate [49]. For example, a line in a 3-dimensional space is considered degenerate, while a plane in a 3-dimensional space is not. A disconnected front has several sets that are not connected. If the geometry is disconnected, it can be challenging to ascertain if the front is concave, convex, or mixed as this is not well defined for disconnected sets [47]. However, the separate disconnected sets can be described in their geometry.
The properties of the test problems are listed in Table 2.1. The test problems DTLZ5 and DTLZ6 are excluded from this thesis as their properties are unknown with more than three objectives [47].
The DTLZ and WFG problems are scaleable, concerning the number of ob-

jectives and decision variables. All ZDT problems have two objectives and are not scaleable. The ZDT test suite is far from comprehensive because of its non-scalability and the few numbers of objectives, which is not appropriate for multi-objective optimization and neither many-objective optimization [47]. Most works related to this thesis often rely on the DTLZ and the WFG test problems for these reasons.

# 3 Hyper-Heuristics for Optimization

A heuristic is a guided search for finding approximate solutions to specific optimization problems of which only limited knowledge is available. Meta-heuristics are search methods that utilize higher-level strategies, which aim to escape local optima, to perform robust search operations for a wide variety of optimization problems [3].
Hyper-heuristics are high-level strategies, which aim to select the best heuristic (referred to as low-level heuristic) at different decision points to solve an optimization problem [1]. A hyper-heuristic is essentially a heuristic for choosing heuristics and does not directly operate on the underlying search space of the optimization problem [3].
Burke et al. define a classification of hyper-heuristic using two considerations. The hyper-heuristic search space and the feedback during learning. Regarding the hyper-heuristic search space, hyper-heuristics can select or generate. Hyper-heuristic selection methods choose existing heuristics while hyper-heuristic generation methods generate new heuristics from specific components [3].
The feedback during learning can be categorized into online learning, offline learning, and non-learning. In online learning, the feedback is directly used while the low-level-heuristic solves a problem. Offline learning is the opposite. The feedback is used after the problem is solved, or there are training instances from which knowledge can be gathered. A non-learning hyper-heuristic simply chooses randomly or in a predefined order [3].
In the literature, selection hyper-heuristics are usually divided into two main stages: the heuristic selection, and move acceptance strategy. The heuristic selection decides which heuristic is applied next at a specific decision point. The move acceptance decides if a solution produced by the applied heuristic is accepted or not. That is mainly based on whether the solution quality is improved, stays the same, or worsens and can be deterministic or non-

deterministic [14].
The literature has previously mainly focused on applying hyper-heuristics for combinatorial (non-continuous) optimization problems, such as bin packing, scheduling, routing and timetabling problems [1, 4, 12]. Much less focus has been given to continuous multi-objective problems and even less attention was given to hyper-heuristics using MOEAs [1, 3–5].
In 2011, McClymont and Keedwell were among the first to present a hyper-heuristic for multi-objective problems using the DTLZ test problems. They developed a novel Markov chain hyper-heuristic with reinforcement learning using low-level heuristics based on peturbative local search [15].
The work by Maashi, Özcan, and Kendall is one of the first studies that investigated selection hyper-heuristics for MOEAs [18]. Since then other related works have been published [2, 11, 13, 50, 51].
This chapter aims to give an overview of the most important related works in the current literature about hyper-heuristics using MOEAs. In the end, a short overview of some related works focusing on offline learning for hyper-heuristics is given.

## 3.1 Selection Hyper-heuristic

Maashi, Özcan, and Kendall were one of the first to use a selection hyper-heuristic in the context of Multi-objective Evolutionary Algorithms. Their work is based on previous achievements in the field of hyper-heuristics. However, which were previously exclusively used for optimization encompassing non-continuous optimization problems such as combinatorial problems or without the usage of MOEAs [1, 18].
For the selection strategy Maashi, Özcan, and Kendall employ a ranking scheme based on four performance metrics. They use the algorithm effort (AE), the ratio of non-dominated individuals (RNI), the hypervolume (HV) (see Section 2.2) and the Uniform distribution of the non-dominated population (UD). The AE measures the computation effort of an algorithm, the RNI determines the fraction of non-dominated individuals in the population, and the UD measures the spread of the non-dominated solutions along the Pareto-optimal front. Each available heuristic is ranked in each of these performance metrics. Then, each heuristic is ranked based on how frequently it achieves the best rank, which is the final rank of each heuristic. This final rank is then combined with the RNI value. Finally, this result is weighted and combined

with the number of CPU seconds elapsed since the heuristic was last called. That emphasizes the exploration of heuristics that are rarely selected. The hyper-heuristic then greedily selects the heuristic with the highest value, and all heuristics share the same population.
The hyper-heuristic employs a move acceptance strategy that accepts all solutions irrespective of their quality.
The hyper-heuristic utilizes NSGA-II, SPEA2, and Multi-Objective GA (MOGA) [52] as the low-level heuristics in the selection. The hyper-heuristic is evaluated with experiments using the WFG test suite (WFG1 to WFG9). The results are compared to NSGA-II, SPEA2, and MOGA. Of the nine test problems, the hyper-heuristic produces the best HV in six out of nine instances, with NSGA-II producing better results in the other three test problems [14].
The authors further extended their hyper-heuristic by including non-deterministic move acceptance methods. They employ the great deluge algorithm and the late acceptance method as move acceptance strategies. The great deluge algorithm accepts solutions that perform better and only accepts solutions of a worse quality if they are above a certain threshold. This threshold is increased gradually during the execution of the algorithm. The late acceptance method only accepts solutions of better quality but compares the current solution to a solution in a previous generation [14].
The hyper-heuristic performed best regarding the HV when the great deluge algorithm was used as the move acceptance method.

## 3.2 Reinforcement Learning Hyper-heuristic

Li, Özcan, and John developed a new approach to hyper-heuristics with MOEAs by including a reinforcement learning functionality [13].
In their hyper-heuristic, each MOEA has a certain probability of being chosen next after another specific MOEA. These probabilities are learned during the execution of the hyper-heuristic by a reinforcement learning scheme. The learning is based on the transition between two MOEA, namely the preceding and current MOEA and the performance of the result of the current MOEA. The performance is calculated with the HV, and they use the change in HV before and after applying an MOEA for a linear reward-penalty method.
The transition matrix is then used in conjunction with a selection method. For this hyper-heuristic, $\epsilon$-roulette greedy selection is employed. The selection method first focuses on exploring the transitions in the early stages and then

becomes more and more greedy. In the beginning, it only applies roulette selection based on the transition probabilities for a specific number of iterations. Afterward, it utilizes greedy selection based on the linear increasing probability $\epsilon$. At last, the decision to switch to another MOEA occurs if there is no improvement in the HV compared to the previous iteration below a particular threshold value. The new population of a selected MOEA always replaces the current population.
Additionally, there are two versions of the hyper-heuristic. The first version uses all available MOEAs, while the second version performs a pre-selection mechanism based on the HV and excludes all MOEAs that perform worse than the median MOEA. Both versions then work the same, with the difference that the second version only uses a reduced subset of the originally available MOEAs.
For the hyper-heuristic, they use the low-level heuristics NSGA-II, SPEA2, IBEA. The hyper-heuristic is benchmarked on the test suits WFG and DTLZ and compared to the performance of the included MOEAs. Overall, the hyper-heuristic did not perform best on all problems but, on average, performs better than each of the MOEAs individually. The pre-selection of well-performing MOEAs in the second version of the hyper-heuristic helped significantly to improve the performance.

## 3.3 Collaborative Hyper-heuristic

Fritsche and Pozo described the Hyper-heuristic collaborative Multi-objective Evolutionary Algorithm (HHcMOEA) in which they used a novel method allowing the different MOEAs to collaborate with each other, which is not present in traditional hyper-heuristic approaches [2].
The collaboration was achieved by giving each MOEA its own population. In the hyper-heuristics introduced previously, one population was used for all MOEAs. The new population then usually replaces the current population completely. However, using a single population can destroy either non-dominated solutions of different strategies if applied consecutively [11]. The idea behind using this collaboration method in this hyper-heuristic is that MOEAs use widely different strategies and characteristics to solve a problem. By giving each MOEA its own population, they retain the specific features in the solutions which they are searching for. That is the case with the algorithms MOEA/DD, which uses diversity as its strategy, and NSGA-II, which

uses Pareto-dominance.
The main aspect in this hyper-heuristic is that the MOEAs communicate with each other, called the migration step. In the migration step, the population generated by the currently executed MOEA is sent to the MOEAs in its neighborhood. Fritsche and Pozo used a broadcast neighborhood, i.e., every MOEA is a neighbor of all others. Each MOEA which receives a population from another MOEA merges it with their own population and then filters the result using their own environmental selection.
The selection method in the hyper-heuristic gives each MOEA the same initial probability of being selected and the heuristic selection method then randomly selects an MOEA based on those probabilities. The R2 indicator compares the quality of the current and previous population. If the quality of the current population is improved, the probability of the respective MOEA is increased by a fixed amount, otherwise, it is decreased.
The MOEAs used in their implementation of HHcMOEA are MOEA/D, MOEA/DD, MOMBI-II, NSGA-II, NSGA-III, SPEA2, and ThetaDEA. In their experiments, they also compare two versions of HHcMOEA, with and without the migration step. Both versions and all of the MOEAs were evaluated with 12 optimization problems: DTLZ1, DTLZ2, WFG1, WFG2, WFG3, and their inverted counterparts. They also used four different settings for the number of objectives: 3, 5, 8, and 10.
The results were evaluated using the average HV based on 20 independent runs and compared using a statistical significance test. The result shown that HHcMOEA without the migration step achieved the best average HV in only 1 out of 48 problem instances and was the best or equivalent to the best in 30 problem instances. With the migration step, it achieved the best average HV value for 34 problem instances. In 46 out of 48 problems, it was the best, or equivalent to the best result.
The results show that the collaboration between the MOEAs presents a significant improvement compared to a hyper-heuristic without the collaboration method.

## 3.4 Collaborative Hyper-heuristic for Many-objective Optimization

Similar to the HHcMOEA algorithm described in Section 3.3, Fritsche and Pozo developed the HH-CO algorithm, which is also based on the basic principle that every MOEA has a population and exchanges information between themselves [11]. Contrary to the previous Section 3.3, which focused on multi-objective problems, they developed the algorithm to optimize problems with a high number of objectives (many-objective problems).
The algorithm works almost identical regarding the collaboration of the MOEAs, the migration step. Each MOEA has its own population, and when it is selected for the optimization, it returns a new population, which is shared with the other MOEAs. In this migration step, every MOEA updates its own population based on its own environmental selection. That prevents an MOEA from losing solutions that would qualify for their specific criterion but that another MOEA would not consider.
A crucial difference to the algorithm in Section 3.3 is the credit assignment. For this algorithm, all MOEAs receive a reward at every iteration. The reward is computed using the R2 indicator and compares the population before and after an MOEA is executed. The algorithm then greedily selects the subsequent algorithm with the greatest improvement in the last execution.
The MOEAs used in their implementation are HypeE, MOEA/D, MOEA/DD, MOMBI-II, NSGA-II, NSGA-III, SPEA2, SPEA2SDE, and ThetaDEA. For the evaluation of the algorithm, they compared it to all the MOEAs mentioned above and to the hyper-heuristic presented in Section 3.2, abbreviated with HH-LA. For the comparison, they used the MaF1 to MaF15 optimization problems, with different settings for the number of decision variables. They also used three different settings for the number of objectives: 5, 10, 15.
The results were evaluated using the average HV and the average IGD based on 20 independent runs and compared using a statistical significance test.
The results show that HH-CO achieves the best or equivalent IGD values in 43 out of 45 problem instances. HH-LA achieved the best or statistically equivalent to the best IGD values in only 10 problem instances, and the best value in 1 problem, while HH-CO achieved the best value in 16 problems. The values for the HV show similar results.
Overall, HH-CO performed the best of all compared algorithms, while HH-LA performed even worse than the best MOEA.

Fritsche and Pozo also applied HH-CO to a real-world many-objective problem, the wind turbine design problem [8]. The wind turbine design problem is a constrained real-world many-objective continuous problem and has five objectives, 32 decision variables, and 22 constraints. On the wind turbine design problem, HH-CO achieved the second-best HV, with NSGA-III performing best, but overall the hyper-heuristic did not perform statistically significantly different.

## 3.5 Offline Learning Hyper-heuristic

Offline learning selection hyper-heuristics are a topic that has not received much attention in the literature. The study by Yates and Keedwell is one recent work that examines offline learning in regards to hyper-heuristics [53]. The base of their study is an online hyper-heuristic for combinatorial optimization problems in four different problem domains. The online hyper-heuristic is executed on these problems and can decide between several classes of low-level heuristics to construct sequences. This online learning process is then saved in a database and utilized to learn effective sub-sequences of low-level heuristics statistically offline. The offline learning effectively finds sub-sequences that work well in the same problem domain. However, across different problem domains, offline learning could not generalize well and was examined to be not suitable [53].
The authors further extended this work by not only relying on the statistical connection between heuristics but also examining sub-sequences based on the objective function value they produce on optimization problems [54]. That is done by comparing the logarithmic objective value difference after a sub-sequence has been applied to a problem. Each sub-sequence is then categorized according to that. They show that effective sub-sequences can construct selection hyper-heuristics to provide significant performance improvements.
They have also successfully applied their algorithm to a real-world problem in a subsequent study. The performance improvement from offline learning was statistically significant. Furthermore, they offline learned from a smaller computationally inexpensive problem and successfully transferred the knowledge to a larger, more computationally expensive problem [55].

## 3.6 Parameter Optimization

Parameter optimization generally concerns the optimal setting or configuration of optimization and learning algorithms. This process is known as hyperparameter search or hyperparameter optimization in machine learning. Hyperparameters are the parameters of the machine learning algorithm, while the machine learning algorithm optimizes different parameters in the underlying model. Hyperparameters configure various aspects of the algorithm and substantially affect the resulting model and its performance. Hyperparameter search is also very commonly practiced because the learning process in these methods is influenced heavily by this set of parameters, and they must be set appropriately to maximize the learning result [56].
Hyperparameter optimization is conventionally done manually or by evaluating a set of parameters based on a priori knowledge. In contrast to these labor-intensive methods, automated methods, such as local or exhaustive search algorithms and genetic algorithms, can also be applied [30, 56].
Parameter optimization is also relevant in the field of Multi-objective Evolutionary Algorithms (MOEAs). In MOEAs there are certain parameter configurations that are known to generalize well on a number of different problem domains, while others settings are dependent on the problem domain and some parameters are highly dependent on each other [57–59]. An optimal selection of parameters has also shown to greatly influence the performance of MOEAs [30, 58–61].
If each configuration of an MOEA is considered its own instance of an algorithm, it can be interpreted as finding the MOEA in a set of MOEAs which best optimizes a specific optimization problem. Algorithms that optimize these parameters can then be considered offline hyper-heuristics and hence are related to offline selection hyper-heuristics. Parameter optimization algorithms, or offline hyper-heuristics for MOEAs differ from selection hyper-heuristics in the crucial aspect, that they do not try to find a combination of MOEAs but one single configuration of a MOEA that performs best [30, 57].
A recent example is the parameter optimization of MOEA/D using a genetic algorithm by Pang, Ishibuchi, and Shang [30]. Optimizeable parameters are the population size, the type of scalarization function and sub-parameters for these functions, the type of crossover and mutation operators, and the probabilities of the crossover and mutation operators. A genetic algorithm is used a the optimizer for the offline hyper-heuristic and is configured with a 54-bit

binary string encoding, a population size of 100, and executed for 300 generations. Each algorithm configuration is run for 10.000 function evaluations. In the experiments, the hyper-heuristic was evaluated independently on the three-objective test problems DTLZ1-4 and WFG1-6. The fitness to evaluate the parameter settings is the normalized HV on the resulting population of solutions for the test problems. The experiments obtained better performance results with the optimized parameters for all test problems. The authors further analyzed the ability of the parameters to be scalable concerning the number of objectives for the test problems by increasing them from three to five objectives. Overall, the performance of the optimized parameters decreased and is worse in two test problems compared to the performance of the original MOEA/D without optimized parameters [30].
Based on the work by Pang, Ishibuchi, and Shang two similar studies were conducted using MOEA/D with a solution selection framework, an addition to MOEAs using an external archive to save solutions from which the solutions are selected in the end [60, 61]. A parameter optimization algorithm very similar to [30] was then applied to MOEA/D with and without the addition of the solution selection framework. MOEA/D achieved significantly better results with the solution selection framework and the optimized parameters. The experiments also suggest that the optimal parameters differ substantially when using an external archive [60, 61].

# 4 Proposed Offline Learning Hyper-Heuristic

In the previous Chapter 3, different approaches to constructing hyper-heuristics were introduced. All approaches that used MOEAs as low-level heuristics had in common that their design used online learning as an approach. The offline approaches either did not use MOEAs or are more similarly to parameter optimization i.e., using only one MOEA with different configurations.

Online learning here refers to the procedure that while the hyper-heuristic is running and solving an optimization problem, it has to decide ad hoc which algorithm to execute next. That presents the hyper-heuristic with the difficult task of choosing the best algorithms to solve the problem. As evidently shown by the hyper-heuristics in Chapter 3 they perform well enough to produce better average results than the original MOEAs used.

All the presented algorithms use some kind of reward function for that task, and the selection of the following algorithm is often based on a greedy approach or probabilistic choosing.

The hyper-heuristics assume that a certain algorithm is better than the others by using these reward functions. The algorithm is executed, and afterward, the next algorithm is chosen, which might be the same algorithm as before if it performed well enough. Moreover, at the beginning of the execution of these hyper-heuristics, no knowledge is available about the MOEAs. That will cause the hyper-heuristics to make sub-optimal choices in the beginning. The hyper-heuristics may also make sub-optimal choices during the execution with a certain error percentage by relying on the assumptions made. In the analysis of the HHcMOEA hyper-heuristic on the wind-turbine problem, Fritsche and Pozo have shown that HHcMOEA takes the longest time to converge compared to all the other MOEAs [8]. This behavior will likely get worse the more algorithms are contained in the pool, which makes it worthwhile to investigate offline hyper-heuristics.

There also is the question of whether the reward functions used accurately determine the next best algorithm. Most approaches outlined in the previous chapters compare their hyper-heuristic only to a baseline hyper-heuristic, which decides randomly on the next used algorithm. The different approaches to a reward function also indicate that there may be an enormous amount of possible reward functions, which work well. Mathematically, there are even an infinite amount of reward functions possible. Some reward functions may not even work for certain MOEAs, especially if considering MOEAs designed explicitly for problem instances like many-objective or many-variable problems. Since the user always chooses the reward functions, they may also be biased.
That is why the reward function and the mechanism of choosing the subsequent algorithm present a vulnerability in these hyper-heuristics and a significant point of debate. A possibility to overcome this drawback in these hyper-heuristics is to apply offline learning instead of online learning. In offline learning a reward function is not needed.
The Offline Learning Hyper-Heuristic collaborative Multi-objective Evolutionary Algorithm (OHHMOEA) presents an instance of an offline-learning hyper-heuristic. In this chapter, OHHMOEA is described in its detail. It first begins with an explanation of the theoretical design of the algorithm in Section 4.1. In Section 4.2 the fitness metric used by the hyper-heuristic is introduced, followed by Section 4.3, which describes how the algorithm was implemented in the PlatEMO framework [10].

## 4.1 Algorithm Design

The base design of OHHMOEA is a simple hyper-heuristic algorithm. It is based on the algorithms presented in the Sections 3.3 and 3.4, that is, it also uses the migration step between the MOEAs. Algorithm 2 presents this base algorithm in pseudocode, here named *HHAlgorithm*. The *HHAlgorithm* is designed as an optimization algorithm, which takes the Sequence of MOEAs $\delta$. The sequence of MOEAs $\delta$ is a list of indices with the length $\ell$. Each index in the list corresponds to one specific MOEA in the predefined pool of available MOEAs $\Phi$. As an input, the *HHAlgorithm* also gets an optimization problem $\mathcal{Z}$, which the sequence $\delta$ is executed on. In Lines 2 and Line 3 all the MOEAs are initialized with a population for the optimization problem

**Algorithm 2** HHAlgorithm

```
Input: Optimization Problem $\mathcal{Z}$, Sequence of MOEAs $\delta$
Output: Population of Solutions for $\mathcal{Z}$
 1: $\Phi \leftarrow$ Pool of MOEAs                                ▷ hard-coded value
 2: for each MOEA $\phi \in \Phi$ do
 3:     $\phi$.Population $\leftarrow$ InitializePopulation($\mathcal{Z}$)
 4: for each index $i \in \delta$ do
 5:     $\phi = \Phi$.Get($i$)
 6:     newPop $\leftarrow \phi$.Execute($\mathcal{Z}$, $\phi$.Population)
 7:     for each MOEA $\alpha \in \Phi$ do
 8:         oldPop $\leftarrow \alpha$.Population
 9:         $\alpha$.Population $\alpha$.EnvironmentalSelection(oldPop, newPop)
10: $\phi \leftarrow \Phi$.Get($\delta$.Last)
11: return $\phi$.Population
```

$\mathcal{Z}$, which they retain through the execution of the algorithm. Each initial population is random and distinct from each other.

Afterwards, from Line 4 to Line 9 the Sequence of MOEAs is executed. At first, the specific MOEA $\phi$, which is represented by the integer number $i$, is obtained from the pool. With its current population, the MOEA $\phi$ is executed on the optimization problem $\mathcal{Z}$. The result of that execution is saved in the variable *newPop* and then used to update all populations of the MOEAs in the pool, each with their respective environmental selection. That is the migration step.

When the execution of the sequence is finished, the algorithm returns the population of the last MOEA in $\delta$.

Each MOEA that is executed runs for at least one generation. That means the length $\ell$ of the Sequence $\delta$ is limited by the number of generations for the algorithm, which is usually defined as the maximum number of function evaluations divided by the population size. Usually, a MOEA is given the maximum number of functions it can evaluate until it shall terminate, which can be more than one generation. For simplicity's sake, this mechanism is omitted here. That is further elaborated on in the following Section 4.3.

The main step in OHHMOEA is to learn the optimal Sequence of MOEAs $\delta$ in Algorithm 2 with an offline learning algorithm. This can be thought of as an optimization algorithm, which optimizes another optimization algorithm.

**Algorithm 3** OHHMOEA

```
Input: Optimization Problem $\mathcal{Z}$, Length of Sequence $\ell$, Evaluation Repetitions $\lambda$
Output: Population $\mathcal{P}$ of Sequences
 1: $\mathcal{P} \leftarrow$ Initialization($\ell$)
 2: for each Sequence $\delta \in \mathcal{P}$ do
 3:     $\delta$.Fitness $\leftarrow$ HHFitness($\mathcal{Z}$, $\lambda$, $\delta$)
 4: while stopcondition $\neq$ True do
 5:     MatingPool $\leftarrow$ FitnessSelection($\mathcal{P}$)
 6:     Offspring $\leftarrow$ GeneticOperator(MatingPool)
 7:     for each Sequence $\delta \in$ Offspring do
 8:         $\delta$.Fitness $\leftarrow$ HHFitness($\mathcal{Z}$, $\lambda$, $\delta$)
 9:     $\mathcal{P} \leftarrow$ EnvironmentalSelection($\mathcal{P}$, Offspring)
10: return $\mathcal{P}$
```

That is analogous to parameter optimization as described in Section 3.6. However, the parameter to be tuned here is the sequence of MOEAs $\delta$ that are executed. The design of OHHMOEA is displayed in Algorithm 3 and applies the fitness definition *HHFitness* from Algorithm 4, which is defined in Section 4.2. OHHMOEA takes an optimization problem $\mathcal{Z}$, the length of the Sequence $\ell$, and the number of repetitions $\lambda$ for the *HHFitness* as an input. The population $\mathcal{P}$ of the algorithm consists of individuals, which contain a specific order of MOEAs with the fixed-length $\ell$. Each generated sequence is then evaluated with the fitness metric *HHFitness* as defined by Algorithm 4. The *HHFitness* applies the *HHAlgorithm* (Algorithm 2) with the supplied sequence $\delta$ and returns a singular quality value for the sequence.
The evaluation is applied in Lines 3 and 8. The algorithm design intends the offline learning algorithm to optimize a single-objective problem with each individual having a singular fitness value, which is why Algorithm 3 is based on the Genetic Algorithm in Algorithm 1. However, changing the design to a multi-objective problem would be easily achievable. That would then require a MOEA to the base of the hyper-heuristic optimization.
The usual steps of fitness selection, genetic operation, and environmental selection are applied in Lines 5, 6, and 9. These are the same as in the genetic algorithm in Algorithm 1 in Section 2.3. When the stop condition is reached, the algorithm returns the population of sequences $\mathcal{P}$ with their fitness as a result.

## 4.2 Fitness Metric

One intricacy in the hyper-heuristic algorithm is, that each evaluation of a sequence is in itself an evaluation of an optimization problem with a resulting population of solutions to the given problem $\mathcal{Z}$. The resulting populations to the problem $\mathcal{Z}$ have to be evaluated in some manner to produce a fitness value. This is done by the metric defined by Algorithm 4.
For the evaluation of these populations the quality indicators described in Chapter 2 are utilized. All MOEAs use randomness, which is why each se-

**Algorithm 4** HHFitness

```
Input: Optimization Problem $\mathcal{Z}$, Evaluation Repetitions $\lambda$, Sequence of MOEAs $\delta$
Output: Fitness $\mathcal{F}$ of $\delta$
1: $\mathcal{L} \leftarrow$ List of Fitness Values
2: while $\lambda \neq 0$ do
3:     $\delta$.Population $\leftarrow$ HHAlgorithm($\mathcal{Z}$, $\delta$)
4:     $\mathcal{F} \leftarrow -$HV($\delta$.Population, $\vec{1}$)
5:     if $\mathcal{F} = 0$ then
6:         $\mathcal{F} \leftarrow$ GD($\delta$.Population, $\vec{0}$)
7:     $\mathcal{L}$.append($\mathcal{F}$)
8:     $\lambda \leftarrow \lambda - 1$
9: return Median($\mathcal{L}$)
```

quence is evaluated a fixed number of times denoted by $\lambda$, and then the median result is taken as the fitness. The user sets the number of repetitions. The fitness calculation is also illustrated in Figure 4.1.
At first, the *HHAlgorithm* with the optimization problem $\mathcal{Z}$ and the specific sequence $\delta$ is executed. The resulting population is then evaluated with the normalized hypervolume (HV) indicator, which uses normalized objective values. Larger HV values indicate a better performing population. In OHHMOEA minimization is performed, which is why the negative HV is used. The HV utilizes a reference point for the computation and if a solution does not perform well enough i.e. is outside the reference point, the HV value results in zero. That makes sequences, which perform this poorly, incomparable to each other. Even sequences that perform very poorly should be compared with each other because having a flat fitness landscape is detrimental to the evolutionary process. It could also occur that all solutions in the population have a fitness of zero, resulting in a random search. This is why the fitness is designed to calculate the GD in these cases. Smaller GD values correspond to better solu-

tions, and since the GD values are all positive, individuals using the GD will always be worse in comparison to individuals using the negative HV.
The HV metric used in this thesis normalizes the objective values of the solu-

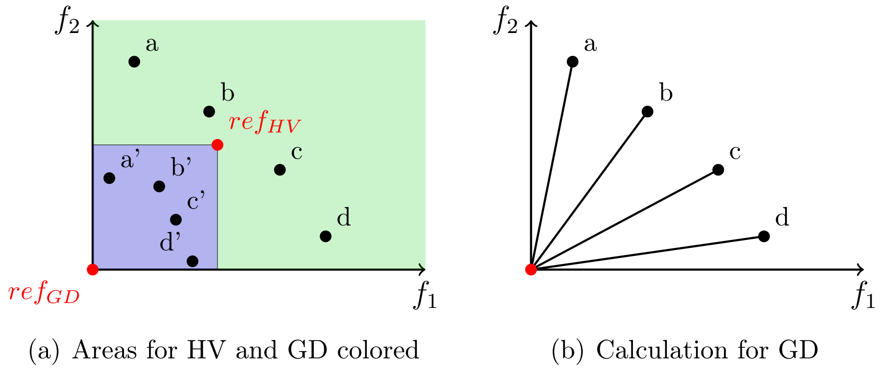


Figure 4.1: Example for the calculation of the fitness for OHHMOEA. In Figure 4.1(a), in the blue area, the HV is calculated, while for populations that are completely outside of this area, the GD is calculated. The area for GD is colored in green. In Figure 4.1(b) an example for the lengths used by the GD indicator is shown.

tions before the calculation and uses a vector of all ones as a reference point. The GD uses a vector of all zeros, i.e., the origin of the objective space, as a single reference point. For GD that essentially means that the resulting value is the average of the lengths of the solution vectors. Since the value of GD is to be minimized, the population will be converging towards the origin. That is until the population produces a HV value that is larger than zero. By using GD the fitness focuses on convergence for poor quality results until it produces acceptable results and then uses the HV to focus on diversity and convergence. In Figure 4.1(b) the calculation of the GD is displayed. In Figure 4.1(a) the calculation of the whole fitness is visualized. For the points a', b', c', and d' in the bluish area the HV is calculated and in the green area for the points a, b, c, and d the GD is calculated.
Using a combination of the HV and the GD works well for this use-case and creates an increasing discontinuous fitness landscape. Figure 4.2 shows the fitness values generated by one single point in the objective space. The figure clearly shows the discontinuity in the fitness landscape, which is, however, not an issue. The fitness landscape will certainly look different with more than one point but likely similar.

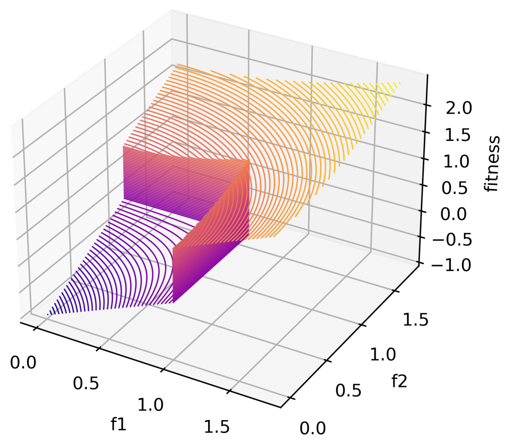


Figure 4.2: Fitness landscape for singular points in the objective space using the point 0.0 as the reference point for GD and the point 1.0 as the reference point for the HV.

## 4.3 Implementation

For the implementation of the algorithm the PlatEMO framework, version 3.2, developed in MATLAB is chosen [10]. It contains more than 150 evolutionary algorithms, more than 300 benchmark problems, and several other useful tools and implementations related to evolutionary multi-objective optimization.
The implementation follows the algorithm design from the previous Section 4.1. However, there are smaller differences or specifics to the PlatEMO framework, which are explained here in further detail.
In Figure 4.3 the hyper-heuristic algorithm as it is implemented in the PlatEMO framework is displayed as a flow-chart. From the algorithm design, it becomes clear that the base of the hyper-heuristic is an optimization problem, which needs to be solved. That is named the *HHProblem* and mainly contains the fitness definition from Algorithm 4. The *HHAlgorithm* is its own MOEA definition inside the PlatEMO framework and based on Algorithm 2. The *HHProblem* is optimized by a Genetic Algorithm. The sequence length $\ell$ of the hyper-heuristic is defined by the number of integer decision variables for the *HHProblem*. The *HHProblem* then runs the *HHAlgorithm* for every sequence in the population of sequences, which it gets as input from the Genetic Algorithm. The resulting population for the *SubProblem* from the *HHAlgorithm* is then evaluated with the *HHFitness* and saved as the objective value of the specific sequence.

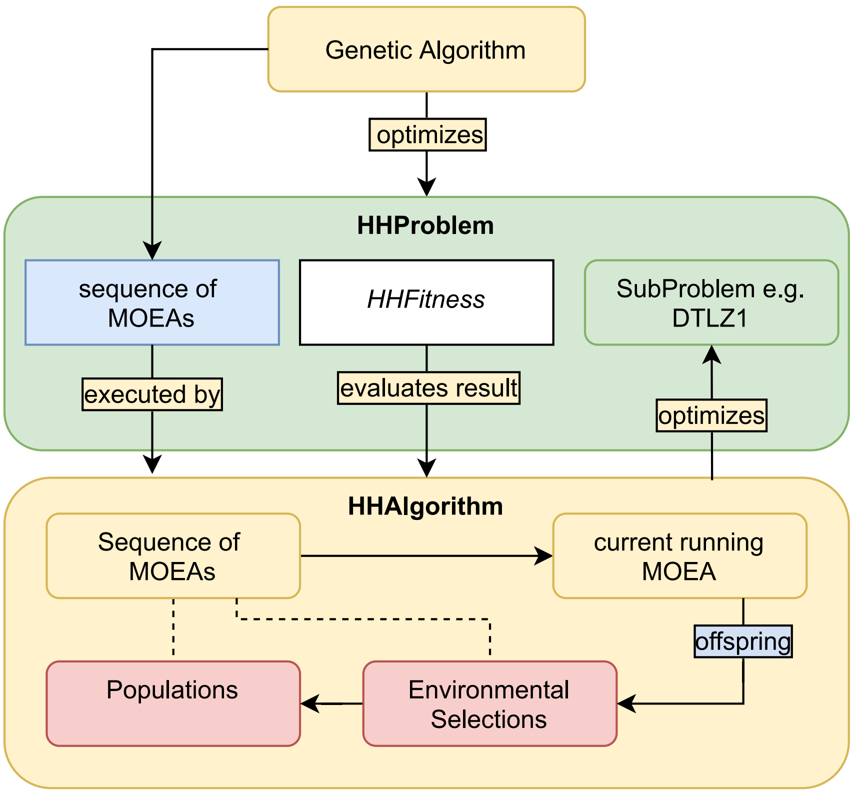


Figure 4.3: Flow-Chart diagram of the hyper-heuristic algorithm.

Theoretically, all MOEAs contained in PlatEMO can be utilized for this hyper-heuristic. Unfortunately, they do not work out of the box, and in order to use them, they need some minor adaptions to their code. The algorithms have to include the migration step after every generation and a separate update function for receiving other offspring and the execution of the environmental selection in the migration step.
In this implementation, it was decided to use ten different MOEAs: GLMO, IBEA, MOEA/D, MOEA/DD, MOMBI-II, NSGA-II, NSGA-III, SPEA2, SPEA2SDE, and theta-DEA. The algorithms use the default parameters from the PlatEMO framework.
The algorithms GLMO, MOEA/D, MOEA/DD, MOMBI-II, NSGA-III, and theta-DEA do not exactly use the set population size. The reason is that these algorithms require uniformly distributed points, and because of the need for uniformity, the population size may be slightly smaller. In order to not give these algorithms a disadvantage because of the smaller population size, the hyper-heuristic calculates the population size necessary to have uniformly

distributed points beforehand and takes this number for the population size of all algorithms.
One aspect of the algorithm that was not further elaborated on is the length of the sequence of MOEAs $\ell$, which is defined by the user. Algorithms in PlatEMO terminate when they reach the maximum function evaluations $\mathcal{FE}_{max}$ set by the user. That is why the hyper-heuristic needs to calculate the maximum function evaluations each algorithm gets ($\mathcal{FE}_{\alpha}$) for its run, which is related to the parameter $\ell$. $\mathcal{FE}_{max}$ is equally divided among all algorithms in the sequence, and $\mathcal{FE}_{\alpha}$ is determined by the formula denoted in Equation 4.1.

$$\mathcal{FE}_{\alpha} = \left\lfloor \frac{\mathcal{FE}_{max}}{|\mathcal{P}| \cdot \ell} \right\rfloor \cdot |\mathcal{P}| \tag{4.1}$$

The variable $\ell$ is the length of the sequence of MOEAs and $|\mathcal{P}|$ is the population size of the *SubProblem* in the hyper-heuristic. The intended effect is, that each algorithm gets exactly as many function evaluations for an integer number of generations and that the number of generations is equally divided. If $|\mathcal{P}| \cdot \ell$ is a divisor of $\mathcal{FE}_{max}$ the equation can be simplified to Equation 4.2.

$$\mathcal{FE}_{\alpha} = \left\lfloor \frac{\mathcal{FE}_{max}}{\ell} \right\rfloor \tag{4.2}$$

However, that $|\mathcal{P}| \cdot \ell$ is a divisor of $\mathcal{FE}_{max}$ can not be guaranteed. Using the Equation 4.2 for instances where $|\mathcal{P}| \cdot \ell$ is not a divisor of $\mathcal{FE}_{max}$ the algorithm would use more function evaluations than it should and the hyper-heuristic will overall use more function evaluations than intended, but at maximum $|\mathcal{P}| \cdot \ell$ more function evaluations. This is because each MOEA in the sequence can at maximum execute one additional generation ($|\mathcal{P}|$ function evaluations) before finishing. Because this is an unwanted behavior the number of function evaluations per algorithm is determined by the formula denoted in Equation 4.1.
Because of the usage of the rounding down function, it may lead the hyper-heuristic to use fewer function evaluations than set by the user, which is why the last MOEA in the sequence uses the remaining function evaluations available. That may still result in the hyper-heuristic using more function evaluations than set by the user, but at maximum, only as much as one generation. However, this is mainly due to the way it is implemented in the underlying PlatEMO framework and not because of the way the hyper-heuristic

is implemented. Each MOEA in the PlatEMO framework shares this behavior.

## 4.4 Discussion

OHHMOEA utilizes a user-defined fixed-length sequence with length $\ell$. Each element in that sequence is one MOEA, which runs for at least one generation. MOEAs are usually not implemented to run for less than one generation, e.g., one half of a generation. The maximum length $\ell$ in the hyper-heuristic is hence also the maximum number of generations for the optimized *SubProblem*. However, the maximum length is not desirable as the search space would get too large to solve the problem in a reasonable time.
The search space of the hyper-heuristic equals the number of available MOEAs to the power of the sequence length. For a population size of 100 and with 10000 maximum function evaluations, the maximum sequence length results in 100. If the maximum number of function evaluations is increased to 50000, the maximum sequence size increases to 500. The search spaces for both values are respectively $10^{100}$ and $10^{500}$, using a pool size of 10 MOEAs. The number of solutions the hyper-heuristic can evaluate in a reasonable time is orders of magnitude lower, usually similar as for the *SubProblem*. That means usually 10000 or 50000 evaluations ($10^4$ and $5 \cdot 10^4$ respectively).
A user-defined length makes it possible to set the length $\ell$ to lower values than the maximum length, which makes the hyper-heuristic more likely to find an approximate optimal solution.
The alternative to a fixed-length is a dynamic-length sequence. However, a dynamic-length sequence would open up the possibility of a sequence with the maximum length, which is not desirable. This would necessitate an upper limit for the dynamic length. It is also not really clear how the hyper-heuristic should behave with dynamic lengths regarding the optimization of the *SubProblem*. A dynamic length would create more issues that need to be solved. Another reason for a fixed-length sequence is the genetic operator crossover. Most crossover methods in literature are designed with fixed-length individuals in mind.

In Section 4.2 the fitness metric for OHHMOEA is presented. The metric calculates the HV, and if the HV value is zero, the GD is calculated instead. Ultimately, the reference points for the calculation of the HV should

be chosen in a way that zero values should not occur, and the reference point has to be identical for all populations in the hyper-heuristic. It would be possible to take the nadir point of all populations and then take the worst one as the reference point for the calculation of all HVs. However, this produces a computational overhead and makes fitness values between hyper-heuristic runs incomparable. Using the GD instead represents a solution to this problem. Furthermore, solutions that have a HV value of zero are generally speaking of such a low quality that they pose no real interest to the user. At least, that is true for how the calculation is implemented in this thesis.
The selection of the fitness metric is also likely subject to debate and is as much biased as the reward functions used in online hyper-heuristics. Here, the HV is mainly chosen because it simultaneously measures convergence and diversity. As described, the main drawback of the HV is that it can result in zero values for solutions of a very low quality. Other indicators such as the R2 indicator might be more suitable. At last, what indicator is used for the hyper-heuristic is also user-dependent. For individual optimization problems, specific properties of the Pareto-set approximation might be of interest, and as such, the metric that is best suited differs in regard to what purpose the optimization has. The modularity of the presented OHHMOEA allows to exchange the fitness calculation with an arbitrary quality indicator easily.

# 5 Evaluation

In this chapter, the proposed Offline Learning Hyper-Heuristic collaborative Multi-objective Evolutionary Algorithm (OHHMOEA) is evaluated. This chapter's main goal is to statistically analyze the resulting sequences of the hyper-heuristic and evaluate the best sequences of the hyper-heuristic on benchmark problems. The following questions are central to this chapter:

1. Do certain sub-sequences of MOEAs occur more often than others?
2. Are the resulting sequences problem-specific?
3. How does the OHHMOEA perform compared to the selected MOEAs?
4. Are the results of the sequences 'stable' if the population size, the function evaluations, the objective size, or decision variables are increased?
5. Are the learned sequences transferable to other problems?
6. Does the offline hyper-heuristic benefit from a reduced algorithm set?

In Section 5.1 the utilized benchmark problems are presented, and their properties are explained. The experimental settings for the test problems and the offline hyper-heuristic are introduced in Section 5.2. The offline selection hyper-heuristic is executed to learn suitable sequences for benchmark problems, and the results are thoroughly evaluated in Section 5.3. In Section 5.3.1 the resulting sequences for the test problems are statistically examined. The performance of the sequences is finally evaluated in Section 5.3.2 and compared to the utilized MOEAs in the hyper-heuristic, namely, GLMO, IBEA, MOEA/D, MOEA/DD, MOMBI-II, NSGA-II, NSGA-III, SPEA2, SPEA2SDE, and theta-DEA. In Section 5.3.3 the scalability of the resulting sequences is evaluated regarding the population size, function evaluations, objective size, and the number of decision variables. Next, it is examined in Section 5.3.4 whether the learned sequences can be transferred to unseen test problems from the DTLZ and WFG benchmark suites. At last, the offline hyper-heuristic is re-evaluated with a reduced subset of MOEAs in Section 5.4.

## 5.1 Benchmark Problems

From the DTLZ and WFG optimization problem test suites, the test problems DTLZ1, DTLZ2, DTLZ3, WFG3, WFG4, WFG5, and WFG6 are used. Additionally, two test problems (ZDT1, ZDT2) from the ZDT test suite are utilized [46, 47].
The problems WFG1 and WFG2 were not chosen for the experiments because the random generation of the points on their Pareto-front takes significantly more computation time than for the other test problems. It takes more time due to the usage of a sorting function in the implementation of this calculation, which is not present in the other problems. That it takes more time is usually not a problem in MOEAs because they have to generate the Pareto-front only once. However, in this thesis, the hyper-heuristic creates each test problem anew for every sequence evaluation and every repetition of a sequence evaluation. On a small scale evaluation of the hyper-heuristic, the WFG1 and WFG2 problems have been shown to take about 60 times longer than the WFG5 and WFG6 problems ($\sim$600 seconds vs. $\sim$10seconds).
The other optimization problems from the test suites are not used because of the experiments' limited time and computation power. For that reason, a smaller subset of the test suites is decided on.
The properties of the test suits are listed in Table 2.1. DTLZ1 is a problem with a linear multimodal Pareto-optimal front. DTLZ2 and DTLZ3 both have a concave Pareto-front but differ in their modality. DTLZ2 has a unimodal front, while DTLZ3 has a multimodal front.
WFG3 has a linear degenerate unimodal Pareto-optimal front, while WFG 4 to 6 have concave Pareto fronts. WFG4 is multimodal, WFG5 deceptive and WFG6 has a unimodal front. The ZDT1 and ZDT2 test problems are unimodal, and the Pareto-front of ZDT1 is convex, and the Pareto-front of ZDT2 is concave.

## 5.2 Experimental Setup

The relevant parameter settings for the hyper-heuristic are shown in Table 5.1. There are settings for the Genetic Algorithm (GA) and the *HHProblem*, the optimization problem the GA is trying to solve. $C_P$ is the probability of crossover, and $M_E$ is the expectation of the number of mutated variables. The parameters $C_P$ and $M_E$ are set to the default values of the PlatEMO framework. The parameters remain unchanged for all test problems. The GA algorithm

Table 5.1: Experiment settings for the OHHMOEA Algorithm and the values they are set on.

| Class | Parameter | Value |
|---|---|---|
| GA | $\|\mathcal{P}\|$ | 100 |
| | $\mathcal{FE}_{max}$ | 10000 |
| | $C_P$ | 1 |
| | $M_E$ | 1 |
| HHProblem | $\|\mathcal{P}\|$ | 36 |
| | $\ell$ | 10 |
| | $\mathcal{FE}_{max}$ | 4032 |
| | $\lambda$ | 3 |

has other crossover and mutation parameters, which are not relevant because they are only used for real encodings, and the encoding for the *HHProblem* is integer. The sequence length $\ell$ is identical to the number of decision variables. The value 10 for $\ell$ was chosen for several reasons. The algorithms need to have a low granularity to apply their specific strategy for searching in the objective space. With this configuration, each algorithm will have about 11 generations. Another reason is the search space size of the sequence, which was already mentioned in Section 4.1. With a sequence length of 10, the search space has a size of $10^{10}$ possible combinations, which results in 10 billion. Compared to the maximum possible combinations the genetic algorithm can evaluate, which is 10000, the search space already seems large. However, evolutionary algorithms are designed to find approximate solutions to such problems. Finally, the number of MOEAs in the pool is 10. Theoretically, each algorithm can be applied at least once with a sequence length of 10.
The population size $|\mathcal{P}|$ and the maximum function evaluations $\mathcal{FE}_{max}$ for GA are arbitrarily chosen but in the range of values that are commonly used in

literature [31, 61–63]. However, for the *HHProblem* smaller values are chosen. Lower values for these parameters result in the hyper-heuristic using less time and resources to evaluate each sequence. In theory the results of the hyper-heuristic should be scaleable in relation to $|\mathcal{P}|$ and $\mathcal{FE}_{max}$. This hypothesis is further examined in the experiments.
At last, $\lambda$ is the number of repetitions for the evaluation of the underlying optimization problem in the *HHProblem*. The value of $\lambda$ is a multiplicative factor influencing the computation time of the hyper-heuristic. It was set to the lowest number acceptable. An odd number is chosen to have the median as a member of the set. Higher values for $\lambda$ such as five or seven are likely preferable. To accurately reflect the distribution, $\lambda > 10$ is likely necessary as a minimum [64]. Fortunately, very precise numbers are not needed in evolutionary algorithms, and approximations of the true values suffice [6].
With this configuration, the hyper-heuristic performs 10000 evaluations of sequences, with each being an optimization of the *HHProblem*, which itself solves another underlying optimization problem with the configuration. The overall number of evaluations can be determined by multiplying $\mathcal{FE}_{max}$ of both GA and the *HHProblem* and then multiplying the result with $\lambda$. That results in 120960000 function evaluations by the hyper-heuristic, and 10000 evaluations of the GA for each population.
The measured run time of preliminary experiments with the introduced configuration was one day on average. Larger settings would result in longer run time and therefore go beyond the scope of this thesis.
Initially, the population size of the *HHProblem* was set to be 40. However, as mentioned earlier in Section 4.3, the algorithms require uniformly distributed points. That is why internally, the hyper-heuristic reduces the population size to 36. Which means the first nine algorithms in the sequence are applied for 396 function evaluations or 11 generations (See Section 4.3), and the last algorithm in the sequence is applied for 436 function evaluations (12.111 generations). Since the number of function evaluations of the last algorithm is not dividable by the population size without remainder, the last algorithm is applied for an additional generation, resulting in 13 generations. That results in the hyper-heuristic using 4032 function evaluations instead of the planned 4000. Optimally, these values should be selected so that each algorithm is executed for the same number of generations.
There are problem-dependent settings like the number of decision variables for the test problems, which change based on what problem is being currently

run. These are shown in Table 5.2. Based on preliminary experiments, the number of decision variables are chosen either higher or lower than the default parameters to enhance the hyper-heuristic evaluation and to produce usable results. Because of the number of decision variables, some of these problems can

Table 5.2: Experiment settings for the test problems.

| Problem | Objectives | Decision Variables |
|---|---|---|
| DTLZ1 | 3 | 5 |
| DTLZ2 | 3 | 40 |
| DTLZ3 | 3 | 5 |
| WFG3 | 3 | 50 |
| WFG4 | 3 | 50 |
| WFG5 | 3 | 12 |
| WFG6 | 3 | 12 |
| ZDT1 | 2 | 30 |
| ZDT2 | 2 | 30 |

be considered many-variable problems. That is the case for DTLZ2, WFG3, WFG4, ZDT1, and ZDT2.

## 5.3 Offline Learning Result

In this Section, the offline learning experiments on OHHMOEA are analyzed. This chapter aims to answer the first two questions outlined at the beginning of this chapter. These are:

1. Do certain sub-sequences of MOEAs occur more often than others?
2. Are the resulting sequences problem-specific?

With the configuration from Section 5.2, the hyper-heuristic is executed 21 times for each of the selected test problems. The experiments were executed on a remote server with 40 CPU cores. However, only 20 cores were utilized for the duration of the experiments. The experiments took several weeks to finish.

The time of the hyper-heuristic to optimize the sequence for a single test problem is visualized in Figure 5.1. DTLZ1 takes the most time of all the problems. DTLZ2, DTLZ3, and WFG3 take a similar amount of time, about 20 to 24 hours. The problems WFG4, WFG5, ZDT1, and ZDT2 took noticeably less

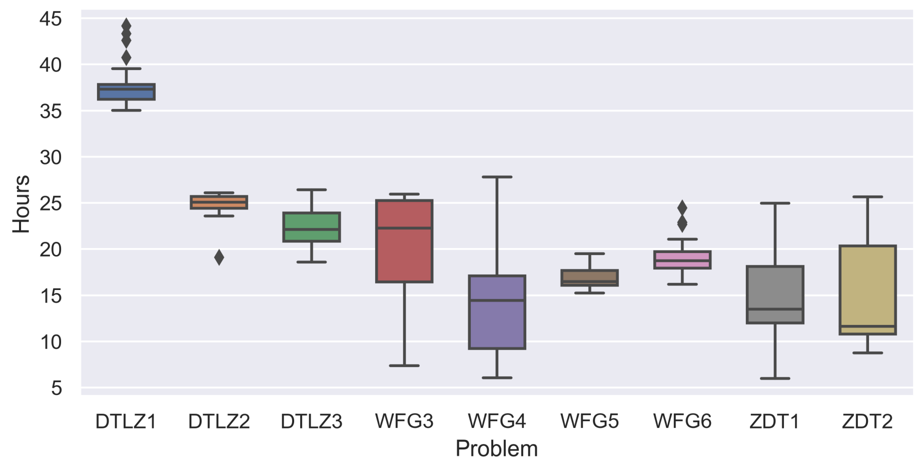


Figure 5.1: Boxplot of the run times of the hyper-heuristic for each test problem.

time than the other problems. As mentioned in Section 5.1, one factor that influences the time of the hyper-heuristic is the generation of the points on the Pareto-optimal front. Another factor related to the test problems is calculating the population's objective values for each solution. This was also explained in Section 5.1 and is the reason why some test problems are not included.

The selected MOEAs for the sequences of the hyper-heuristic exerts an influence on the run time. The two slowest algorithms in the hyper-heuristic are MOEA/D and MOEA/DD. MOEA/D is measured to take about 2 seconds on average to solve the complete *SubProblem* on its own, and MOEA/DD is measured at about 5 seconds. In comparison, the two fastest algorithms (MOMBI-II and NSGA-II) take about 0.3 seconds. The other algorithms are measured at about 0.4 to 0.5 seconds, while SPEA2SDE takes a little longer with an average of 0.9 seconds.

For each test problem, the best solution of the population from each of the 21 independent runs is selected. From these 21 selected solutions for each test problem the median solution is selected. Furthermore, the respective IQR values are calculated. The values are displayed in Table 5.3. The values are calculated with the fitness metric of the hyper-heuristic, which is determined via the HV and the GD indicators. Because all values are negative, they are identical to the negative HV values. The IQR values can indicate how much the population in the hyper-heuristic is converged, with lower values meaning more convergence, i.e., the population is less diverse. However, the IQR values

Table 5.3: For each test problem the median fitness result of the best solutions of the 21 runs and the resulting IQR values. The problems in the table are sorted by their IQR values.

| Problem | Median | IQR |
|---|---|---|
| ZDT1 | -7.125e-01 | 2.129e-04 |
| ZDT2 | -4.365e-01 | 2.668e-04 |
| DTLZ2 | -5.247e-01 | 6.834e-04 |
| WFG5 | -4.847e-01 | 8.489e-04 |
| DTLZ1 | -8.108e-01 | 9.388e-04 |
| DTLZ3 | -5.285e-01 | 1.653e-03 |
| WFG4 | -4.791e-01 | 2.128e-03 |
| WFG6 | -4.935e-01 | 2.717e-03 |
| WFG3 | -3.449e-01 | 4.255e-03 |

can also give an insight into the robustness of the sequence, with lower values associated with more robustness. That is why Table 5.3 is sorted according to the IQR values. The ZDT1 problem has the lowest IQR, and the WFG3 problem has the highest IQR.
The absolute median fitness values do not indicate much if compared between test problems, as the lowest HV possible can vary significantly between problems. However, a lower fitness generally can be interpreted as more convergence and diversity regarding the Pareto-optimal front. The best fitness values are achieved in the DTLZ1 and ZDT1 problems and the worst in the WFG3 and ZDT2 problems.
The offline learning of the hyper-heuristic on the test problems DTLZ1, DTLZ3, WFG6, and ZDT1 is additionally displayed with the respective convergence plot in Figure 5.2. The data is based on the population's fitness after every 1000 evaluations. These figures show that DTLZ3, and ZDT1 show a good convergence behavior, with the relatively monotonous fitness curve in the last generations. The same can not be stated about the test problems DTLZ1 and WFG3. They likely will need more evaluations for convergence, and their results will likely have room for improvement. Additionally, WFG3 has the highest IQR of all the test problems. The issue for this problem could be that the Pareto-optimal front is degenerate, and a high number of decision variables (50) was chosen for this problem (see Table 5.2). In Figure 5.2, DTLZ3 has very good convergence behavior. However, the resulting IQR value is higher than the other test problems shown.

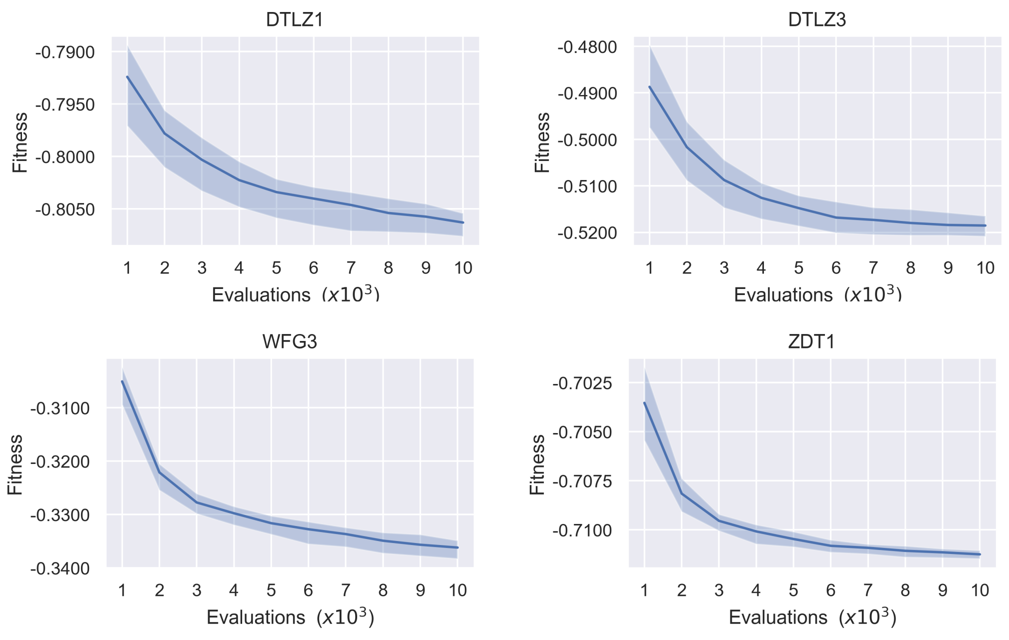
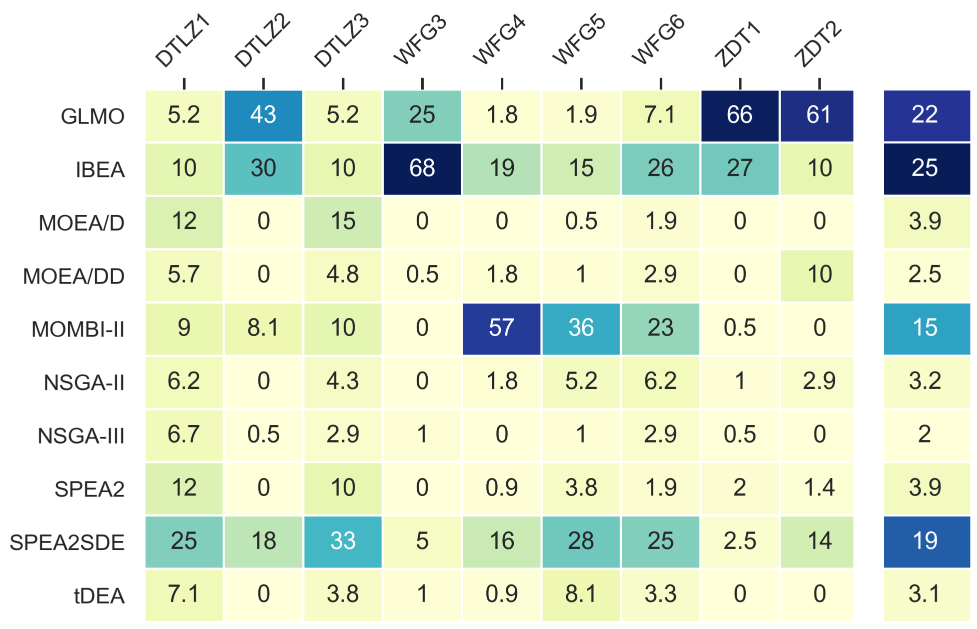


Figure 5.2: Convergence plots for the hyper-heuristic on each of the displayed test problems. The blue line shows the median solution in each generation, and the blue area is the (IQR) range between the 25th and 75th percentile.

## 5.3.1 Sequence Analysis

Each of the 21 independent runs executed on each test problem will result in a population of 100 sequences of length $\ell$. The length $\ell$ is identical to the number of decision variables of the *HHProblem* and was set to 10. Each population contains a sequence with the best fitness. That sequence is selected for each population and further analyzed. The resulting analysis is shown in Figure 5.3. It displays a heatmap showing the percentage of each MOEA that was executed on a specific test problem. The outer right column shows the usage percentages of the MOEAs among all test problems.
First, it shows that specific test problems invoke a preference for specific MOEAs. The GLMO algorithm was most often used for DTLZ2, WFG3, ZDT1, and ZDT2. GLMO is developed for large-scale problems, problems with a high number of decision variables, which are also named many-variable problems [34]. All of the mentioned problems have considerably more decision variables than the other problems and a disproportion between the number of objectives and decision variables (See Table 5.2). Surprisingly, WFG4 also has

many decision variables but very rarely utilizes GLMO. Another observation

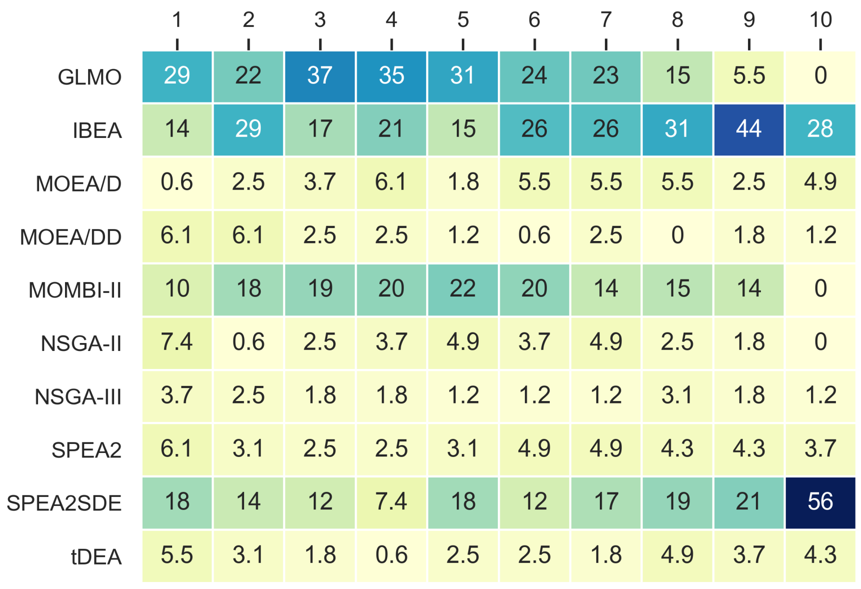


| | DTLZ1 | DTLZ2 | DTLZ3 | WFG3 | WFG4 | WFG5 | WFG6 | ZDT1 | ZDT2 | |
|---|---|---|---|---|---|---|---|---|---|---|
| GLMO | 5.2 | 43 | 5.2 | 25 | 1.8 | 1.9 | 7.1 | 66 | 61 | 22 |
| IBEA | 10 | 30 | 10 | 68 | 19 | 15 | 26 | 27 | 10 | 25 |
| MOEA/D | 12 | 0 | 15 | 0 | 0 | 0.5 | 1.9 | 0 | 0 | 3.9 |
| MOEA/DD | 5.7 | 0 | 4.8 | 0.5 | 1.8 | 1 | 2.9 | 0 | 10 | 2.5 |
| MOMBI-II | 9 | 8.1 | 10 | 0 | 57 | 36 | 23 | 0.5 | 0 | 15 |
| NSGA-II | 6.2 | 0 | 4.3 | 0 | 1.8 | 5.2 | 6.2 | 1 | 2.9 | 3.2 |
| NSGA-III | 6.7 | 0.5 | 2.9 | 1 | 0 | 1 | 2.9 | 0.5 | 0 | 2 |
| SPEA2 | 12 | 0 | 10 | 0 | 0.9 | 3.8 | 1.9 | 2 | 1.4 | 3.9 |
| SPEA2SDE | 25 | 18 | 33 | 5 | 16 | 28 | 25 | 2.5 | 14 | 19 |
| tDEA | 7.1 | 0 | 3.8 | 1 | 0.9 | 8.1 | 3.3 | 0 | 0 | 3.1 |

Figure 5.3: The Heatmap is showing the MOEAs together with the test problems. Each box displays the percentage (rounded down to 1 decimal) of the specific MOEA used on a test problem. The percentage is calculated based on the frequency of the MOEA divided by the total number of occurrences in the column. The right column shows the absolute percentages among all test problems.

that can be made is that MOMBI-II was very frequently used for the problems WFG4, WFG5, and WFG6. It is unclear why exactly that is. MOMBI-II is an indicator algorithm using the R2-indicator with achievement scalarizing functions as utility functions [40]. It could signalize that this strategy is beneficial for these problems.
Some test problems in the selection do not inhibit an obvious picture of which algorithms they prefer the most. That is especially visible in DTLZ1, with the most used algorithm only preferred for 25%. Another test problem with similar behavior is DTLZ3. All other test problems either have one algorithm with a majority preference or a small group of algorithms they prefer the most. For DTLZ1 and DTLZ3, the only preference is SPEA2SDE, and the distribution over the other algorithms seems somewhat random. The reason in the case of DTLZ1 could be that the problem is too easy to solve. In Table 5.3 DTLZ1

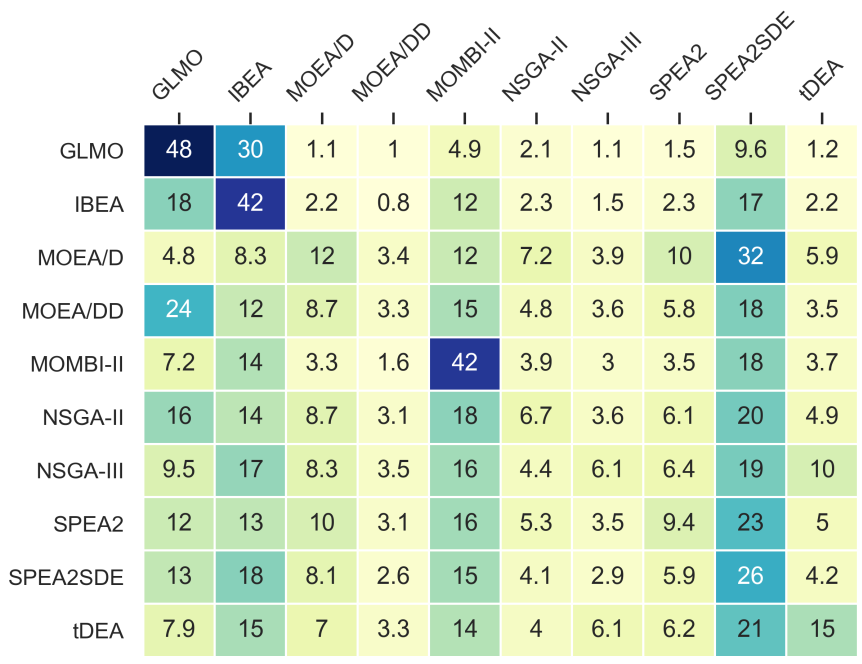
has achieved the highest HV among all test problems. This will result in the selection pressure to fade, and the selection of algorithms being irrelevant because all of them work relatively equally well. The test problems likely prefer SPEA2SDE because it boosts the diversity and convergence of the population. To some extent, the same is likely true for DTLZ3. Both test problems also have the lowest number of decision variables.
IBEA is an indicator algorithm and is frequently utilized among a diverse set of test problems. That is likely because the indicator used in the algorithm is based on the HV concept [39]. Since the fitness in the hyper-heuristic is based on the HV, selecting IBEA will help in improving the fitness. Another algorithm frequently used is SPEA2SDE, which follows a dominance and decomposition-based approach with a focus on diversity and convergence [32]. The HV indicator also measures convergence and diversity, which is likely why SPEA2SDE is preferred here over other algorithms.
Another interesting property of these sequences is, that it is possible to examine the position at which a MOEA was most frequently used. This is displayed in Figure 5.4. In this Figure, the numbers in each column represent the percentage for each MOEA to be in this position, compared to the other algorithms. In each column the percentages sum up to 100. Over 50 percent of all used



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| GLMO | 29 | 22 | 37 | 35 | 31 | 24 | 23 | 15 | 5.5 | 0 |
| IBEA | 14 | 29 | 17 | 21 | 15 | 26 | 26 | 31 | 44 | 28 |
| MOEA/D | 0.6 | 2.5 | 3.7 | 6.1 | 1.8 | 5.5 | 5.5 | 5.5 | 2.5 | 4.9 |
| MOEA/DD | 6.1 | 6.1 | 2.5 | 2.5 | 1.2 | 0.6 | 2.5 | 0 | 1.8 | 1.2 |
| MOMBI-II | 10 | 18 | 19 | 20 | 22 | 20 | 14 | 15 | 14 | 0 |
| NSGA-II | 7.4 | 0.6 | 2.5 | 3.7 | 4.9 | 3.7 | 4.9 | 2.5 | 1.8 | 0 |
| NSGA-III | 3.7 | 2.5 | 1.8 | 1.8 | 1.2 | 1.2 | 1.2 | 3.1 | 1.8 | 1.2 |
| SPEA2 | 6.1 | 3.1 | 2.5 | 2.5 | 3.1 | 4.9 | 4.9 | 4.3 | 4.3 | 3.7 |
| SPEA2SDE | 18 | 14 | 12 | 7.4 | 18 | 12 | 17 | 19 | 21 | 56 |
| tDEA | 5.5 | 3.1 | 1.8 | 0.6 | 2.5 | 2.5 | 1.8 | 4.9 | 3.7 | 4.3 |

Figure 5.4: Heatmap showing the MOEAs together with the positions in the sequence. Each box displays the percentage (rounded down to 1 decimal) of the specific MOEA used on a position in the sequence.

MOEAs at the last position are SPEA2SDE, which is not surprising as using this algorithm at last helps in increasing diversity in an already converged population. The same is true for IBEA. Both algorithms were frequently used at the last place or the second to last.
Finally, knowledge can be gathered about which algorithms work well together based on their frequency of switching to one another. That is shown in Figure 5.5. Some algorithms switch most often to themselves. That is the case

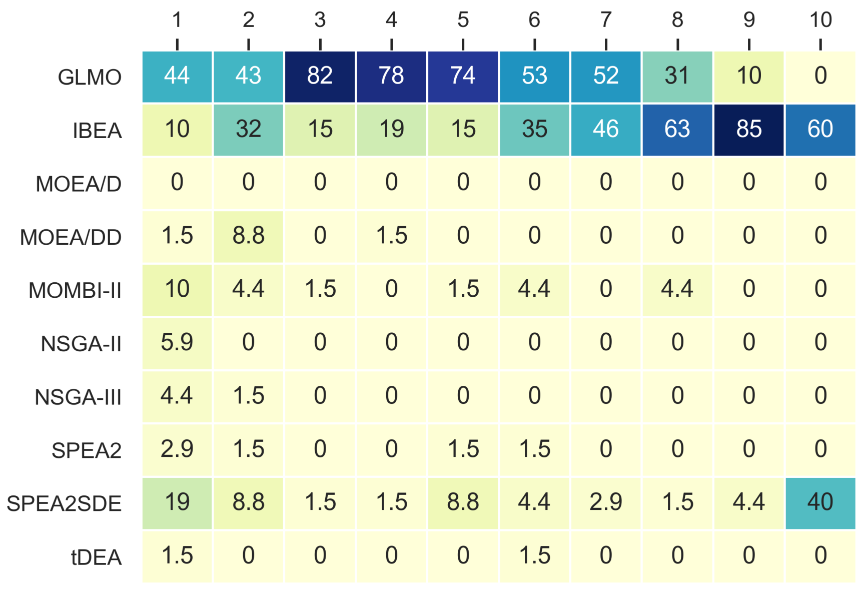


| | GLMO | IBEA | MOEA/D | MOEA/DD | MOMBI-II | NSGA-II | NSGA-III | SPEA2 | SPEA2SDE | tDEA |
|---|---|---|---|---|---|---|---|---|---|---|
| GLMO | 48 | 30 | 1.1 | 1 | 4.9 | 2.1 | 1.1 | 1.5 | 9.6 | 1.2 |
| IBEA | 18 | 42 | 2.2 | 0.8 | 12 | 2.3 | 1.5 | 2.3 | 17 | 2.2 |
| MOEA/D | 4.8 | 8.3 | 12 | 3.4 | 12 | 7.2 | 3.9 | 10 | 32 | 5.9 |
| MOEA/DD | 24 | 12 | 8.7 | 3.3 | 15 | 4.8 | 3.6 | 5.8 | 18 | 3.5 |
| MOMBI-II | 7.2 | 14 | 3.3 | 1.6 | 42 | 3.9 | 3 | 3.5 | 18 | 3.7 |
| NSGA-II | 16 | 14 | 8.7 | 3.1 | 18 | 6.7 | 3.6 | 6.1 | 20 | 4.9 |
| NSGA-III | 9.5 | 17 | 8.3 | 3.5 | 16 | 4.4 | 6.1 | 6.4 | 19 | 10 |
| SPEA2 | 12 | 13 | 10 | 3.1 | 16 | 5.3 | 3.5 | 9.4 | 23 | 5 |
| SPEA2SDE | 13 | 18 | 8.1 | 2.6 | 15 | 4.1 | 2.9 | 5.9 | 26 | 4.2 |
| tDEA | 7.9 | 15 | 7 | 3.3 | 14 | 4 | 6.1 | 6.2 | 21 | 15 |

Figure 5.5: The Heatmap shows the pair-wise frequencies of MOEAs. The values are normalized row-wise. Each box displays the percentage (rounded down to 1 decimal) for a MOEA on the y-Axis to transition to a MOEA on the x-Axis.

for GLMO, IBEA, and MOMBI-II. That is likely because they are the best performing algorithms in the selection. In this Figure, clear preferences are visible for the algorithms GLMO, IBEA, MOMBI-II, and SPEA2SDE. All other algorithms most frequently switch to either of these four. That is no surprise as they are the MOEAs, which seem to work best on the selected test problems. A fascinating observation is that GLMO and IBEA seem to work very well together. Both algorithms very frequently switch to one another. As previously observed, GLMO is most often used for problems with many decision variables. From Figure 5.3 it is visible that whenever GLMO is used

for a problem, IBEA is used as well. That behavior is also reflected in this Figure. GLMO seems to work well for convergence, but because it internally uses NSGA-III, it seems to lack in producing a diverse set of solutions. This drawback is overcome by selecting IBEA at the last or second to last places in the sequence, which is visible in Figure 5.4.
Because of these observations and the preference of the test problems, it can be assumed that there are three clusters in the test problems, which can be categorized based on their algorithm preference. The first group is DTLZ1 and DTLZ3. The second group, which contains many-variable problems, is DTLZ2, WFG3, ZDT1, and ZDT2. The last group is WFG4, WFG5, and WFG6. Other problems which are similar to these groups likely illicit similar sequence frequencies.
It is possible to construct best performing sequences for the above groups if

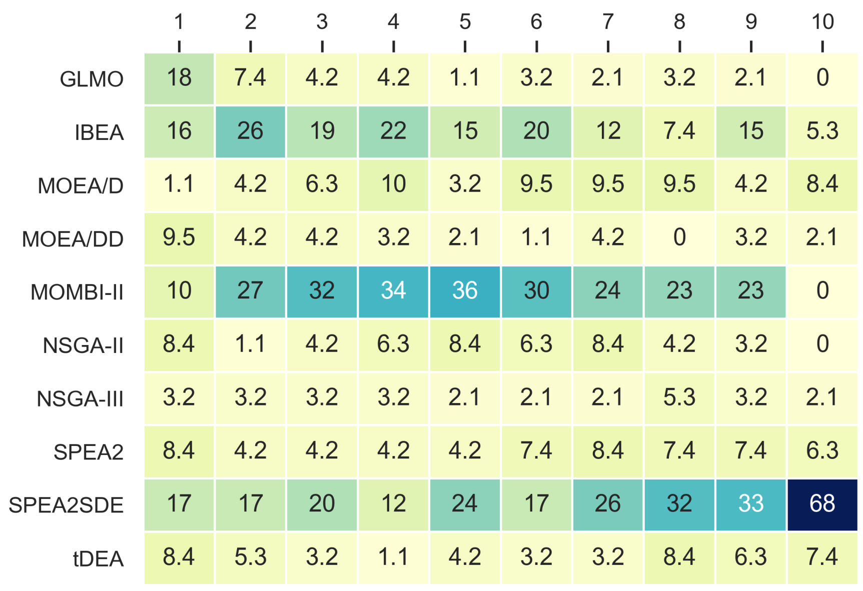


| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| GLMO | 44 | 43 | 82 | 78 | 74 | 53 | 52 | 31 | 10 | 0 |
| IBEA | 10 | 32 | 15 | 19 | 15 | 35 | 46 | 63 | 85 | 60 |
| MOEA/D | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MOEA/DD | 1.5 | 8.8 | 0 | 1.5 | 0 | 0 | 0 | 0 | 0 | 0 |
| MOMBI-II | 10 | 4.4 | 1.5 | 0 | 1.5 | 4.4 | 0 | 4.4 | 0 | 0 |
| NSGA-II | 5.9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| NSGA-III | 4.4 | 1.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SPEA2 | 2.9 | 1.5 | 0 | 0 | 1.5 | 1.5 | 0 | 0 | 0 | 0 |
| SPEA2SDE | 19 | 8.8 | 1.5 | 1.5 | 8.8 | 4.4 | 2.9 | 1.5 | 4.4 | 40 |
| tDEA | 1.5 | 0 | 0 | 0 | 0 | 1.5 | 0 | 0 | 0 | 0 |

Figure 5.6: The Heatmap is showing the MOEAs together with the positions in the sequence. Each box displays the percentage (rounded down to 1 decimal) of the specific MOEA used on a position in the sequence. Only for the problems DTLZ2, WFG3, ZDT1, and ZDT2.

the frequency for the positions is analyzed separately. For the many-variable problems the percentages for each position are displayed in Figure 5.6. In this Figure, the numbers in each column represent the percentage for each MOEA to be in this position. The percentages in each column sum up to 100. From

this heatmap, the MOEA to be chosen for each position is decided greedily based on the percentages. That selection results in GLMO utilizing the positions 1 to 7 and IBEA using the positions 8 to 10.
For simplicity's sake, the remaining two groups were merged into one group, which now contains the following problems: DTLZ1, DTLZ3, WFG4, WFG5, WFG6. The result is displayed in Figure 5.7. The same strategy for the other



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| GLMO | 18 | 7.4 | 4.2 | 4.2 | 1.1 | 3.2 | 2.1 | 3.2 | 2.1 | 0 |
| IBEA | 16 | 26 | 19 | 22 | 15 | 20 | 12 | 7.4 | 15 | 5.3 |
| MOEA/D | 1.1 | 4.2 | 6.3 | 10 | 3.2 | 9.5 | 9.5 | 9.5 | 4.2 | 8.4 |
| MOEA/DD | 9.5 | 4.2 | 4.2 | 3.2 | 2.1 | 1.1 | 4.2 | 0 | 3.2 | 2.1 |
| MOMBI-II | 10 | 27 | 32 | 34 | 36 | 30 | 24 | 23 | 23 | 0 |
| NSGA-II | 8.4 | 1.1 | 4.2 | 6.3 | 8.4 | 6.3 | 8.4 | 4.2 | 3.2 | 0 |
| NSGA-III | 3.2 | 3.2 | 3.2 | 3.2 | 2.1 | 2.1 | 2.1 | 5.3 | 3.2 | 2.1 |
| SPEA2 | 8.4 | 4.2 | 4.2 | 4.2 | 4.2 | 7.4 | 8.4 | 7.4 | 7.4 | 6.3 |
| SPEA2SDE | 17 | 17 | 20 | 12 | 24 | 17 | 26 | 32 | 33 | 68 |
| tDEA | 8.4 | 5.3 | 3.2 | 1.1 | 4.2 | 3.2 | 3.2 | 8.4 | 6.3 | 7.4 |

Figure 5.7: Heatmap showing the MOEAs together with the positions in the sequence. Each box displays the percentage (rounded down to 1 decimal) of the specific MOEA used on a position in the sequence. Only for the problems DTLZ1, DTLZ3, WFG4, WFG5, and WFG6.

group is applied, and the resulting sequence is: GLMO first, then from 2 to 6 MOMBI-II, and then from 7 to 10 SPEA2SDE. These sequences will be evaluated in Section 5.3.4.
Furthermore, it would be possible to initialize the population of the hyper-heuristic based on these learned percentages. Currently, the sequences are initialized randomly. Instead, the percentages could be handled as probabilities for each position. Additionally, the statistics from Figure 5.3 and 5.4 show that some algorithms are used very infrequently, which indicates that some algorithms are not of use to the hyper-heuristic. The authors of [13] used a reduced subset for their hyper-heuristic based on preliminary experiments. Overall, their hyper-heuristic performed better with the reduced sub-

set. Therefore, the experiments conducted in this section are repeated with a smaller selection of MOEAs in Section 5.4. Producing this reduced subset of MOEAs does not have to come from experiments of the hyper-heuristic. Li, Özcan, and John build the pre-selection of MOEAs by running direct experiments for each algorithm on the test problems [13]. In light of the time needed to run experiments directly in the hyper-heuristic, this could save time. Using a reduced subset has the potential to significantly boost the hyper-heuristic's convergence and run time.

## 5.3.2 Performance Analysis

The goal of this section is to answer the third question, how the performance of the learned sequences of OHHMOEA compare to the used MOEAs.
From each of the 21 runs on each test problem a representative with the best fitness is selected. From these 21 representatives the solution with the median fitness is further selected as the final sequence for the evaluation. The sequences are listed in Table 5.4. The sequences are evaluated on the respective

Table 5.4: The sequences used for each respective test problem. The numbers are the indices of the MOEAs: (1) GLMO, (2) IBEA, (3) MOEA/D, (4) MOEA/DD, (5) MOMBI-II, (6) NSGA-II, (7) NSGA-III, (8) SPEA2, (9) SPEA2SDE, (10) tDEA

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| DTLZ1 | 10 | 10 | 5 | 6 | 9 | 3 | 3 | 9 | 9 | 3 |
| DTLZ2 | 9 | 1 | 1 | 1 | 9 | 1 | 1 | 2 | 2 | 9 |
| DTLZ3 | 5 | 5 | 10 | 5 | 2 | 3 | 9 | 3 | 4 | 9 |
| WFG3 | 1 | 9 | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 2 |
| WFG4 | 4 | 5 | 5 | 2 | 5 | 5 | 5 | 5 | 5 | 9 |
| WFG5 | 6 | 5 | 5 | 5 | 10 | 5 | 5 | 9 | 2 | 9 |
| WFG6 | 3 | 2 | 1 | 5 | 5 | 9 | 2 | 5 | 9 | 2 |
| ZDT1 | 9 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 2 |
| ZDT2 | 1 | 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 9 |

nine test problems for 31 independent runs, and the HV, GD, and IGD values are calculated. The maximum function evaluations and the population size are identical to the offline hyper-heuristic OHHMOEA. The Mann–Whitney U rank test is used to determine if the differences in the distribution of the results are significant. All tests are conducted with a p-value of 0.01 and pair-wise with the hyper-heuristic.

Table 5.5: Median sequence of the offline selection hyper-heuristic evaluated with the HV, GD and IGD in the benchmark problems with the settings of the OHHMOEA. Values in bold indicate the best value. Values that have a colored cell indicate no significant statistical difference to the hyper-heuristic method OHHMOEA.

| | | GLMO | IBEA | MOEA/D | MOEA/DD | MOMBI-II | NSGA-II | NSGA-III | SPEA2 | SPEA2SDE | tDEA | OHHMOEA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HV | DTLZ1 | .6217 | .1862 | .3345 | .2530 | .6576 | .4887 | .5616 | .7451 | **.7693** | .7339 | .4437 |
| | DTLZ2 | .4876 | .4830 | .3128 | .3004 | .4708 | .3257 | .3593 | .3643 | .4776 | .3801 | **.5150** |
| | DTLZ3 | .1737 | .2189 | .0000 | .0870 | .4238 | .2690 | .0000 | .1557 | **.4291** | .0000 | .0975 |
| | WFG3 | .2248 | .3082 | .1389 | .1124 | .2602 | .2217 | .2264 | .2189 | .2710 | .2369 | **.3176** |
| | WFG4 | .4198 | .4550 | .3664 | .3921 | .4065 | .4041 | .4285 | .4176 | .4608 | .4356 | **.4621** |
| | WFG5 | .4699 | .4674 | .4363 | .4442 | .4159 | .4411 | .4686 | .4615 | .4724 | .4724 | **.4729** |
| | WFG6 | .4215 | .4537 | .4020 | .4041 | .4008 | .4089 | .4319 | .4368 | **.4589** | .4416 | .4475 |
| | ZDT1 | .6640 | .6704 | .0438 | .4705 | .5231 | .6126 | .5762 | .5837 | .6282 | .5602 | **.6929** |
| | ZDT2 | .0909 | .0722 | .0000 | .1363 | .1085 | .2731 | .1500 | .2730 | .1836 | .1392 | **.3138** |
| GD | DTLZ1 | .0616 | .0037 | .0568 | .0889 | .0186 | .0589 | .0549 | .0074 | **.0027** | .0213 | .0590 |
| | DTLZ2 | .0052 | .0047 | .0226 | .0247 | .0054 | .0181 | .0179 | .0184 | .0058 | .0160 | **.0023** |
| | DTLZ3 | .1657 | .0063 | .1813 | .2153 | .0050 | .1695 | .1726 | .1916 | **.0041** | .1732 | .1669 |
| | WFG3 | .0906 | .0429 | .1739 | .1541 | .0670 | .1122 | .0863 | .1196 | .0457 | .0711 | **.0367** |
| | WFG4 | .0317 | .0202 | .0425 | .0342 | .0205 | .0285 | .0291 | .0311 | .0202 | .0268 | **.0195** |
| | WFG5 | .0147 | **.0138** | .0204 | .0164 | .0141 | .0166 | .0149 | .0166 | .0141 | .0142 | .0144 |
| | WFG6 | .0325 | .0205 | .0343 | .0296 | **.0193** | .0299 | .0284 | .0289 | .0205 | .0256 | .0240 |
| | ZDT1 | .0002 | .0049 | .1352 | .0326 | .0069 | .0131 | .0177 | .0157 | .0099 | .0181 | **.0002** |
| | ZDT2 | **.0000** | .0333 | .1123 | .0498 | .0187 | .0127 | .0376 | .0161 | .0131 | .0378 | .0001 |
| IGD | DTLZ1 | .1004 | .2762 | .2154 | .2455 | .0960 | .1535 | .1074 | .0579 | **.0519** | .0625 | .1509 |
| | DTLZ2 | **.1019** | .1271 | .1684 | .1796 | .1302 | .1676 | .1533 | .1505 | .1309 | .1454 | .1203 |
| | DTLZ3 | .3500 | .4837 | 1.0354 | .4401 | **.1694** | .3545 | 1.0252 | .6070 | .1782 | 1.0445 | .4047 |
| | WFG3 | .4481 | .2518 | .6454 | .6889 | .4100 | .4730 | .4569 | .4473 | .2799 | .4146 | **.2218** |
| | WFG4 | .4160 | .4935 | .4686 | .4422 | .4969 | .4713 | .4091 | **.4036** | .5078 | .4043 | .5172 |
| | WFG5 | .3765 | .4879 | .4103 | .4025 | .4858 | .4435 | .3773 | **.3654** | .4985 | .3763 | .5112 |
| | WFG6 | .4206 | .5035 | .4632 | .4496 | .5089 | .4749 | .4093 | **.3989** | .5239 | .4029 | .5067 |
| | ZDT1 | .0880 | .0394 | .7278 | .2133 | .2539 | .0798 | .1157 | .1041 | .0705 | .1276 | **.0366** |
| | ZDT2 | .6095 | .5129 | .7385 | .2784 | .4020 | .1344 | .2792 | .1360 | .2294 | .3218 | **.1252** |

The median value of the 31 independent runs and the result of the tests are displayed in Table 5.5. The best values in the table are visualized in bold, and non-significantly different results to the hyper-heuristic are highlighted with a grey cell. The last column shows the result of the hyper-heuristic. The corresponding interquartile (IQR) values are listed in Table 6.1 in the Appendix.
What immediately stands out from the table is that almost all algorithms are not significantly different for DTLZ1 and DTLZ3. That is especially the case for the HV and IGD values. In Section 5.3.1 it was assumed that there is an issue with these problems and that the selection pressure is not high enough to produce usable results. The fact that most algorithms are not significantly different for these problems seems to support this.
For the HV the hyper-heuristic produces outstanding results. It produces the best results in six out of nine problems. SPEA2SDE is a strong contender, having the best results in three of the nine problems. However, the statistical test indicates that the results are not significantly different.
The results displayed in Table 5.5 for the GD and IGD values are not as great as for the HV. For the GD indicator, the hyper-heuristic performed in four test problems the best, with non significantly different values in one test problem. For the IGD indicator, it produced for three test problems the best value, with no significantly different values in two test problems. However, this is not an indication that the hyper-heuristic does not work, but is rather an indication that the fitness of the hyper-heuristic (see Section 4.2) might not be optimal. The fitness is partially calculated with the HV indicator, which is intended as a metric to measure convergence and diversity. GD serves as an indicator for the convergence, and the IGD serves as a measure for the diversity [28]. The hyper-heuristic fulfilled its goal to find a sequence that produces great results for HV metric.
For each experiment series, the critical difference (CD) is calculated. These are displayed in the Figures 5.8, 5.9, and 5.10, respectively for the metrics HV, GD, and IGD. Each algorithm is associated with its average ranking over all test problems in the critical difference plots. The performance of two algorithms is significantly different if the corresponding average ranks differ by at least the critical difference, which is determined based on the pair-wise Nemenyi test, with a p-value of 0.1 [65]. The algorithms connected by a bold horizontal line are considered statistically equivalent. The critical difference plot for the HV shows that the three best performing algorithms are also the algorithms that

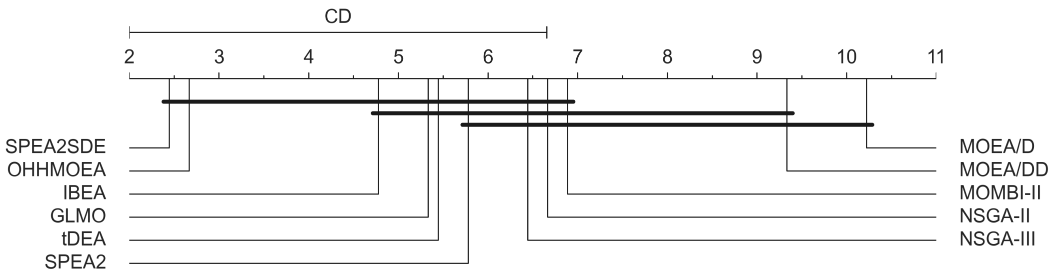


Figure 5.8: Critical Difference Plot for HV

were identified as being the most used in the sequences in Section 5.3.1. These are SPEA2SDE, IBEA, and GLMO. MOMBI-II did not perform as expected if used singularly on the test problems. However, this is different for the critical difference plot for the GD values in Figure 5.9. There, MOMBI-II performs as the third-best MOEA, behind IBEA and SPEA2SDE. That likely means that MOMBI-II is very well usable for producing converging solutions but lacks in producing diversity. That is also likely why MOMBI-II was rarely selected in the last positions of the sequence. In Section 5.3.1 in Figure 5.7, it is visible that MOMBI-II is used at the beginning of the algorithms, and in the end, SPEA2SDE was selected.

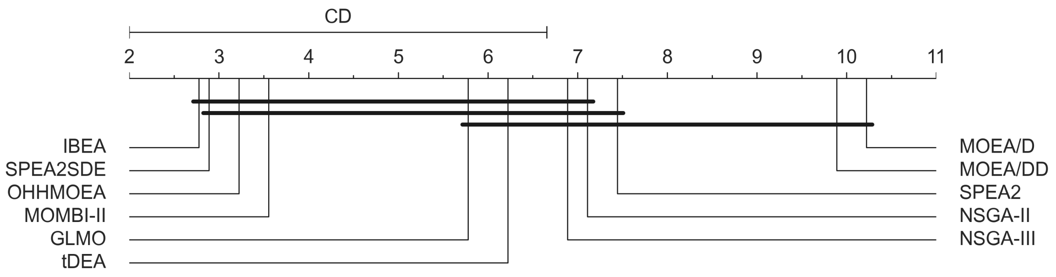


Figure 5.9: Critical Difference Plot for GD

This is also highlighted in the critical difference plot for the IGD values in Figure 5.10, in which MOMBI-II has a relatively worse rank. Surprisingly good performing algorithms in Figure 5.10 are SPEA2 and tDEA, which were rarely used by the hyper-heuristic. Overall, the hyper-heuristic performed

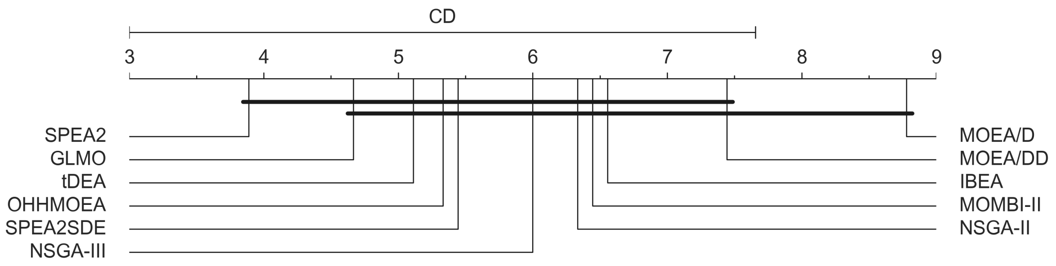


Figure 5.10: Critical Difference Plot for IGD

worse in the IGD indicator and rarely applied algorithms, which performed very well in the IGD indicator. That can be interpreted as the fitness metric, which uses the HV indicator, having a stronger preference for convergence than diversity. It could also be interpreted that the hyper-heuristic prefers algorithms which can deliver both, such as GLMO and SPEA2SDE. On average, both algorithms performed very well in GD, and IGD.
The correlation between the diversity and the HV indicator was studied in [66]. They investigated three different diversity indicators in relation to the HV indicator and found no significant correlation. Their results suggest that the effect of the convergence of the Pareto-approximation set has a greater impact on the HV values than the diversity of the set.
At last, this is also an issue that depends on the user's goal and whether the goal is more diversity or more convergence, or perhaps both. The fitness function of the hyper-heuristic should be designed to fit the user's needs. There is no one-fits-all solution for this problem.
Another fitness function for the hyper-heuristic could be a weighted sum function between an indicator metric for diversity and convergence, making the hyper-heuristic more flexible for different needs. The different metrics could be weighted according to what the user wants. Nevertheless, the HV is universally acclaimed as a metric for MOEAs and produces satisfactory results. A major drawback of the GD and IGD indicator is, that once the solutions are on the Pareto-front their values approach zero, which makes these populations incomparable.
Generally, the average rank of all algorithms is worse for IGD in Figure 5.10. That can be interpreted in all used algorithms here lacking the ability to produce solutions of high diversity over a wide range of different problems.

The hyper-heuristic OHHMOEA does not show a critical difference regarding the ranks in neither HV, GD, nor IGD, in Figures 5.8, 5.9, and 5.10. However, there is a considerable overlap in most groups in all three diagrams, which means the experimental data is not sufficient and the test is not powerful enough to reach any conclusion regarding these algorithms [65].

### 5.3.3 Scalability

The population size $|\mathcal{P}|$ and the maximum function evaluations $\mathcal{FE}_{max}$ for the *HHProblem* are chosen unusually small for test problems. They are chosen smaller due to the high computation time of the hyper-heuristic. Because of that one of the evaluation goals is to find out if the results of the hyper-heuristic are scalable, especially in regards to the function evaluations and the population size. That is question four at the beginning of this chapter. Other scalability parameters that are investigated are the number of objectives and decision variables.
The scalability of the hyper-heuristic likely boils down to the scalability of the inherent MOEAs used. If the utilized MOEAs are scaleable, that sequence will likely be scalable as well. There are certain scenarios in which specific MOEA are more sensitive towards scaling. That is especially true for Pareto-based algorithms such as NSGA-II. With increasing objectives, Pareto-based algorithms will suffer the curse of dimensionality and work less efficiently. Increasing the objective size will also cross the line from multi-objective to many-objective problems. The same corresponds to the number of decision variables. The problem will be considered a many-variable problem with increasing decision variables, and different strategies will be needed to solve the problem.
The previous Section 5.3.1 has shown that the number of decision variables heavily influences the choice of MOEAs. For that reason, the scalability of the decision variables is generally only investigated for the problems which already have a high number of decision variables. In Section 5.3.1 it was already determined that the test problems could be divided into two groups. The test problems for which the decision variables are scaled are DTLZ2, WFG3, ZDT1, and ZDT2. For the other problems, only the objectives are scaled.
For the evaluation the experiments of the previous chapter as presented in Table 5.5 are repeated with scaled values for the *HHProblem*. At first, the same experiments from the previous section are repeated with $|\mathcal{P}| = 105$ and

$\mathcal{FE}_{max} = 10500$ and then with $|\mathcal{P}| = 210$ and $\mathcal{FE}_{max} = 21000$. These two experiments will be called $PF_1$ and $PF_2$. These unique numbers occur be-

Table 5.6: Experiment settings for the test problems. M stands for the number of objectives and D for the number of decision variables. The table heads state which settings are utilized by which experiment.

| | | | | | $PF_2$ | | | |
|---|---|---|---|---|---|---|---|---|
| | $PF_1$ | | | | MD | | | |
| Problem | $\|\mathcal{P}\|$ | $\mathcal{FE}_{max}$ | M | D | $\|\mathcal{P}\|$ | $\mathcal{FE}_{max}$ | M | D |
| DTLZ1 | | | 3 | 5 | | | 6 | 5 |
| DTLZ2 | | | 3 | 40 | | | 3 | 80 |
| DTLZ3 | | | 3 | 5 | | | 6 | 5 |
| WFG3 | | | 3 | 50 | | | 3 | 100 |
| WFG4 | 105 | 10500 | 3 | 50 | 210 | 21000 | 6 | 50 |
| WFG5 | | | 3 | 12 | | | 6 | 12 |
| WFG6 | | | 3 | 12 | | | 6 | 12 |
| ZDT1 | | | 2 | 30 | | | 2 | 60 |
| ZDT2 | | | 2 | 30 | | | 2 | 60 |

cause the hyper-heuristic and other algorithms need well-distributed points in the population. If the population size were set to 100, some algorithms would use an actual population size of 91, while others would use 100, which gives them an unfair advantage. A population size of 105 is perfectly distributable, and all algorithms will use the same population size. The same is true for the population size 210.

Afterward, the experiments are repeated with the number of objectives, and the decision variables doubled for the respective algorithms. The number of decision variables is doubled for DTLZ2, WFG3, ZDT1, and ZDT2, and the number of objectives is doubled for DTLZ1, DTLZ3, WFG4, WFG5, and WFG6. The population size and the maximum function evaluations are set to $|\mathcal{P}| = 210$ and $\mathcal{FE}_{max} = 21000$, since test problems get more difficult to solve with an increasing number of objectives or decision variables. This experiment is named MD. That leaves us with the settings as displayed in Table 5.6. The head of the table displays the experiment, which uses the settings in the respective columns.

The median value of 31 independent runs and the result of statistical tests are displayed in Table 5.7. The Mann–Whitney U rank test is used to determine if the differences in the distribution of the results are significant.

Table 5.7: Median sequence of the offline selection hyper-heuristic evaluated with the HV in the benchmark problem with three different scalability experiments. Values in bold indicate the best value. Values that have a colored cell indicate no significant statistical difference to the hyper-heuristic method OHHMOEA.

| | | GLMO | IBEA | MOEA/D | MOEA/DD | MOMBI-II | NSGA-II | NSGA-III | SPEA2 | SPEA2SDE | tDEA | OHHMOEA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $PF_1$ | DTLZ1 | .8117 | .5532 | .8296 | .8231 | **.8379** | .8166 | .8318 | .8349 | .8341 | .8361 | .8364 |
| | DTLZ2 | .5389 | .5485 | .4820 | .4948 | .5426 | .4776 | .5008 | .5040 | .5444 | .5060 | **.5540** |
| | DTLZ3 | .5111 | .2463 | .5272 | .5105 | .5546 | .5233 | .5414 | .5528 | **.5583** | .5485 | .5520 |
| | WFG3 | .2711 | **.3557** | .1929 | .1280 | .3281 | .2834 | .2784 | .2768 | .3427 | .2963 | .3472 |
| | WFG4 | .4649 | **.5182** | .4161 | .4624 | .4975 | .4525 | .4723 | .4782 | .5096 | .4796 | .5098 |
| | WFG5 | .5063 | .5099 | .4813 | .4988 | .4884 | .4852 | .5062 | .5031 | **.5132** | .5100 | .5115 |
| | WFG6 | .4652 | **.4999** | .4447 | .4573 | .4802 | .4482 | .4786 | .4811 | .4954 | .4739 | .4960 |
| | ZDT1 | .7171 | .7160 | .5896 | .6616 | .6887 | .7080 | .6902 | .7070 | .7105 | .6966 | **.7194** |
| | ZDT2 | .4397 | .3188 | .0990 | .3802 | .3938 | .4262 | .3856 | .4233 | .4299 | .4028 | **.4439** |
| $PF_2$ | DTLZ1 | .8409 | .6503 | .8490 | .8479 | **.8514** | .8384 | .8497 | .8501 | .8488 | .8509 | .8507 |
| | DTLZ2 | .5593 | .5692 | .5286 | .5385 | .5651 | .5205 | .5359 | .5441 | .5654 | .5402 | **.5701** |
| | DTLZ3 | .5597 | .2483 | .5617 | .5632 | .5739 | .5540 | .5706 | .5714 | **.5748** | .5700 | .5727 |
| | WFG3 | .2994 | **.3705** | .2275 | .2433 | .3632 | .3232 | .2974 | .3120 | .3640 | .3228 | .3656 |
| | WFG4 | .4859 | **.5451** | .4425 | .4951 | .5298 | .4799 | .4973 | .5021 | .5303 | .5021 | .5319 |
| | WFG5 | .5202 | **.5317** | .5025 | .5176 | .5146 | .5013 | .5235 | .5219 | .5290 | .5216 | .5282 |
| | WFG6 | .4852 | **.5216** | .4687 | .4889 | .5060 | .4741 | .4991 | .5005 | .5121 | .4976 | .5170 |
| | ZDT1 | **.7220** | .7211 | .7012 | .6998 | .7168 | .7181 | .7116 | .7174 | .7198 | .7142 | .7219 |
| | ZDT2 | **.4466** | .4428 | .2696 | .4175 | .4396 | .4420 | .4283 | .4410 | .4427 | .4311 | .4465 |
| MD | DTLZ1 | .9905 | .9842 | .9880 | .9896 | .9758 | .9868 | .9905 | **.9917** | .9816 | .9903 | .9893 |
| | DTLZ2 | .5506 | .5352 | .2623 | .3512 | .5203 | .4051 | .3862 | .4383 | .5072 | .4081 | **.5635** |
| | DTLZ3 | .8608 | **.8724** | .8600 | .8605 | .8568 | .7953 | .8611 | .8530 | .8706 | .8612 | .8661 |
| | WFG3 | .2550 | **.3462** | .1909 | .1551 | .3200 | .2591 | .2420 | .2535 | .3280 | .2774 | .3397 |
| | WFG4 | .6632 | **.7691** | .4088 | .6373 | .7127 | .5649 | .6691 | .6138 | .7266 | .6813 | .7517 |
| | WFG5 | .7581 | **.7995** | .5092 | .7157 | .6699 | .6088 | .7630 | .7026 | .7655 | .7687 | .7632 |
| | WFG6 | .7127 | .7945 | .3162 | .6772 | .6809 | .6132 | .7261 | .6778 | .7483 | .7312 | **.7975** |
| | ZDT1 | **.7215** | .7045 | .5488 | .6105 | .6935 | .6928 | .6505 | .6865 | .6917 | .6667 | .7210 |
| | ZDT2 | **.4460** | .2019 | .0871 | .2990 | .3958 | .3957 | .3137 | .3878 | .3933 | .3341 | .4456 |

All tests are conducted with a p-value of 0.01 and pair-wise with the hyper-heuristic. The best values in the table are visualized in bold, and non-significantly different results to the hyper-heuristic are highlighted with a grey cell. The corresponding interquartile (IQR) values are listed in Table 6.2 in the Appendix.
For the experiment $PF_1$, an immediate observation is that the results of the hyper-heuristic generally decreased in quality compared to the other algorithms and compared to the results in Table 5.5. Before, the hyper-heuristic produced the best results in six out of nine test problems, and now, the results are only the best in three test problems. However, in two test problems the hyper-heuristic is not performing significantly differently from the best result. The results are very close but significantly different in the other two problems. The quality of the results decreases even further with the experiments $PF_2$. The results of $PF_1$ and $PF_2$ indicate that IBEA benefits greatly from an increased number of individuals in the population and more function evaluations.
$PF_2$ and MD use the same population size and function evaluations. The results show that most algorithms are relatively insensitive to the changes in the number of objectives and decision variables presented here.
Overall, from these results, it has to be stated that the sequences of the hyper-heuristic are generally not very stable regarding changes in the problem settings. However, what can be stated is that the performance of the hyper-heuristic is exceptionally robust in the performance.

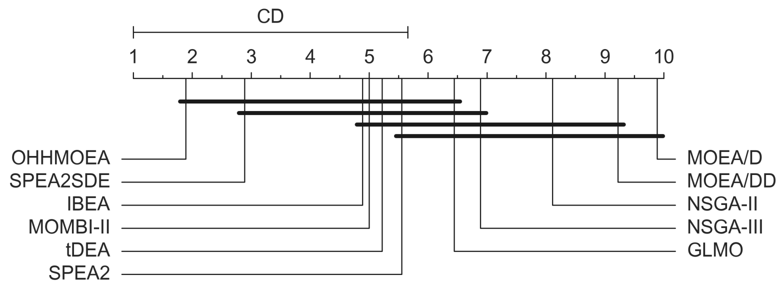


Figure 5.11: Critical Difference Plot for $PF_1$

This is visible in the critical difference plots using the HV for all three experiments in the Figures 5.11, 5.12, and 5.13. Each algorithm is associated with its average ranking over all test problems in the critical difference plots. The performance of two algorithms is significantly different if the correspond-

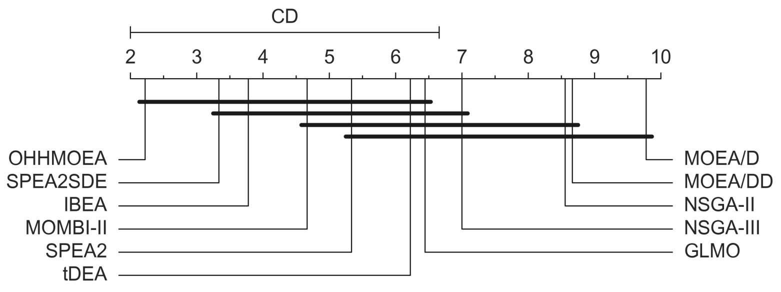


Figure 5.12: Critical Difference Plot for $PF_2$

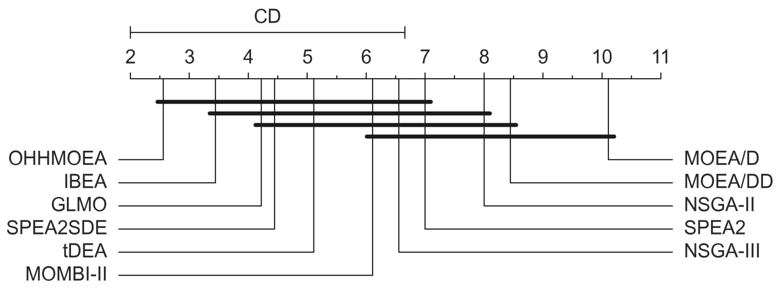


Figure 5.13: Critical Difference Plot for MD

ing average ranks differ by at least the critical difference, which is determined based on the pair-wise Nemenyi test, with a p-value of 0.1 [65]. The algorithms connected by a bold horizontal line are considered statistically equivalent. In all three experiments, the hyper-heuristic has the best average performance. At last, these experiments show that the hyper-heuristic should be learned on the test problem settings on which it will later be evaluated or used. As mentioned previously in this chapter, the hyper-heuristic can significantly benefit if a reduced algorithm set is used to learn the hyper-heuristic, which results in less computation time or earlier convergence.

### 5.3.4 Transfer Learning

The second last question of the evaluation, question five, asks whether the learned sequences of the offline hyper-heuristic can be transferred to other problems. The offline hyper-heuristic learned distinct sequences, or rather a population of sequences, for each respective problem. That makes it difficult to accurately answer this question, as it is unclear how the learned sequences

should be transferred to other problems. In Section 5.3.2, experiments were conducted by using one distinct sequence for each of the respective problems. It would be possible to identify similar optimization problems for each of the utilized problems in that experiment. However, simply transferring learned sequences to other problems or problem domains is likely not feasible. First, it is difficult in the first place to decide which group or problem domain a problem belongs to. And even if that is possible, a sequence might not work well because of some other properties of the problem. For example, in Section 5.3.2 the identified MOEAs that work well on the many-variable problems DTLZ2, WFG3, ZDT1, and ZDT2 do not work well on the problem WFG4, which also has a large number of decision variables. Additionally, the hyper-heuristic has learned a sequence for the problems DTLZ1 and DTLZ3, but it is likely not usable because the problems have too low evolutionary pressure.
In Section 5.3.1, results were separated between two identified optimization problem categories. The categories identified are many-variable problems and those which are not, i.e., normal multi-objective problems. For these two problem categories, a respective sequence is identified by greedily choosing the MOEA which was chosen most often for each position in all sequences in the last populations overall. In Section 5.3.1, for the many-variable problems in Figure 5.6, the sequence positions 1 to 7 are used by GLMO and 8 to 10 by IBEA. For the remaining problems in Figure 5.7, the multi-objective problems, GLMO is used at first, then from position 2 to 6 MOMBI-II, and then from 7 to 10 SPEA2SDE. The sequences are displayed in Table 5.8. Whether sequences

Table 5.8: The sequences used for each respective problem category. MVP stands for many-variable problems and MOP for the multi-objective problems. The numbers are the indices of the MOEAs: (1) GLMO, (2) IBEA, (5) MOMBI-II, (9) SPEA2SDE

| Position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| MOP | 1 | 5 | 5 | 5 | 5 | 5 | 9 | 9 | 9 | 9 |
| MVP | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 |

can be transferred to other problems can be answered by experimenting with these sequences and with problems of a similar nature. For the evaluation of these sequences, the optimization problems of the test suites DTLZ and WFG are utilized. Only the test problems are used, which have not been used in the offline hyper-heuristic to learn these sequences. The experiments are conducted using the population size and functions evaluations utilized by the

offline hyper-heuristic. The population size is 36, and the number of function evaluations is 4032. To simulate the same problem characteristics as identified in the initial experiment of the offline hyper-heuristic in Section 5.3.2 similar problem settings are chosen.
At first, the test problems and their default parameters are used to test the sequence for the multi-objective problems. That means all test problems have three objectives, and the definition of the respective problems determines the number of decision variables. Afterward, the sequence for the many-variable problems is evaluated. For that experiment, the decision variables are increased to 40 for the DTLZ problems and 50 for the WFG problems. The objective size stays the same for all experiments and all test problems. The number of decision variables is chosen lower for the DTLZ problems because the initial HV values indicate that they are harder to solve for the algorithms. The experiment settings for the test problems are displayed in Table 5.9. The

Table 5.9: Experiment settings for the test problems and for the experiments for multi-objective problems (MOP) and for the many-variable problems (MVP). M stands for the number of objectives and D for the number of decision variables. The table heads state which settings are utilized by which experiment.

| | MOP | | MVP | |
|---|---|---|---|---|
| Problem | M | D | M | D |
| DTLZ4 | | 12 | | 40 |
| DTLZ5 | | 12 | | 40 |
| DTLZ6 | | 12 | | 40 |
| DTLZ7 | | 22 | | 40 |
| WFG1 | 3 | 12 | 3 | 50 |
| WFG2 | | 12 | | 50 |
| WFG7 | | 12 | | 50 |
| WFG8 | | 12 | | 50 |
| WFG9 | | 12 | | 50 |

results of the experiments are displayed in Table 5.10, and the algorithms are evaluated using the HV indicator. The Mann–Whitney U rank test is used to determine if the differences in the distribution of the results are significant. All tests are conducted with a p-value of 0.01 and pair-wise with the hyper-heuristic.

Table 5.10: Median sequence of the offline selection hyper-heuristic evaluated with the HV in the benchmark problem with different sequences for multi-objective problems (MOP) and for the many-variable problems (MVP). Values in bold indicate the best value. Values that have a colored cell indicate no significant statistical difference to the hyper-heuristic method OHHMOEA.

| | | GLMO | IBEA | MOEA/D | MOEA/DD | MOMBI-II | NSGA-II | NSGA-III | SPEA2 | SPEA2SDE | tDEA | OHHMOEA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MOP | DTLZ4 | .3255 | .3391 | .0909 | .5174 | .3306 | .3377 | .5192 | .3399 | .3398 | .3107 | **.5269** |
| | DTLZ5 | .1823 | .1932 | .1699 | .1701 | .1771 | .1914 | .1792 | .1917 | **.1939** | .1719 | .1933 |
| | DTLZ6 | .1643 | **.1868** | .0000 | .0000 | .1671 | .0000 | .0000 | .0000 | .1710 | .0000 | .1681 |
| | DTLZ7 | .2216 | .2297 | .1024 | .0879 | .2146 | .1618 | .1421 | .1762 | **.2455** | .1616 | .2351 |
| | WFG1 | .4170 | **.6669** | .3791 | .2305 | .5781 | .5831 | .5058 | .5666 | .6418 | .5403 | .5897 |
| | WFG2 | .8746 | **.8874** | .7564 | .8352 | .8305 | .8532 | .8753 | .8727 | .8753 | .8802 | .8844 |
| | WFG7 | .4713 | .5132 | .3524 | .4058 | .4532 | .4466 | .4703 | .4680 | **.5171** | .4927 | .5135 |
| | WFG8 | .3952 | **.4254** | .3526 | .3645 | .3594 | .3729 | .4002 | .3939 | .4216 | .4029 | .4197 |
| | WFG9 | .4174 | .4704 | .2925 | .3592 | .4262 | .3733 | .4186 | .4226 | .4772 | .4507 | **.4776** |
| MVP | DTLZ4 | **.4409** | .3251 | .0897 | .2631 | .3092 | .2843 | .2538 | .2753 | .3019 | .2545 | .3329 |
| | DTLZ5 | .1790 | .1704 | .0709 | .0385 | .1543 | .1070 | .1018 | .0864 | .1659 | .0875 | **.1882** |
| | DTLZ6 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 | .0000 |
| | DTLZ7 | .2230 | .1198 | .0113 | .0000 | .0675 | .0200 | .0194 | .0134 | .0907 | .0268 | **.2430** |
| | WFG1 | .3034 | **.3867** | .1994 | .2004 | .3331 | .3454 | .3349 | .3618 | .3768 | .3454 | .3446 |
| | WFG2 | .7762 | .7948 | .6376 | .6619 | .7277 | .7371 | .7576 | .7665 | .7795 | .7654 | **.8094** |
| | WFG7 | .3357 | **.4468** | .2337 | .2449 | .3902 | .2850 | .3081 | .3243 | .4382 | .3533 | .3994 |
| | WFG8 | .3914 | .4050 | .3035 | .3173 | .3306 | .3454 | .3733 | .3696 | .3965 | .3820 | **.4079** |
| | WFG9 | .3198 | **.4068** | .1403 | .1748 | .3376 | .3038 | .3251 | .2871 | .4000 | .3107 | .3877 |

The best values in the table are visualized in bold, and non-significantly different results to the hyper-heuristic are highlighted with a grey cell. The corresponding interquartile (IQR) values are listed in Table 6.3 in the Appendix.
For the multi-objective problems (MOP) in Table 5.10, the hyper-heuristic performed best in two problems and performed not significantly different from four other algorithms, which performed better than the hyper-heuristic. For two test problems, other algorithms performed better and were significantly different. While the hyper-heuristic did not perform best in most problems, it performed statistically equal in most problems.
For the many-variable problems in Table 5.10, the hyper-heuristic performed best in four test problems and significantly different in three test problems. The performance is a great achievement and highlights the power of hyper-heuristics in combining different strategies to deliver better results than singular MOEAs. Nevertheless, the hyper-heuristic performed significantly worse in three other test problems. The same as for the MOP problems is true here as well that the MOEAs, which performed best on the original problems, also perform well on the unseen problems. Test problem DTLZ6 is an outlier, which was too difficult to solve for all algorithms and generally resulted in a HV value of zero. This shows the importance of including the GD indicator in the fitness metric of the hyper-heuristic.
From these results, it can be concluded that the sequences can generally not be directly transferred to other problems. However, the learned sequences of other problems can be a great starting point for hyper-heuristic optimization in other problems. The hyper-heuristic should be optimized offline for better results in other unfamiliar test problems.
For two selected test problems, separate independent runs were conducted. From these runs, the HV over the number of evaluations is displayed in Figure 5.14. The Figure depicts a comparison between the hyper-heuristic and two MOEAs. The left diagram in Figure 5.14 shows the DLTZ4 problem with the MOP settings. The hyper-heuristic is compared to a run with MOMBI-II because MOMBI-II uses the earlier positions in the sequences for the MOP problems.
In the first few evaluations for MOP-DLTZ4, both algorithms behave almost identical. In the diagram, it is easily discernible at which evaluation step the hyper-heuristic changes algorithms. The hyper-heuristic changes from MOMBI-II to SPEA2SDE at evaluation step seven. Using only MOMBI-II, the algorithm converges and does not improve anymore. In the hyper-heuristic,

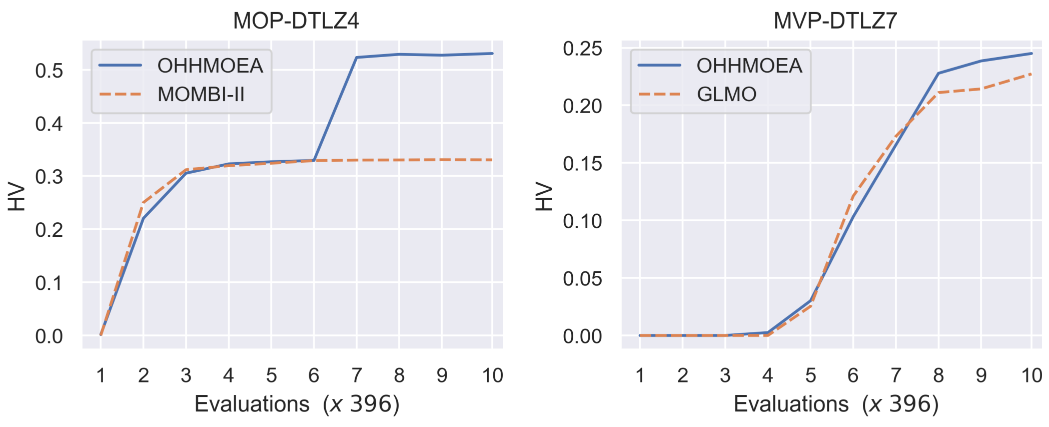


Figure 5.14: The HV indicator plotted over the number of evaluations for two selected test problems and two algorithms per test problem.

the switch to SPEA2SDE improves the HV value significantly. That is because the solutions are likely already on or close to the Pareto-front. Changing to SPEA2SDE at this point in the evaluation will improve the diversity of the already very well optimized solutions.
The right diagram in Figure 5.14 shows the DTLZ7 problem with the MVP settings and the hyper-heuristic compared to the GLMO algorithm. For the MVP problems, the first seven positions in the sequence are used by the GLMO algorithm. In the last three positions, IBEA is utilized. A similar effect as noticed in MOP-DTLZ4 is observable in the last evaluation steps in MVP-DTLZ7, although less severe.
From these observations, it can be concluded that changing to another MOEA will increase the diversity and prevent convergence to some degree. Both IBEA and SPEA2SDE are designed to improve the diversity in the population. The performance highlights how beneficial a change in strategy during optimization can be to overcome converging behavior and increase the population's diversity.

## 5.4 Reduced Algorithm Set

The goal of the last experiment is to investigate the behavior of the offline hyper-heuristic with a reduced algorithm set. More specifically, the question is

whether the hyper-heuristic does benefit from a carefully selected reduced set of MOEAs. The reduced set of MOEAs is identified by analyzing the resulting sequences from the first experiment in this chapter in Section 5.3.1. From these sequences, the MOEAs, which have been used most, is determined by calculating their usage percentage. This is displayed in the beginning of Section 5.3.1 in Figure 5.3. The algorithms that are overwhelmingly most used are IBEA (25%), GLMO (22%), SPEA2SDE (19%), and MOMBI-II (15%). All the other algorithms in the selection are applied almost equally as often with around 3%. An application of only 3% can be considered insignificant and interpreted as statistical noise. That is the reason why these algorithms are not included. With the selected four MOEAs the offline hyper-heuristic is re-evaluated on the test problems. The experiment is conducted identical as outlined in the experimental setup in Section 5.2. One difference is the number of function evaluations $\mathcal{FE}_{max}$ for the Genetic Algorithm, which has been set to 5000. That is half the size of the original experiment, which used 10000 function evaluations. That can be justified by having fewer algorithms and, because of that, a smaller search space. The size of the search space is one million times smaller, $10^4$ instead of $10^{10}$.
The results of the experiment with the reduced set are compared to the original experiment in Table 5.11. The results in Table 5.11 show almost no difference

Table 5.11: For each test problem the median fitness result of the best solutions of the 21 runs and the resulting IQR values for the original experiment and the experiment with the reduced data set.

| | original set | | reduced set | |
|---|---|---|---|---|
| Problem | Median | IQR | Median | IQR |
| DTLZ1 | -.81084 | 9.388e-04 | -.80877 | 3.003e-03 |
| DTLZ2 | -.52473 | 6.834e-04 | -.52464 | 8.898e-04 |
| DTLZ3 | -.52847 | 1.653e-03 | -.52738 | 1.972e-03 |
| WFG3 | -.34492 | 4.255e-03 | -.34251 | 3.072e-03 |
| WFG4 | -.47914 | 2.128e-03 | -.47942 | 1.757e-03 |
| WFG5 | -.48475 | 8.489e-04 | -.48567 | 9.801e-04 |
| WFG6 | -.49353 | 2.717e-03 | -.49669 | 2.176e-03 |
| ZDT1 | -.71252 | 2.129e-04 | -.71255 | 2.651e-04 |
| ZDT2 | -.43650 | 2.668e-04 | -.43648 | 3.510e-04 |

in the median values. In some test problems, the values are even down to the third decimal identical. More significant differences might have been expected,

but the results seem reasonable when it is noted that the values in the table stem from the best solution in each population. The best solutions likely will have less difference and are subject to less variation than the whole population of each run since the best solution in each population might stay constant throughout several generations.
For further evaluation, the results of the reduced set are subjected to the Mann–Whitney U rank test to determine whether the differences in the distribution of the results are significant. All tests are conducted with a p-value of 0.05. The test shows significant differences in the results of the test problems DTLZ1, DTLZ3, WFG5, and WFG6. The MOEA application distribution in the original experiment can explain the significant differences in DTLZ1 and DTLZ3. In the original experiment, the resulting sequences for these test problems contain a substantial amount of the MOEAs that are not selected for the reduced set, e.g., MOEA/D ( 12% and 15%) and SPEA2 (12% and 10%). Since these algorithms are not contained anymore in the reduced set, these positions have to be filled by other MOEAs. That explains why the distributions are now significantly different. The same seems to be true for WFG5 and WFG6. Although the distributions for these test problems are not as varied as for DTLZ1 and DTLZ3, they are still more varied than the other test problems.
Only half the amount of function evaluations are utilized for this experiment than for the original one, resulting in significantly less computation time while simultaneously delivering results almost identical to the original set. That shows that the hyper-heuristic can significantly benefit from a reduced algorithm set. However, that can also result in drawbacks in the performance of some test problems. The reduced set in this experiment was derived from statistics over all the test problems. Some test problems could be outliers and might require different MOEAs in their optimization to yield good results than the majority of test problems. That highlights the need for individual evaluations on specific optimization problems when using the hyper-heuristic with a reduced set.

## 5.5 Summary and Discussion

In this chapter, the Offline Learning Hyper-Heuristic collaborative Multi-objective Evolutionary Algorithm (OHHMOEA) was thoroughly evaluated on nine different optimization problems from three different benchmark suites.

The resulting sequences of the offline hyper-heuristic show a distinct behavior towards problems of specific characteristics. The characteristics identified are many-variable problems, and multi-objective problems. For each of the test problems, the sequences are further analyzed regarding their performance compared to the MOEAs. The hyper-heuristic delivered results of high quality regarding the HV. However, comparing the results with the GD and IGD indicators, the performance quality fades.
Next, the scalability of these sequences was evaluated regarding the population size, function evaluations, objectives, and decision variables. Afterward, whether the results of the offline hyper-heuristic can be transferred to other test problems is examined. The MOEA sequences that perform best on the test problems on average are identified, and evaluated on unseen test problems. At last, a reduced algorithm set was created from the results of the offline hyper-heuristic. The reduced algorithm set is re-evaluated on the test problems.

The selection of test problems for the evaluation of the hyper-heuristic can be improved upon. More specifically, the number of test problems for the evaluation can be increased and the diversity of the test problems expanded. The selected test problems especially lack a diversity in the number of objectives. Additionally, the size of the population and the number of function evaluations for the test problems was chosen relatively small. These values could be increased to further enhance the validity of the evaluation.
The experiments contain test problems that are labeled as many-variable problems. However, the number of decision variables is set to a value, which the literature would consider at the lower end for many-variable problems [67]. Nevertheless, the problems that are labeled as many-variable problems in this chapter have shown a distinct behavior in the selection of MOEAs they prefer, compared to the multi-objective problems.

In the experiments, the average ranking of the algorithms is compared in critical difference plots. It is questionable how the average rank should be interpreted. One issue is that the hyper-heuristic is a meta-algorithm. Therefore the comparison between the average rank of an algorithm with the rank of a meta-algorithm is unreasonable. If a simple single choice hyper-heuristic would be used, which hypothetically always chooses the best algorithm, it could beat the hyper-heuristic in all experiments regarding the average ranking. A good performing sequence-based selection hyper-heuristic should perform equally as well or even better than other algorithms. Having a better average ranking than other algorithms is not sufficient. Nevertheless, the average ranking can be utilized as a performance measure of how well the hyper-heuristic has learned.

The computation cost of the offline hyper-heuristic and the gain in performance on the test problems is a point of discussion. Applying hyper-heuristics makes sense if the results are significantly better or if insights can be gained into the collaboration of different optimization strategies. The implication is that a great application area of offline hyper-heuristics is for problems, in which heuristics fail to find well-performing solutions or the user's goal is to optimize the solutions subject to certain evaluation metrics such as the HV.

In Section 5.4 a reduced subset of MOEAs is utilized. The reduced subset is based on the initial experiment evaluated in Section 5.3.1. Instead of a reduced algorithm set, which requires that the algorithms are evaluated before, the hyper-heuristic itself could implement such an evaluation step. That could be done by altering the method of how the population of sequences is initialized. Instead of initializing the population completely randomly, a specific amount of sequences could be initialized with just one and the same MOEA at each position. If these sequences perform well, they would automatically have a higher chance of being selected in the evolutionary process. Additionally, the mutation operator could be altered to include a probability for each MOEA to be selected as the replacement. Currently, this is done uniformly at random. The probability for each MOEA could be based on their performance.

# 6 Conclusion and Future Work

In this chapter, the thesis is summarized, and conclusions regarding the questions from Chapter 1 are provided. At last, some promising topics for future research are presented.

In Chapter 2, the fundamental knowledge regarding Evolutionary Algorithms is explained. The quality indicators HV, R2, GD, and IGD are introduced. Afterward, 11 state-of-the-art MOEAs are presented, and their differences are explained. At the end of the chapter, the popular benchmark suites DTLZ, WFG, and ZDT are presented.
An overview of the related work regarding hyper-heuristics is given in Chapter 3. The chapter focused on selection hyper-heuristics and gave an introduction to offline hyper-heuristics, including parameter optimization.
Chapter 4 provided an overview of the theoretical design and implementation of the Offline Learning Hyper-Heuristic collaborative Multi-objective Evolutionary Algorithm (OHHMOEA). To the best of the author's knowledge, the OHHMOEA is one of the first offline selection hyper-heuristics using MOEAs.
The evaluation and the results of the experiments on OHHMOEA are presented in Chapter 5. The offline hyper-heuristic is evaluated on nine different optimization problems. The resulting sequences of the offline hyper-heuristic are the key element of the evaluation chapter. Central to the evaluation chapter and this thesis are the following six questions, which have been introduced in Chapter 1:

**Do certain sub-sequences of MOEAs occur more often than others?**
The evaluation in Section 5.3.1 has shown that certain sub-sequences occur more often than others. That is especially the case for the best performing MOEAs in the selection, which occur very frequently. Besides that, there are distinct pairs of MOEAs which seem to work quite well in collaboration. GLMO and IBEA, and MOMBI-II with SPEA2SDE were often selected together. An interesting observation is that MOEAs have a specific place in the sequence. MOEAs which optimize towards diversity are most often chosen at

the end. The beginning of the sequence is most often occupied by MOEAs applying distinct strategies for specific problem classes, such as many-variable problems.

**Are the resulting sequences problem-specific?**
In Section 5.3.1, the sequences are analyzed based on their properties towards being problem-specific. The sequences could be separated between multi-objective problems and many-variable problems. Each of these problem classes has shown to tend towards specific MOEAs. However, each benchmark problem maintains its unique sequence that works best. These sequences are often only slight variations from each other.

**How does the OHHMOEA perform compared to the selected MOEAs?**
The hyper-heuristic is compared to the utilized MOEAs in the hyper-heuristic in Section 5.3.2. The utilized MOEAs are GLMO, IBEA, MOEA/D, MOEA/DD, MOMBI-II, NSGA-II, NSGA-III, SPEA2, SPEA2SDE, and theta-DEA. For the HV, the hyper-heuristic produced outstanding results, while the quality of the results regarding the GD and the IGD indicator diminished. The major achievement of the hyper-heuristic is that it produced significantly better results in some benchmark problems. The results highlight the advantage of hyper-heuristics in combining several different strategies to solve optimization problems.

**Are the results of the sequences 'stable' if the population size, the function evaluations, the objective size, or decision variables are increased?**
The results from Section 5.3.3 show that the hyper-heuristic can generally not be considered stable towards these parameters. On average, the hyper-heuristic still performs very well.To achieve better performance, the offline hyper-heuristic needs to be evaluated with the problem settings it will be evaluated with later. Generally, the performance of the hyper-heuristic on other problem settings can be utilized as a starting point for further learning.

**Are the learned sequences transferable to other problems?**
It is examined whether the results of the offline hyper-heuristic can be transferred to other test problems in Section 5.3.4. The MOEA sequences that perform best on the test problems on average are identified and evaluated on unseen test problems. It is shown that the sequences can generally not be directly transferred to other problems. However, the learned sequences still

perform quite well and can be a starting point for offline learning of the hyper-heuristic. It is also shown how switching to different MOEAs during optimization can have significant impacts on the performance in terms of diversity and convergence.

**Does the offline hyper-heuristic benefit from a reduced algorithm set?**
At last, a reduced algorithm set was created from the results of the offline hyper-heuristic. The reduced algorithm set is re-evaluated on the test problems in Section 5.4. The re-evaluation results have shown no significantly different results in five out of nine test problems. While the results are significantly different in the other four test problems, it can be expected that their performances are likely very similar to the original algorithm set since the median values of the results do not differ very much. The main advantage of using a reduced algorithm set is that less computation time is required for the convergence of the hyper-heuristic.

## 6.1 Future Work

Although the OHHMOEA has been shown to perform well and produced fascinating results, improvements are still possible. The hyper-heuristic design presented in this thesis can be used as a starting point for promising future work.
The integer encoding used in the hyper-heuristic worked well. However, other implementations are imaginable. The implemented encoding is fixed in length. It would be possible to develop an encoding with a dynamic length or to increase fixed-length size.
The fitness metric designed for the hyper-heuristic is a major improvement point but likely also subject to the user's preference.

The hyper-heuristic should be evaluated on test problems with even more decision variables or objectives than presented in this thesis to accurately evaluate the hyper-heuristic. Further research on the topic of many-variable and many-objective problems is needed.

Comparing the OHHMOEA to an online learning hyper-heuristic is one major point of interest that has not been addressed in this work.

The selection of MOEAs for the hyper-heuristic can be made more meticulous. The selection was made based on the related work to this thesis, on previous experiences of the author, and more or less arbitrarily. It is shown that the MOEAs for the hyper-heuristic could be broken down into four relevant algorithms. That means some MOEAs in the hyper-heuristic can be regarded as redundant. A more diverse set of MOEAs with distinct strategies to solve optimization problems could also potentially benefit the performance of the hyper-heuristic.

Finally, one major future work is the application of the offline hyper-heuristic to real-world problems.

# Bibliography

[1] Edmund K Burke et al. "Hyper-Heuristics: A Survey of the State of the Art". In: *Journal of the Operational Research Society* 64.12 (Dec. 2013), pp. 1695–1724. ISSN: 0160-5682, 1476-9360. DOI: 10.1057/jors.2013.71.

[2] Gian Fritsche and Aurora Pozo. "A Hyper-Heuristic Collaborative Multi-Objective Evolutionary Algorithm". In: *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*. 2018 7th Brazilian Conference on Intelligent Systems (BRACIS). Sao Paulo, Brazil: IEEE, Oct. 2018, pp. 354–359. ISBN: 978-1-5386-8023-0. DOI: 10.1109/BRACIS.2018.00068.

[3] Edmund K. Burke et al. "A Classification of Hyper-Heuristic Approaches: Revisited". In: *Handbook of Metaheuristics*. Ed. by Michel Gendreau and Jean-Yves Potvin. Vol. 272. International Series in Operations Research & Management Science. Cham: Springer International Publishing, 2019, pp. 453–477. ISBN: 978-3-319-91085-7 978-3-319-91086-4. DOI: 10.1007/978-3-319-91086-4_14.

[4] John H. Drake et al. "Recent Advances in Selection Hyper-Heuristics". In: *European Journal of Operational Research* 285.2 (Sept. 2020), pp. 405–428. ISSN: 03772217. DOI: 10.1016/j.ejor.2019.07.073.

[5] David J. Walker and Ed Keedwell. "Towards Many-Objective Optimisation with Hyper-Heuristics: Identifying Good Heuristics with Indicators". In: *Parallel Problem Solving from Nature – PPSN XIV*. Ed. by Julia Handl et al. Vol. 9921. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 493–502. ISBN: 978-3-319-45822-9 978-3-319-45823-6. DOI: 10.1007/978-3-319-45823-6_46.

[6] Rudolf Kruse et al. *Computational Intelligence*. Texts in Computer Science. London: Springer London, 2016. ISBN: 978-1-4471-7294-9 978-1-4471-7296-3. DOI: 10.1007/978-1-4471-7296-3.

[7] Kalyanmoy Deb and A Raji Reddy. "Reliable classification of two-class cancer data using evolutionary algorithms". In: *Biosystems* 72.1 (2003).

Computational Intelligence in Bioinformatics, pp. 111–129. ISSN: 0303-2647. DOI: 10.1016/S0303-2647(03)00138-2.

[8] Gian Fritsche and Aurora Pozo. "The Analysis of a Cooperative Hyper-Heuristic on a Constrained Real-World Many-Objective Continuous Problem". In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. 2020 IEEE Congress on Evolutionary Computation (CEC). Glasgow, United Kingdom: IEEE, July 2020, pp. 1–8. ISBN: 978-1-72816-929-3. DOI: 10.1109/CEC48606.2020.9185904.

[9] J. Munoz-Paniagua and J. Garcia. "Aerodynamic drag optimization of a high-speed train". In: *Journal of Wind Engineering and Industrial Aerodynamics* 204 (2020), p. 104215. ISSN: 0167-6105. DOI: 10.1016/j.jweia.2020.104215.

[10] Ye Tian et al. "PlatEMO: A MATLAB platform for evolutionary multi-objective optimization". In: *IEEE Computational Intelligence Magazine* 12.4 (2017), pp. 73–87.

[11] Gian Fritsche and Aurora Pozo. "Cooperative Based Hyper-Heuristic for Many-Objective Optimization". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '19: Genetic and Evolutionary Computation Conference. Prague Czech Republic: ACM, July 13, 2019, pp. 550–558. ISBN: 978-1-4503-6111-8. DOI: 10.1145/3321707.3321740.

[12] Ahmed Kheiri and Ed Keedwell. "A Sequence-Based Selection Hyper-Heuristic Utilising a Hidden Markov Model". In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. GECCO '15: Genetic and Evolutionary Computation Conference. Madrid Spain: ACM, July 11, 2015, pp. 417–424. ISBN: 978-1-4503-3472-3. DOI: 10.1145/2739480.2754766.

[13] Wenwen Li, Ender Özcan, and Robert John. "A Learning Automata-Based Multiobjective Hyper-Heuristic". In: *IEEE Transactions on Evolutionary Computation* 23.1 (2017), pp. 59–73. ISSN: 1089-778X, 1089-778X, 1941-0026. DOI: 10.1109/TEVC.2017.2785346.

[14] Mashael Maashi, Graham Kendall, and Ender Özcan. "Choice Function Based Hyper-Heuristics for Multi-Objective Optimization". In: *Applied Soft Computing* 28 (Mar. 2015), pp. 312–326. ISSN: 15684946. DOI: 10.1016/j.asoc.2014.12.012.

[15] Kent McClymont and Edward C. Keedwell. "Markov Chain Hyper-Heuristic (MCHH): An Online Selective Hyper-Heuristic for Multi-

Objective Continuous Problems". In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. GECCO '11. Dublin, Ireland: Association for Computing Machinery, 2011, pp. 2003–2010. ISBN: 9781450305570. DOI: 10.1145/2001576.2001845.

[16] Vinicius Renan de Carvalho and Jaime Simão Sichman. "Multi-Agent Election-Based Hyper-Heuristics". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. Twenty-Seventh International Joint Conference on Artificial Intelligence {IJCAI-18}. Stockholm, Sweden: International Joint Conferences on Artificial Intelligence Organization, July 2018, pp. 5779–5780. ISBN: 978-0-9992411-2-7. DOI: 10.24963/ijcai.2018/833.

[17] Tailong Yang, Shuyan Zhang, and Cuixia Li. "A Multi-Objective Hyper-Heuristic Algorithm Based on Adaptive Epsilon-Greedy Selection". In: *Complex & Intelligent Systems* 7.2 (Apr. 2021), pp. 765–780. ISSN: 2199-4536, 2198-6053. DOI: 10.1007/s40747-020-00230-8.

[18] Mashael Maashi, Ender Özcan, and Graham Kendall. "A Multi-Objective Hyper-Heuristic Based on Choice Function". In: *Expert Systems with Applications* 41.9 (July 2014), pp. 4475–4493. ISSN: 09574174. DOI: 10.1016/j.eswa.2013.12.050.

[19] E. Zitzler and L. Thiele. "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach". In: *IEEE Transactions on Evolutionary Computation* 3.4 (Nov./1999), pp. 257–271. ISSN: 1089778X. DOI: 10.1109/4235.797969.

[20] L. While et al. "A Faster Algorithm for Calculating Hypervolume". In: *IEEE Transactions on Evolutionary Computation* 10.1 (Feb. 2006), pp. 29–38. ISSN: 1089-778X. DOI: 10.1109/TEVC.2005.851275.

[21] Andreia P. Guerreiro, Carlos M. Fonseca, and Luís Paquete. "The Hypervolume Indicator: Computational Problems and Algorithms". In: *ACM Computing Surveys* 54.6 (July 2021), pp. 1–42. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3453474.

[22] Johannes Bader and Eckart Zitzler. "HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization". In: *Evolutionary Computation* 19.1 (Mar. 2011), pp. 45–76. ISSN: 1063-6560, 1530-9304. DOI: 10.1162/EVCO_a_00009.

[23] Anne Auger et al. "Hypervolume-Based Multiobjective Optimization: Theoretical Foundations and Practical Implications". In: *Theoretical*

*Computer Science* 425 (Mar. 2012), pp. 75–103. ISSN: 03043975. DOI: 10.1016/j.tcs.2011.03.012.

[24] Dimo Brockhoff, Tobias Wagner, and Heike Trautmann. “On the Properties of the R2 Indicator”. In: *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference - GECCO ’12*. The Fourteenth International Conference. Philadelphia, Pennsylvania, USA: ACM Press, 2012, p. 465. ISBN: 978-1-4503-1177-9. DOI: 10.1145/2330163.2330230.

[25] Anne Auger et al. “Theory of the Hypervolume Indicator: Optimal $\mu$-Distributions and the Choice of the Reference Point”. In: *Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms - FOGA ’09*. The Tenth ACM SIGEVO Workshop. Orlando, Florida, USA: ACM Press, 2009, p. 87. ISBN: 978-1-60558-414-0. DOI: 10.1145/1527125.1527138.

[26] N. Beume et al. “On the Complexity of Computing the Hypervolume Indicator”. In: *IEEE Transactions on Evolutionary Computation* 13.5 (Oct. 2009), pp. 1075–1082. ISSN: 1941-0026, 1089-778X. DOI: 10.1109/TEVC.2009.2015575.

[27] David A Van Veldhuizen. “Multiobjective Evolutionary Algorithms: Classi Cations, Analyses, and New Innovations”. In: (), p. 270.

[28] P.A.N. Bosman and D. Thierens. “The Balance between Proximity and Diversity in Multiobjective Evolutionary Algorithms”. In: *IEEE Transactions on Evolutionary Computation* 7.2 (Apr. 2003), pp. 174–188. ISSN: 1089-778X. DOI: 10.1109/TEVC.2003.810761.

[29] Yanan Sun, Gary G. Yen, and Zhang Yi. “IGD Indicator-Based Evolutionary Algorithm for Many-Objective Optimization Problems”. In: *IEEE Transactions on Evolutionary Computation* 23.2 (Apr. 2019), pp. 173–187. ISSN: 1089-778X, 1089-778X, 1941-0026. DOI: 10.1109/TEVC.2018.2791283. arXiv: 1802.08792.

[30] Lie Meng Pang, Hisao Ishibuchi, and Ke Shang. “Offline Automatic Parameter Tuning of MOEA/D Using Genetic Algorithm”. In: *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2019 IEEE Symposium Series on Computational Intelligence (SSCI). Xiamen, China: IEEE, Dec. 2019, pp. 1889–1897. ISBN: 978-1-72812-485-8. DOI: 10.1109/SSCI44817.2019.9002787.

[31] Kalyanmoy Deb and Himanshu Jain. “An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated

Sorting Approach, Part I: Solving Problems With Box Constraints". In: *IEEE Transactions on Evolutionary Computation* 18.4 (Aug. 2014), pp. 577–601. ISSN: 1089-778X, 1089-778X, 1941-0026. DOI: 10.1109/TEVC.2013.2281535.

[32] Miqing Li, Shengxiang Yang, and Xiaohui Liu. "Shift-Based Density Estimation for Pareto-Based Algorithms in Many-Objective Optimization". In: *IEEE Transactions on Evolutionary Computation* 18.3 (June 2014), pp. 348–365. ISSN: 1089-778X, 1089-778X, 1941-0026. DOI: 10.1109/TEVC.2013.2262178.

[33] Richard Bellman. *Dynamic Programming.* 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.

[34] Heiner Zille et al. "A Framework for Large-Scale Multiobjective Optimization Based on Problem Transformation". In: *IEEE Transactions on Evolutionary Computation* 22.2 (Apr. 2018), pp. 260–275. ISSN: 1089-778X, 1089-778X, 1941-0026. DOI: 10.1109/TEVC.2017.2704782.

[35] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. "SPEA2: Improving the Strength Pareto Evolutionary Algorithm". In: (2001). DOI: 10.3929/ETHZ-A-004284029.

[36] K. Deb et al. "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (Apr. 2002), pp. 182–197. ISSN: 1089778X. DOI: 10.1109/4235.996017.

[37] Qingfu Zhang and Hui Li. "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition". In: *IEEE Transactions on Evolutionary Computation* 11.6 (Dec. 2007), pp. 712–731. ISSN: 1941-0026, 1089-778X. DOI: 10.1109/TEVC.2007.892759.

[38] Ke Li et al. "An Evolutionary Many-Objective Optimization Algorithm Based on Dominance and Decomposition". In: *IEEE Transactions on Evolutionary Computation* 19.5 (Oct. 2015), pp. 694–716. ISSN: 1089-778X, 1089-778X, 1941-0026. DOI: 10.1109/TEVC.2014.2373386.

[39] Eckart Zitzler and Simon Künzli. "Indicator-Based Selection in Multiobjective Search". In: *Parallel Problem Solving from Nature - PPSN VIII.* Ed. by Xin Yao et al. Red. by David Hutchison et al. Vol. 3242. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 832–842. ISBN: 978-3-540-23092-2 978-3-540-30217-9. DOI: 10.1007/978-3-540-30217-9_84.

[40] Raquel Hernández Gómez and Carlos A. Coello Coello. "Improved Metaheuristic Based on the R2 Indicator for Many-Objective Optimization".

In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. GECCO '15: Genetic and Evolutionary Computation Conference. Madrid Spain: ACM, July 11, 2015, pp. 679–686. ISBN: 978-1-4503-3472-3. DOI: 10.1145/2739480.2754776.

[41] Raquel Hernandez Gomez and Carlos A. Coello Coello. "MOMBI: A New Metaheuristic for Many-Objective Optimization Based on the R2 Indicator". In: *2013 IEEE Congress on Evolutionary Computation*. 2013 IEEE Congress on Evolutionary Computation (CEC). Cancun, Mexico: IEEE, June 2013, pp. 2488–2495. ISBN: 978-1-4799-0454-9 978-1-4799-0453-2 978-1-4799-0451-8 978-1-4799-0452-5. DOI: 10.1109/CEC.2013.6557868.

[42] Yuan Yuan et al. "A New Dominance Relation-Based Evolutionary Algorithm for Many-Objective Optimization". In: *IEEE Transactions on Evolutionary Computation* 20.1 (Feb. 2016), pp. 16–37. ISSN: 1089-778X, 1089-778X, 1941-0026. DOI: 10.1109/TEVC.2015.2420112.

[43] Heiner Zille et al. "Mutation Operators Based on Variable Grouping for Multi-Objective Large-Scale Optimization". In: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2016 IEEE Symposium Series on Computational Intelligence (SSCI). Athens, Greece: IEEE, Dec. 2016, pp. 1–8. ISBN: 978-1-5090-4240-1. DOI: 10.1109/SSCI.2016.7850214.

[44] Kalyanmoy Deb et al. "Scalable Test Problems for Evolutionary Multiobjective Optimization". In: *Evolutionary Multiobjective Optimization*. Ed. by Ajith Abraham, Lakhmi Jain, and Robert Goldberg. Advanced Information and Knowledge Processing. London: Springer-Verlag, 2005, pp. 105–145. ISBN: 978-1-85233-787-2. DOI: 10.1007/1-84628-137-7_6.

[45] Simon Huband et al. "A Scalable Multi-objective Test Problem Toolkit". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 280–295. DOI: 10.1007/978-3-540-31880-4_20.

[46] Javier Del Ser et al. "Bio-Inspired Computation: Where We Stand and What's Next". In: *Swarm and Evolutionary Computation* 48 (Aug. 2019), pp. 220–250. ISSN: 22106502. DOI: 10.1016/j.swevo.2019.04.008.

[47] S. Huband et al. "A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit". In: *IEEE Transactions on Evolutionary Computation* 10.5 (Oct. 2006), pp. 477–506. ISSN: 1089-778X. DOI: 10.1109/TEVC.2005.861417.

[48] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results”. In: *Evolutionary Computation* 8.2 (June 2000), pp. 173–195. ISSN: 1063-6560, 1530-9304. DOI: `10.1162/106365600568202`.

[49] Hisao Ishibuchi, Hiroyuki Masuda, and Yusuke Nojima. “Pareto Fronts of Many-Objective Degenerate Test Problems”. In: *IEEE Transactions on Evolutionary Computation* 20.5 (Oct. 2016), pp. 807–813. ISSN: 1089-778X, 1089-778X, 1941-0026. DOI: `10.1109/TEVC.2015.2505784`.

[50] Jesús Guillermo Falcón-Cardona and Carlos A. Coello Coello. “A Multi-Objective Evolutionary Hyper-Heuristic Based on Multiple Indicator-Based Density Estimators”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO ’18: Genetic and Evolutionary Computation Conference. Kyoto Japan: ACM, July 2, 2018, pp. 633–640. ISBN: 978-1-4503-5618-3. DOI: `10.1145/3205455.3205463`.

[51] Valdivino Alexandre de Santiago Júnior, Ender Özcan, and Vinicius Renan de Carvalho. “Hyper-Heuristics Based on Reinforcement Learning, Balanced Heuristic Selection and Group Decision Acceptance”. In: *Applied Soft Computing* 97 (Dec. 2020), p. 106760. ISSN: 15684946. DOI: `10.1016/j.asoc.2020.106760`.

[52] C.M. Fonseca and P.J. Fleming. “Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation”. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 28.1 (1998), pp. 26–37. DOI: `10.1109/3468.650319`.

[53] William B. Yates and Edward C. Keedwell. “Offline Learning for Selection Hyper-Heuristics with Elman Networks”. In: *Artificial Evolution*. Ed. by Evelyne Lutton et al. Vol. 10764. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 217–230. ISBN: 978-3-319-78132-7 978-3-319-78133-4. DOI: `10.1007/978-3-319-78133-4_16`.

[54] W. B. Yates and E. C. Keedwell. “An Analysis of Heuristic Subsequences for Offline Hyper-Heuristic Learning”. In: *Journal of Heuristics* 25.3 (June 2019), pp. 399–430. ISSN: 1381-1231, 1572-9397. DOI: `10.1007/s10732-018-09404-7`.

[55] William B. Yates and Edward C. Keedwell. “Offline Learning with a Selection Hyper-Heuristic: An Application to Water Distribution Network

Optimisation". In: *Evolutionary Computation* (June 22, 2020), pp. 1–24. ISSN: 1063-6560, 1530-9304. DOI: 10.1162/evco_a_00277.

[56] Marc Claesen and Bart De Moor. *Hyperparameter Search in Machine Learning*. Apr. 6, 2015. arXiv: 1502.02127 [cs, stat].

[57] A.E. Eiben, R. Hinterding, and Z. Michalewicz. "Parameter Control in Evolutionary Algorithms". In: *IEEE Transactions on Evolutionary Computation* 3.2 (July 1999), pp. 124–141. ISSN: 1089778X. DOI: 10.1109/4235.771166.

[58] Ryoji Tanabe and Hisao Ishibuchi. "An Analysis of Control Parameters of MOEA/D under Two Different Optimization Scenarios". In: *Applied Soft Computing* 70 (Sept. 2018), pp. 22–40. ISSN: 15684946. DOI: 10.1016/j.asoc.2018.05.014.

[59] Kenneth De Jong. "Parameter Setting in EAs: A 30 Year Perspective". In: *Parameter Setting in Evolutionary Algorithms*. Ed. by Fernando G. Lobo, Cláudio F. Lima, and Zbigniew Michalewicz. Red. by Janusz Kacprzyk. Vol. 54. Studies in Computational Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–18. ISBN: 978-3-540-69431-1 978-3-540-69432-8. DOI: 10.1007/978-3-540-69432-8_1.

[60] Lie Meng Pang, Hisao Ishibuchi, and Ke Shang. "Algorithm Configurations of MOEA/D with an Unbounded External Archive". In: *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC). Toronto, ON, Canada: IEEE, Oct. 11, 2020, pp. 1087–1094. ISBN: 978-1-72818-526-2. DOI: 10.1109/SMC42975.2020.9283395.

[61] Lie Meng Pang, Hisao Ishibuchi, and Ke Shang. "Decomposition-Based Multi-Objective Evolutionary Algorithm Design Under Two Algorithm Frameworks". In: *IEEE Access* 8 (2020), pp. 163197–163208. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3022164.

[62] Huangke Chen et al. "PEA: Parallel Evolutionary Algorithm by Separating Convergence and Diversity for Large-Scale Multi-Objective Optimization". In: *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). Vienna: IEEE, July 2018, pp. 223–232. ISBN: 978-1-5386-6871-9. DOI: 10.1109/ICDCS.2018.00031.

[63] Huangke Chen et al. "Solving Large-Scale Many-Objective Optimization Problems by Covariance Matrix Adaptation Evolution Strategy

with Scalable Small Subpopulations". In: *Information Sciences* 509 (Jan. 2020), pp. 457–469. ISSN: 00200255. DOI: 10.1016/j.ins.2018.10.007.

[64] Gregory W. Corder and Dale I. Foreman. *Nonparametric Statistics - A Step-by-Step Approach.* New York, 2014. ISBN: 978-1-118-84031-3.

[65] Janez Demšar. "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *J. Mach. Learn. Res.* 7 (Dec. 2006), pp. 1–30. ISSN: 1532-4435. DOI: 10.5555/1248547.1248548.

[66] Carlos R. B. Azevedo and Aluizio F. R. Araujo. "Correlation between Diversity and Hypervolume in Evolutionary Multiobjective Optimization". In: *2011 IEEE Congress of Evolutionary Computation (CEC).* 2011 IEEE Congress on Evolutionary Computation (CEC). New Orleans, LA, USA: IEEE, June 2011, pp. 2743–2750. ISBN: 978-1-4244-7834-7. DOI: 10.1109/CEC.2011.5949962.

[67] Sedigheh Mahdavi, Mohammad Ebrahim Shiri, and Shahryar Rahnamayan. "Metaheuristics in large-scale global continues optimization: A survey". In: *Information Sciences* 295 (2015), pp. 407–428. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2014.10.042.

# Appendix

Table 6.1: Corresponding IQR values for Table 5.5

| | | GLMO | IBEA | MOEA/D | MOEA/DD | MOMBI-II | NSGA-II | NSGA-III | SPEA2 | SPEA2SDE | tDEA | OHHMOEA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HV | DTLZ1 | 6.013e-1 | 3.016e-1 | 6.098e-1 | 4.977e-1 | 6.314e-1 | 6.956e-1 | 7.032e-1 | 5.217e-1 | **6.513e-1** | 6.543e-1 | 6.488e-1 |
| | DTLZ2 | 8.371e-3 | 1.660e-2 | 6.141e-2 | 5.287e-2 | 1.633e-2 | 3.073e-2 | 5.011e-2 | 5.490e-2 | 1.836e-2 | 6.160e-2 | **4.885e-3** |
| | DTLZ3 | 3.920e-1 | 2.377e-1 | 3.054e-1 | 2.560e-1 | 4.284e-1 | 4.363e-1 | 3.372e-1 | 4.109e-1 | **4.901e-1** | 2.820e-1 | 4.588e-1 |
| | WFG3 | 1.305e-2 | 1.959e-2 | 3.127e-2 | 4.522e-2 | 2.015e-2 | 2.474e-2 | 1.404e-2 | 1.770e-2 | 2.731e-2 | 2.192e-2 | **1.480e-2** |
| | WFG4 | 6.093e-3 | 1.376e-2 | 2.482e-2 | 1.625e-2 | 1.670e-2 | 1.393e-2 | 1.179e-2 | 1.616e-2 | 7.106e-3 | 6.128e-3 | **8.122e-3** |
| | WFG5 | 4.486e-3 | 5.460e-3 | 1.113e-2 | 5.991e-3 | 1.238e-2 | 1.214e-2 | 5.923e-3 | 7.700e-3 | 6.595e-3 | 8.747e-3 | **7.707e-3** |
| | WFG6 | 2.781e-2 | 2.185e-2 | 2.808e-2 | 4.237e-2 | 2.884e-2 | 2.929e-2 | 1.892e-2 | 2.605e-2 | **2.284e-2** | 3.355e-2 | 2.211e-2 |
| | ZDT1 | 4.235e-2 | 2.366e-2 | 7.363e-2 | 6.372e-2 | 9.843e-2 | 4.841e-2 | 6.588e-2 | 4.639e-2 | 5.339e-2 | 5.935e-2 | **3.903e-2** |
| | ZDT2 | 2.110e-1 | 1.187e-1 | 0.000e+00 | 5.900e-2 | 9.980e-2 | 1.568e-1 | 1.859e-1 | 2.358e-1 | 2.087e-1 | 1.890e-1 | **2.948e-1** |
| GD | DTLZ1 | 6.531e-2 | 1.235e-1 | 6.156e-2 | 8.824e-2 | 6.074e-2 | 7.186e-2 | 7.233e-2 | 5.622e-2 | **5.613e-2** | 6.192e-2 | 5.888e-2 |
| | DTLZ2 | 1.278e-3 | 1.508e-3 | 5.872e-3 | 6.082e-3 | 1.826e-3 | 4.585e-3 | 5.354e-3 | 6.628e-3 | 2.106e-3 | 6.396e-3 | **4.119e-4** |
| | DTLZ3 | 2.685e-1 | 3.054e-1 | 2.612e-1 | 2.941e-1 | 1.540e-1 | 2.194e-1 | 2.534e-1 | 7.796e-1 | **1.655e-1** | 2.155e-1 | 1.739e-1 |
| | WFG3 | 1.062e-2 | 8.819e-3 | 1.348e-2 | 2.321e-2 | 1.247e-2 | 1.517e-2 | 1.434e-2 | 1.478e-2 | 8.014e-3 | 1.057e-2 | **5.863e-3** |
| | WFG4 | 1.922e-3 | 2.876e-3 | 8.890e-3 | 5.296e-3 | 2.855e-3 | 2.491e-3 | 3.764e-3 | 3.882e-3 | 2.510e-3 | 2.261e-3 | **2.507e-3** |
| | WFG5 | 1.308e-3 | **4.568e-4** | 4.208e-3 | 1.966e-3 | 1.086e-3 | 1.970e-3 | 8.386e-4 | 2.036e-3 | 4.989e-4 | 1.154e-3 | 5.945e-4 |
| | WFG6 | 8.044e-3 | 5.897e-3 | 6.281e-3 | 1.142e-2 | **9.119e-3** | 8.804e-3 | 5.572e-3 | 7.655e-3 | 7.462e-3 | 9.071e-3 | 6.064e-3 |
| | ZDT1 | 1.885e-4 | 3.470e-3 | 4.414e-2 | 1.051e-2 | 4.492e-3 | 6.026e-3 | 7.011e-3 | 6.527e-3 | 7.821e-3 | 7.730e-3 | **9.093e-5** |
| | ZDT2 | **1.053e-4** | 4.012e-2 | 9.780e-2 | 2.023e-2 | 1.644e-2 | 8.844e-3 | 2.941e-2 | 7.441e-3 | 1.506e-2 | 1.929e-2 | 1.965e-4 |
| IGD | DTLZ1 | 2.475e-1 | 2.812e-1 | 2.719e-1 | 2.478e-1 | 2.554e-1 | 3.060e-1 | 2.934e-1 | 2.116e-1 | **2.583e-1** | 2.575e-1 | 2.666e-1 |
| | DTLZ2 | **3.993e-3** | 7.355e-3 | 2.783e-2 | 2.711e-2 | 1.444e-2 | 1.832e-2 | 2.952e-2 | 2.760e-2 | 8.790e-3 | 2.682e-2 | 4.773e-3 |
| | DTLZ3 | 6.598e-1 | 5.327e-1 | 9.210e-1 | 8.540e-1 | **3.791e-1** | 9.108e-1 | 9.005e-1 | 9.159e-1 | 8.933e-1 | 8.680e-1 | 9.301e-1 |
| | WFG3 | 3.816e-2 | 4.374e-2 | 1.453e-1 | 1.054e-1 | 3.172e-2 | 4.365e-2 | 3.702e-2 | 4.998e-2 | 7.694e-2 | 4.974e-2 | **2.774e-2** |
| | WFG4 | 6.587e-3 | 2.826e-2 | 2.639e-2 | 1.711e-2 | 1.748e-2 | 3.026e-2 | 9.944e-3 | **9.645e-3** | 2.153e-2 | 6.235e-3 | 4.371e-2 |
| | WFG5 | 7.053e-3 | 2.056e-2 | 5.920e-3 | 8.165e-3 | 2.436e-2 | 3.135e-2 | 5.553e-3 | **7.635e-3** | 3.528e-2 | 4.499e-3 | 2.650e-2 |
| | WFG6 | 3.156e-2 | 2.661e-2 | 2.555e-2 | 4.371e-2 | 2.541e-2 | 3.621e-2 | 1.301e-2 | **1.990e-2** | 4.049e-2 | 1.873e-2 | 3.254e-2 |
| | ZDT1 | 7.885e-2 | 1.873e-2 | 1.913e-1 | 5.795e-2 | 1.891e-1 | 4.390e-2 | 7.343e-2 | 4.106e-2 | 5.004e-2 | 6.390e-2 | **6.893e-2** |
| | ZDT2 | 4.710e-1 | 2.961e-1 | 1.623e-1 | 9.517e-2 | 2.111e-1 | 1.805e-1 | 2.692e-1 | 3.217e-1 | 3.161e-1 | 4.588e-1 | **4.248e-1** |

Table 6.2: Corresponding IQR values for Table 5.7

| | | GLMO | IBEA | MOEA/D | MOEA/DD | MOMBI-II | NSGA-II | NSGA-III | SPEA2 | SPEA2SDE | tDEA | OHHMOEA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $PF_1$ | DTLZ1 | 1.163e-2 | 5.866e-2 | 1.229e-2 | 2.186e-2 | **6.060e-3** | 1.241e-2 | 8.495e-3 | 7.897e-3 | 4.358e-3 | 3.736e-3 | 5.072e-3 |
| | DTLZ2 | 3.073e-3 | 3.310e-3 | 2.742e-2 | 1.310e-2 | 3.402e-3 | 1.426e-2 | 1.239e-2 | 1.271e-2 | 3.171e-3 | 1.296e-2 | **1.791e-3** |
| | DTLZ3 | 3.567e-2 | 4.021e-3 | 5.531e-2 | 5.927e-2 | 1.444e-2 | 1.521e-2 | 1.755e-2 | 1.231e-2 | **5.377e-3** | 2.088e-2 | 1.781e-2 |
| | WFG3 | 1.215e-2 | **7.111e-3** | 3.463e-2 | 2.820e-2 | 9.827e-3 | 2.257e-2 | 1.386e-2 | 1.538e-2 | 1.275e-2 | 1.006e-2 | 1.072e-2 |
| | WFG4 | 4.110e-3 | **3.809e-3** | 1.512e-2 | 6.971e-3 | 7.042e-3 | 8.095e-3 | 4.366e-3 | 8.046e-3 | 6.286e-3 | 4.536e-3 | 3.583e-3 |
| | WFG5 | 6.153e-3 | 8.463e-3 | 8.145e-3 | 4.806e-3 | 6.563e-3 | 7.540e-3 | 6.423e-3 | 7.708e-3 | **9.677e-3** | 8.640e-3 | 7.056e-3 |
| | WFG6 | 2.310e-2 | **2.149e-2** | 2.135e-2 | 4.150e-2 | 1.916e-2 | 1.665e-2 | 1.505e-2 | 1.315e-2 | 1.440e-2 | 1.544e-2 | 1.801e-2 |
| | ZDT1 | 9.528e-3 | 2.082e-3 | 5.667e-2 | 1.027e-2 | 3.641e-2 | 3.076e-3 | 9.462e-3 | 3.876e-3 | 3.590e-3 | 7.891e-3 | **4.781e-4** |
| | ZDT2 | 1.099e-1 | 2.028e-1 | 3.413e-2 | 1.735e-2 | 1.272e-1 | 5.335e-3 | 3.007e-2 | 9.825e-3 | 5.913e-3 | 2.536e-2 | **5.107e-4** |
| $PF_2$ | DTLZ1 | 4.618e-3 | 4.517e-2 | 4.817e-3 | 4.714e-3 | **1.578e-3** | 5.058e-3 | 2.042e-3 | 2.120e-3 | 2.155e-3 | 1.814e-3 | 1.530e-3 |
| | DTLZ2 | 1.369e-3 | 7.901e-4 | 9.555e-3 | 5.209e-3 | 1.521e-3 | 8.599e-3 | 5.477e-3 | 4.111e-3 | 2.341e-3 | 5.762e-3 | **1.022e-3** |
| | DTLZ3 | 6.652e-3 | 7.731e-4 | 1.468e-2 | 8.809e-3 | 1.261e-3 | 5.811e-3 | 3.987e-3 | 3.841e-3 | **1.668e-3** | 2.727e-3 | 3.409e-3 |
| | WFG3 | 8.047e-3 | **6.511e-3** | 3.739e-2 | 2.045e-2 | 8.137e-3 | 1.479e-2 | 1.115e-2 | 9.952e-3 | 8.885e-3 | 9.112e-3 | 7.265e-3 |
| | WFG4 | 3.019e-3 | **1.973e-3** | 1.162e-2 | 4.217e-3 | 4.152e-3 | 5.739e-3 | 3.659e-3 | 5.431e-3 | 1.960e-3 | 3.079e-3 | 4.068e-3 |
| | WFG5 | 5.070e-3 | **1.047e-3** | 3.627e-3 | 7.699e-3 | 6.615e-3 | 6.155e-3 | 5.750e-3 | 7.389e-3 | 1.272e-3 | 7.419e-3 | 3.401e-3 |
| | WFG6 | 1.202e-2 | **1.462e-2** | 2.191e-2 | 1.731e-2 | 1.327e-2 | 1.275e-2 | 1.421e-2 | 1.583e-2 | 1.496e-2 | 1.147e-2 | 1.492e-2 |
| | ZDT1 | **2.987e-4** | 4.325e-4 | 1.933e-2 | 4.255e-3 | 2.845e-3 | 1.258e-3 | 3.668e-3 | 1.009e-3 | 1.064e-3 | 2.294e-3 | 1.051e-4 |
| | ZDT2 | **7.507e-5** | 2.265e-2 | 1.634e-1 | 6.542e-3 | 2.427e-3 | 1.847e-3 | 5.370e-3 | 1.803e-3 | 1.567e-3 | 3.642e-3 | 8.898e-5 |
| HV | DTLZ1 | 1.565e-4 | 3.448e-3 | 8.075e-4 | 5.265e-4 | 1.520e-2 | 6.190e-4 | 1.465e-4 | **1.565e-4** | 1.789e-3 | 1.700e-4 | 4.510e-4 |
| | DTLZ2 | 1.530e-3 | 5.248e-3 | 5.505e-2 | 2.431e-2 | 9.229e-3 | 2.652e-2 | 3.130e-2 | 1.951e-2 | 1.629e-2 | 2.570e-2 | **1.279e-3** |
| | DTLZ3 | 8.610e-4 | **1.181e-3** | 6.135e-4 | 8.130e-4 | 2.258e-3 | 6.633e-3 | 6.685e-4 | 3.801e-3 | 2.254e-3 | 6.710e-4 | 3.545e-3 |
| | WFG3 | 7.105e-3 | **8.893e-3** | 2.932e-2 | 2.641e-2 | 1.126e-2 | 7.343e-3 | 8.484e-3 | 1.220e-2 | 6.391e-3 | 8.770e-3 | 6.269e-3 |
| | WFG4 | 6.353e-3 | **7.696e-3** | 6.558e-2 | 1.603e-2 | 1.191e-2 | 2.089e-2 | 9.940e-3 | 1.618e-2 | 8.204e-3 | 1.368e-2 | 9.958e-3 |
| | WFG5 | 3.046e-3 | **3.057e-3** | 3.263e-2 | 5.700e-3 | 1.742e-2 | 1.741e-2 | 6.348e-3 | 1.620e-2 | 6.308e-3 | 3.485e-3 | 4.803e-3 |
| | WFG6 | 3.612e-2 | 3.006e-2 | 3.587e-2 | 3.971e-2 | 2.293e-2 | 5.789e-2 | 3.129e-2 | 3.349e-2 | 2.858e-2 | 2.359e-2 | **3.199e-2** |
| | ZDT1 | **3.637e-4** | 5.290e-3 | 8.310e-2 | 2.865e-2 | 5.566e-3 | 5.608e-3 | 2.075e-2 | 1.000e-2 | 7.536e-3 | 9.072e-3 | 2.034e-4 |
| | ZDT2 | **2.364e-4** | 2.642e-1 | 6.786e-3 | 3.095e-2 | 1.245e-2 | 1.386e-2 | 3.375e-2 | 1.280e-2 | 1.278e-2 | 1.335e-2 | 3.475e-4 |

Table 6.3: Corresponding IQR values for Table 5.10

| | | GLMO | IBEA | MOEA/D | MOEA/DD | MOMBI-II | NSGA-II | NSGA-III | SPEA2 | SPEA2SDE | tDEA | OHHMOEA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MOP | DTLZ4 | 2.070e-1 | 1.828e-1 | 2.420e-1 | 2.144e-1 | 1.935e-1 | 3.974e-1 | 2.002e-1 | 4.187e-1 | 4.312e-1 | 1.430e-2 | **1.896e-1** |
| | DTLZ5 | 3.678e-3 | 1.399e-3 | 1.089e-3 | 1.040e-3 | 1.136e-4 | 1.252e-3 | 3.335e-3 | 1.745e-3 | **8.235e-4** | 3.905e-3 | 9.996e-4 |
| | DTLZ6 | 1.291e-1 | **4.404e-2** | 0.000e+00 | 0.000e+00 | 1.028e-1 | 1.556e-2 | 3.133e-2 | 1.598e-1 | 9.298e-2 | 1.418e-1 | 1.416e-1 |
| | DTLZ7 | 2.505e-2 | 2.890e-2 | 6.096e-2 | 3.926e-3 | 3.203e-2 | 4.027e-2 | 5.883e-2 | 4.685e-2 | **2.458e-2** | 3.500e-2 | 2.658e-2 |
| | WFG1 | 3.833e-2 | **5.124e-2** | 6.939e-2 | 3.171e-2 | 1.367e-1 | 5.366e-2 | 8.944e-2 | 6.988e-2 | 5.692e-2 | 7.214e-2 | 8.374e-2 |
| | WFG2 | 7.764e-3 | **1.502e-2** | 6.734e-2 | 2.580e-2 | 6.346e-2 | 1.889e-2 | 1.846e-2 | 2.367e-2 | 3.565e-2 | 2.797e-2 | 1.235e-2 |
| | WFG7 | 2.356e-2 | 5.642e-3 | 4.073e-2 | 4.398e-2 | 1.245e-2 | 6.257e-2 | 3.040e-2 | 4.118e-2 | **5.304e-3** | 1.024e-2 | 5.548e-3 |
| | WFG8 | 7.775e-3 | **6.901e-3** | 2.494e-2 | 2.622e-2 | 6.745e-3 | 1.262e-2 | 9.024e-3 | 9.283e-3 | 8.488e-3 | 1.257e-2 | 7.239e-3 |
| | WFG9 | 5.600e-2 | 1.192e-2 | 7.771e-2 | 8.610e-2 | 2.642e-2 | 6.619e-2 | 7.739e-2 | 4.804e-2 | 1.546e-2 | 5.088e-2 | **2.404e-2** |
| MVP | DTLZ4 | **1.721e-1** | 1.612e-1 | 2.156e-1 | 4.076e-2 | 1.373e-2 | 9.040e-2 | 2.364e-2 | 2.112e-1 | 2.301e-1 | 1.209e-1 | 1.722e-1 |
| | DTLZ5 | 5.962e-3 | 5.341e-3 | 4.258e-2 | 2.921e-2 | 9.313e-3 | 3.814e-2 | 3.003e-2 | 3.327e-2 | 7.490e-3 | 3.560e-2 | **2.323e-3** |
| | DTLZ6 | 0.000e+0 | 0.000e+0 | 0.000e+0 | 0.000e+0 | 0.000e+0 | 0.000e+0 | 0.000e+0 | 0.000e+0 | 0.000e+0 | 0.000e+0 | 0.000e+0 |
| | DTLZ7 | 1.963e-2 | 6.744e-2 | 1.617e-2 | 1.952e-2 | 7.744e-2 | 3.095e-2 | 1.659e-2 | 1.491e-2 | 4.380e-2 | 3.324e-2 | **2.532e-2** |
| | WFG1 | 1.427e-2 | **3.253e-2** | 8.126e-2 | 2.162e-2 | 4.265e-2 | 2.271e-2 | 1.782e-2 | 1.681e-2 | 1.839e-2 | 1.929e-2 | 1.886e-2 |
| | WFG2 | 1.458e-2 | 2.431e-2 | 6.420e-2 | 4.108e-2 | 4.580e-2 | 6.788e-2 | 6.369e-2 | 5.627e-2 | 3.654e-2 | 8.869e-2 | **1.461e-2** |
| | WFG7 | 3.227e-2 | **2.218e-2** | 3.193e-2 | 2.911e-2 | 2.220e-2 | 1.563e-2 | 2.130e-2 | 3.510e-2 | 2.652e-2 | 3.827e-2 | 3.530e-2 |
| | WFG8 | 1.282e-2 | 9.976e-3 | 1.997e-2 | 2.558e-2 | 1.741e-2 | 6.095e-3 | 1.359e-2 | 9.575e-3 | 9.306e-3 | 1.444e-2 | **1.284e-2** |
| | WFG9 | 5.092e-2 | **2.989e-2** | 3.448e-2 | 5.831e-2 | 2.204e-2 | 1.032e-1 | 1.048e-1 | 8.368e-2 | 2.748e-2 | 6.269e-2 | 3.320e-2 |

# Declaration of Authorship

I hereby declare that this thesis was created by me and me alone using only the stated sources and tools.

Hans-Martin Wulfmeyer　　　　Magdeburg, December 31, 2021