Otto-von-Guericke University Magdeburg

Faculty of Computer Science

Master Thesis

Adaptive Crossover Operators in Evolutionary Algorithms using Online Learning Hyper-Heuristics

Author:

Julia Heise

February 12, 2022

Supervisor:

Prof. Dr.-Ing. habil. Sanaz Mostaghim
Chair of Computational Intelligence
Otto-von-Guericke University Magdeburg

Heise, Julia:
Adaptive Crossover Operators in Evolutionary Algorithms using Online Learning Hyper-Heuristics
Master Thesis, Otto-von-Guericke University Magdeburg, 2022.

Abstract

In this work, we present Hyper-Heuristic Online Learners as Selectors for crossover operators in multi-objective evolutionary algorithms. We answer whether Hyper-Heuristics are suitable for this task, by designing different Hyper-Heuristics and evaluating them by different means.

We first examine known implementations of crossover operators and Hyper-Heuristics. We gain information about the different components of Hyper-Heuristic Online Learner, namely the selection heuristic, the reward function and the selection pool. We chose two different selection heuristics, the roulette wheel selection and the generation distribution, and four different reward function. Therefore, we design eight different Hyper-Heuristics, which are afterwards experimental evaluated by benchmark tests.

In our evaluation, we compare the quality of the results of the Hyper-Heuristics and single crossover operators and evaluate afterwards in-depth the learning and selection behaviour of the Hyper-Heuristics. Throughout those analyses, we note that crossover operators perform dependent on more circumstance, than only the problem's properties, but also the current distribution of the population and the phase of the algorithm. Thus, crossover operators can achieve an enormously better result, when utilized in combination.

After those benchmark tests and analyses, we conclude that Hyper-Heuristics are not only suitable as crossover operators selectors, but improve also the quality in nearly all cases and give a more general approach that can be used for a wider range of problems.

Acknowledgments

I want to thank my advisor, Dr. Heiner Zille, who supported me throughout my whole thesis, although we could only meet remotely due to the pandemic situation.

I also want to thank my proofreaders, Arne Herdick, Ulrich Bätjer, Markus Hempel and Olja Mozheiko, for finding all my typos.

And lastly, I want to thank my employers, Sven Beckmann and Marcel Hesse, for giving me all the time I needed to finish this thesis.

Contents

List of Figures

List of Tables

List of Code Listings

List of Acronyms

$\mathcal{PF}$	Pareto Front
BLX	Blend Crossover
CD	Crowding Distance
CMA-ES	Covariance Matrix Adaptation Evolutionary Strategy
CMAX	Covariance Matrix Adaption Crossover
DE	Differential Evolution Crossover
DM	Decision Maker
EA	Evolutionary Algorithm
FE	Function Evaluation
HHD	Hyper-Heuristic Distributor
HHS	Hyper-Heuristic Selector
HV	Hyper Volume
IGD	Inverted Generational Distance
IQR	Interquartile Range
LCX	Linear Combination Crossover
LX	Laplace Crossover
MOEA	Mulit-Objective Evolutionary Algorithm
MOP	Multi-objective problem
NC	Non-Dominated Sorting + Crowding Distance
NDS	Non-Dominated Sorting
NFL	No-Free-Lunch
NSGA-II	Non-Dominated Sorting Genetic Algorithm II
PO	Pareto Optimal
RSBX	Rotation-Based Simulated Binary Crossover
SBX	Simulated Binary Crossover
SPX	Simplex Crossover
UX	Uniform Crossover

1. Introduction

Many real-world optimization problems require a human decision maker to find their best compromise. For everyday decisions, decision makers can simply compare their possibilities, but the more complex a problem is, the more possibilities need to be compared. In numerous instances, the considerations should not be done manually and a computational pre-evaluation is necessary. Those can be implemented with EAs. These algorithms are Meta-Heuristics, that can be applied to different problems without more profound knowledge. In this work, the focus lies on multi-objective problems (MOPs). Those have more than one conflicting optimization goal and therefore, compromises need to be made. For this case, Mulit-Objective Evolutionary Algorithm (MOEA) are utilized. They can compute the solutions that give the best compromises, namely the Pareto Optimal (PO) Solutions.

Usually, EAs consist of four different operators. To perform an evolution, a set of parent individuals needs to be selected, crossover and mutation operators create and modify offspring and lastly a second selection selects the best individuals to form the next generation. Due to this modular structure, all those components can be exchanged, and different MOEAs were already developed (NSGA-II [2], NSGA-III [5], MOEA/D [6] for instance).

Those algorithms are widely used in many fields like designing schedules for employees and timetables for universities and schools, constructing networks, planning transportation in terms of path finding and scheduling, building energy application system and maintaining plans or optimization of business processes or inventory location [7]. Although Meta-Heuristics, like MOEAs, are meant to solve any of those problems, there is no algorithm that performs best in all cases. There is always a tradeoff: If an algorithm is improved on one problem class, it will be impaired on another problem class. This is the No-Free-Lunch (NFL), which was already stated in 1995 by Wolpert et Al. [8, 9]. With an increasing number of possible applications and problems, the number of variants of MOEAs increased too. According to that, a human decision maker needs to select the algorithm that might fit best to the current problem. As the decision maker does may not have the knowledge about the problem's classification nor about the interaction between this and the algorithm's

features, the idea of Hyper-Heuristics on MOEAs arose. Hyper-Heuristics form a computational decision maker for the selection of an algorithm or its operators, that can either learn from gathered information of the current optimization (online) or from previous optimization (offline). A few studies were already carried out in this field. The results from those studies underline the fact, that each of the algorithm's components affect the quality differently depending on the problem's properties. Furthermore, they show that Hyper-Heuristics on MOEAs can improve the solution's quality and create algorithms that perform well on a wider subset of problem classes.

This work extents the researches in the field of crossover operators and Hyper-Heuristics on MOEAs. The crossover operator is responsible for the offspring creation. Consequently, the essential properties of the next generation are based on the methods of the crossover operator. Therefore, the selection of the crossover operator is already crucial for the quality of the results. Our overall goal is to examine, whether Hyper-Heuristic are suitable as selectors of crossover operators and if this improves the quality of the solutions produced by the MOEA. To achieve this goal, we stated the following subgoals:

1. Research on existing crossover operators and Hyper-Heuristics on MOEAs to identify their components
2. Design new variants of those components and combine them to propose different versions of Hyper-Heuristic online selectors for crossover operators
3. Perform an experimental evaluation with the selected crossover operators and identify their interplay with different problem properties
4. Perform an experimental evaluation with the proposed Hyper-Heuristics and identify the influence of the components on the result's quality
5. Readjust parameters of selected Hyper-Heuristics and analyse the result's quality to show, that further refinements are possible
6. Choose the best performing Hyper-Heuristics and compare them to NSGA-II with single crossover operators

We implement different single crossover operators for NSGA-II, an exemplary MOEA, and compared the results on different problem classes. We developed 8 new Hyper-Heuristics by combing different selection and scoring heuristics. The Hyper-Heuristics firstly scores all available operators according to survival rate, domination criteria or improvement rate of generated offspring. Afterwards, either one operator is picked by Roulette Wheel selection with probabilities proportional to score or the parent generation is distributed between the operators proportionally to the score. Additionally, we implement a uniform scoring function, so that the selection probability or the number of parents per operator is always the same. We evaluate the quality of Hyper-Heuristics and examine the selection behaviour. By comparing these measurements with the quality of single crossover operators, we can make statements about the influence of the scoring and selection heuristics in Hyper-Heuristics. Finally, we show, that the presented Hyper-Heuristics have again adjusting parameter, that affect the quality of the results.

Work structure

In the following chapter, the necessary information is clarified, and similar researches are presented to put this work into context. In chapter 3, the selected crossover operators and the different implementation of Hyper-Heuristics are explained in detail. In chapter 4 the results of experiments, done to verify the improvements made on evolutionary algorithms, are presented and discussed. In the end, in chapter 5, the work is summarized and an outlook on future works is given.

2. Background & Related Work

In this chapter, the mathematical background to this work is clarified and put into context. First, the definition of MOPs and the approach of solving those with EAs as meta-heuristics is illuminated. Afterwards, different crossover operators and their properties are introduced. Subsequently, basic methods to evaluate the quality of EAs are explained. To finalize, the idea of Hyper-Heuristics and few exemplary works on this idea are presented.

Annotations

In this work, the focus lies on crossover operators, which take different solutions to a problem as parents to form new solutions as offspring. In the context of EAs, solutions are often referred to as individuals and a set of individuals produced in one iteration is a generation. One individual contains different attributes, which are called genes. For simplification, the set of parents, the current generation, is named $\mathcal{X}$. The set contains vectors, which are the N individuals $\boldsymbol{x}_i \,\forall\, i = 1 \ldots N$ with their D genes $x_{ij} \,\forall\, j = 1 \ldots D$. When talking about the individuals, only bold letters are used to signal that they are vectors in a mathematical sense. Their offspring set, the next generation, is, therefore, $\mathcal{Y}$, the individuals corresponding $\boldsymbol{y}_i \,\forall\, i = 1 \ldots N$ and their genes $y_{ij} \,\forall\, j = 1 \ldots D$.

The optimization goal in this work is usually to minimize the M objective functions noted as $f_{1:M}(\boldsymbol{x}_i)$.

2.1 Multi Objective Problems

When attempting to optimize decision-making processes, there is always a definition of an optimization problem to solve. In the field of computer science this usually means, that there is a mathematical definable problem which is either single-objective, which means there is one optimization goal, multi-objective hence means there are two or three conflicting optimization goals or many-objective which means there are more than three conflicting optimization goals.

Definition 2.1: Multi-Objective Minimization Problem

$$\begin{aligned} \min \quad & \boldsymbol{f} = \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})\} \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathcal{S} \\ & \boldsymbol{g}(\boldsymbol{x}) \leq 0 \\ & \boldsymbol{h}(\boldsymbol{x}) = 0 \end{aligned}$$

The Definition 2.1 is the general form of a MOP, with M conflicting objective functions. Those functions transform the decision vector $\boldsymbol{x}$ from search space $\mathcal{S}$ to the objective space $\mathcal{M}$ ($\boldsymbol{f} : \mathcal{S} \rightarrow \mathcal{M}$). Additionally, inequality constraints $\boldsymbol{g}$ and equality constraints $\boldsymbol{h}$ can be defined. Those constraints limit the feasibility of solutions. A solution is considered as feasible if it fulfils the constraints. In case that the solution space and search space are equal, the decision vector is also a solution. Otherwise, the decision vector is a decoded solution, that can be transformed to solution space with the encoding function. In this work, the solution space is of no interest, and we treat all decision vectors as solutions.

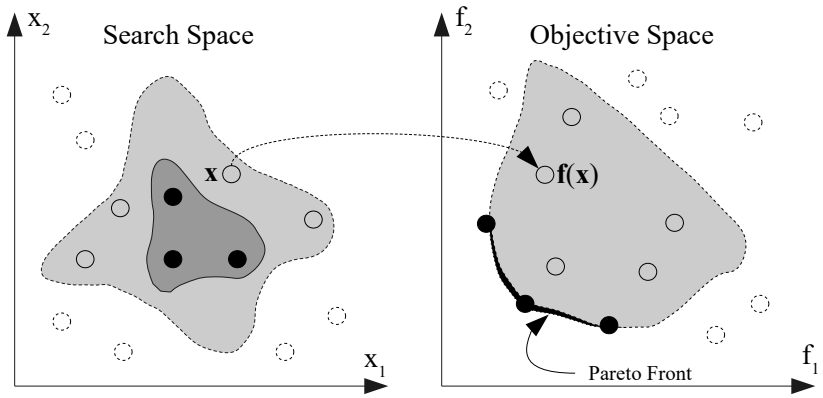


Figure 2.1: Exemplary visualization of Search and Objective Space[1]

In Figure 2.1 the relationship between search space and objective space is visualized. On the left side is the search space with its current solutions or decision vectors $\boldsymbol{x}$. Those individuals are transformed to the objective space on the right side using the functions $\boldsymbol{f}$. The lighter area in the dotted line shows the feasible region in both spaces. This region is defined by the constraint functions, if given. Every solution outside the feasible region does not fulfil the constraint and is not accepted. Those constraints can also be transformed to the objective space by $\boldsymbol{f}$. In Figure 2.1 the lighter area in the objective space is also the feasible area.

Due to those conflicting objective functions, it is not possible to optimize one objective without a tradeoff on another objective. Thus, the result of those optimization problems are sets of solutions with the best compromises. Whether those solutions

are better than the others, is decided by the PO criterion in Definition 2.2. A solution is better if it dominates another. The best solutions are non-dominated.

Definition 2.2: Pareto Domination

A solution $\boldsymbol{x}_1$ Pareto dominates $\boldsymbol{x}_2$ if and only if

$$f_i(\boldsymbol{x_1}) \leq f_i(\boldsymbol{x_2}) \; \forall \; \mathrm{i} = 1,...,\mathrm{l}$$
$$f_j(\boldsymbol{x_1}) < f_j(\boldsymbol{x_2}) \; \exists \; \mathrm{j} = 1,...,\mathrm{l}$$

The domination of $\boldsymbol{x}_1$ over $\boldsymbol{x}_2$ is here symbolized with $\boldsymbol{x}_1 \prec \boldsymbol{x}_2$. Therefore, the evaluation of solutions takes place in the objective space. The thicker line on the lower left border of the feasible region in objective space in Figure 2.1 shows the Pareto Front ($\mathcal{PF}$). Those solutions are non-dominated after Definition 2.2, thus they are under the current knowledge the best. If there are no other feasible solutions, that dominate the current solutions, those are the Pareto Optimal solutions after Definition 2.3.

Definition 2.3: Pareto Optimality

A solution $\boldsymbol{x}^* \in \mathcal{S}$ is PO if there exists no $\boldsymbol{x} \in \mathcal{S}$ where

$$\boldsymbol{x} \prec \boldsymbol{x}^* \text{ and } \boldsymbol{x} \neq \boldsymbol{x}$$

There are different possible solving methods for problems of that kind. For example, could there be a direct way, a so-called heuristic, to solve a specific problem. If there are multiple problems or problem classes to solve, using a meta-heuristic is more beneficial. Those are generic, as the solving process is independent of the problem itself. Thus, it can be applied to problems without knowing their properties. EAs, for instance, are meta-heuristics.

2.2 Evolutionary Algorithms

The goal of EAs is to find this set of PO solutions. This is done by enduring a loop of operations that is repeated as long as no termination conditions like a maximum number of repetitions or a stagnation of solutions quality is met.

As the name says, EAs are inspired by biological evolution. Every species experiences the generational life cycle, where new hurdles are taken in different ways and only the individuals with the best approaches survive. Thereby, those approaches or solutions prevail to the upcoming generations. EAs adapt this evolutionary process and its so-called "survival of the fittest" to approximate the best solution for a problem. In this work, those best solutions are the PO solutions of MOPs. This adapted life cycle is visualized in Figure 2.2.

To initialize the population, the solutions are, for example, randomly or greedy set. The quality of those solutions are measured in the next step and the termination condition is checked. If it is not met, a first selection step is processed using the quality measured before. The selected solutions or individuals are used as a parent

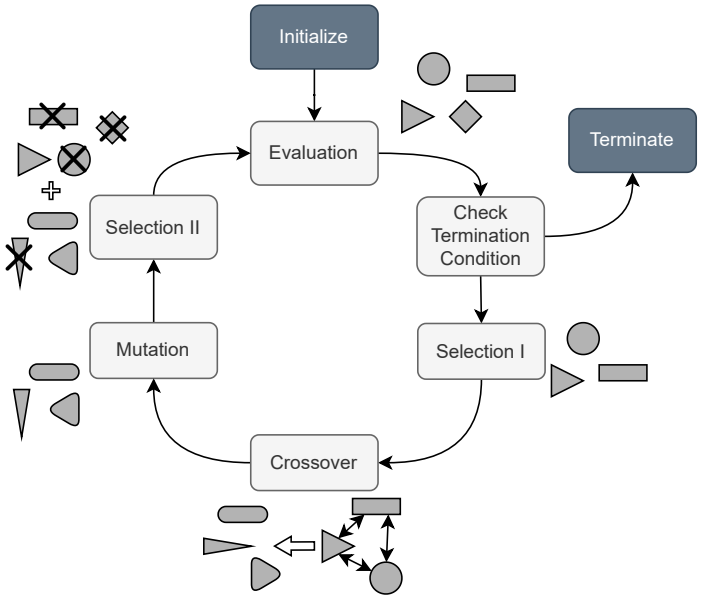


Figure 2.2: Scheme of a basic EA. The geometric forms illustrate the population during the different steps.

generation. They create an offspring generation by using the crossover operator. In general, the parent solutions are divided into pairs or groups, that mix their genes to create their children. Mostly, the parent generation and the offspring generation are equally sized. Afterwards, the mutation operator makes sure that some of these children have newer properties by randomly modifying their genes. Subsequently, a second selection algorithms selects the survivors of this generation from the set of either both, parents and children, or only children to form the next generation.

The concept applies to all kinds of EA, thus as well for MOEA. Since this work focuses on MOPs, we continue with only MOEAs.

There are different combinations of parent selection, crossover, mutation operators, and environmental selection that build an MOEA. One of those is the Non-Dominated Sorting Genetic Algorithm II (NSGA-II), which is mostly used in this work. This algorithm and other optimization approaches, that are of interest for this work, are described in the next subsections.

2.2.1 NSGA-II

The NSGA-II developed by Deb et Al. [2] is a MOEA based on Pareto dominance after Definition 2.2 of the individuals. As all MOEAs, it consists of the four basic components. However, the most significant part of NSGA-II is the second selection. Common implementations of NSGA-II utilize a Tournament Selection, Simulated Binary Crossover, Polynomial Mutation and a selection, that combines Non-dominated Sorting (NDS) and Crowding Distance (CD).

Non-Dominated Sorting

The NDS is, as the name says, an algorithm to sort individuals after their Pareto Dominance. The Algorithm 2.1 shows, that in the beginning all individuals are

compared to all other individuals in terms of Pareto Dominance. The individuals $\boldsymbol{q}$ that are dominated by another individual $\boldsymbol{p}$ are saved in its corresponding subset $\mathcal{S}_p$. The number of individuals that are dominating $\boldsymbol{p}$ is saved in n_p. All individuals that are not dominated by any other individual, hence n_p is zero, are added to the first front.

The individuals that belong to the first front are here considered as the current best solutions. Those are desired to survive and thus are added to the next generation. If there are not enough individuals in the first front, to satisfy the desired size of the next generation, the next front needs to be considered.

Algorithm 2.1: Non-Dominated Sorting.

```
input
    P   *set of solutions
begin
    *identify first front
    foreach p ∈ P
        S_p = {},
        foreach q ∈ P
            if p ≺ q then
                S_p = S_p ⋃ {q}
            else
                n_p = n_p + 1
            end if
        end loop

        if n_p == 0 then
            p_rank = 1
            F_1 = F_1 ⋃ {p}
        end if
    end loop

    *identify the fronts of the leftover individuals
    i = 1
    while F_i ≠ {}
        Q = {}
        foreach p ∈ F_i
            foreach q ∈ S_p
                n_q = n_q − 1
                if n_q = 0 then
                    q_rank = i + 1
                    Q = Q ⋃ {q}
                end if
            end loop
        end loop
        i = i + 1
        F_i = Q
    end loop
end
```

In the second part of the Algorithm 2.1 the procedure is explained. The first front contains all non-dominated solutions. Each $\boldsymbol{p}$ of those solutions has a subset $\mathcal{S}_p$ with all individuals $\boldsymbol{q}$ that are dominated by $\boldsymbol{p}$. If $\boldsymbol{q}$ is not dominated by any other solution besides those, that are in the first front, $\boldsymbol{q}$ is added to the second front. In the next iteration, this is repeated for the individuals in the second front. This continues until all individuals are sorted into fronts.

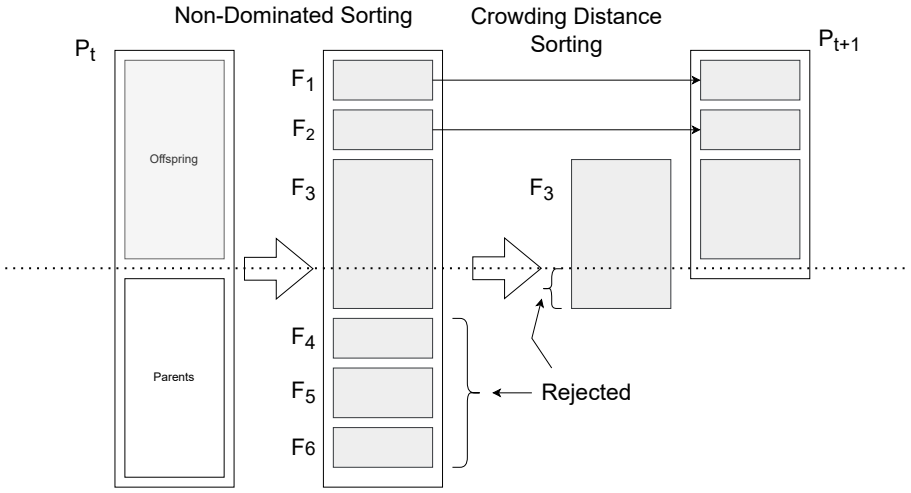


Figure 2.3: Visualization of Selection procedure in NSGA-II.

If there are not enough individuals in the first front, more individuals are selected from the next front until the desired size of the next generation is reached. The whole process of the NSGA-II is visualized in Figure 2.3. The current generation is first divided into fronts and each front is revised whether it is selected as a whole or it needs to be shrunk before. In the latter case, CD is used to decrease the number of individuals in the front until it is sufficiently small to fit in the next generation. The CD is described in the next paragraph.

Crowding Distance

The CD described by Deb et Al. [2] is a metric to determine the density of a set of solutions. If the solutions are crowded at some point in objective space, it can be calculated, which one could be sorted out. The Definition 2.4 says, to calculate the CD of one individual $\boldsymbol{x}_i$, the two neighbouring individual in each objective are identified and the distances between those are summed over all objectives.

Definition 2.4: Crowding Distance

For each individual $\boldsymbol{x}_j$ and objective f_k:

$$CD_i = \sum_{k=1}^{M} |(f_k(\boldsymbol{x}_{i+1}) - f_k(\boldsymbol{x}_{i-1})|$$

Where $\boldsymbol{x}_{i-1}$ and $\boldsymbol{x}_{i+1}$ are the two individuals with the least distance in the k'th objective.

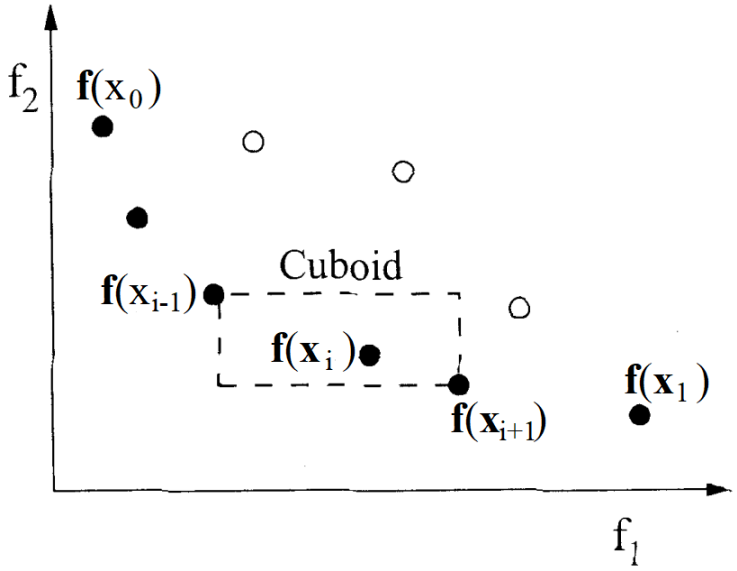


Figure 2.4: Calculation of Crowding Distance by Deb et Al. [2]

In Figure 2.4 the calculation for two objectives, $M = 2$, is visualized. In that case, the neighbouring individuals can be identified as the ones that are next to the individual in objective space. For higher M, each objective function is considered separately. The individuals, that are on the edges of the Pareto front, do not have two neighbouring individuals, and their CD is usually set to ∞.

The diversity of solutions is an indication of the information the whole generation contains. With wider distributed solutions in objectives space, the search area is increased and optima are found easier. Therefore, individuals that are similar to others and have a lower CD, carry less information than other with a higher CD. For NSGA-II this means, that after this calculation, the population can be stocked up with the solutions with the highest CD.

2.2.2 Covariance Matrix Adaptation Evolutionary Strategy

Evolutionary Strategies are a subgroup of EAs, which focus on numerical problems. The Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES), introduced by Hansen et Al. [10], is generally used for Single-Objective Problems, but can be extended to the use on MOPs. It uses the population as an entirety and produces new solutions by sampling with multivariate normal distribution. For this distribution, a mean point and a covariance matrix is needed. Both of them are updated in each iteration with the information the sampled individuals offer. Thus, the covariance matrix moves in the form of an ellipsoid over the search space.

Algorithm 2.2: Covariance Matrix Adaptation Evolutionary Strategy.

```
input
    x_c ,        *centre point
    σ ,          *step size
    c_cov ,      *learning rate
    N,           *number of individuals
    τ            *number of individuals to sample
```

initialize
$\boldsymbol{C} = \boldsymbol{I}$, *Covariance Matrix, initialized with Identity
begin
while termination condition not met
$\mathcal{Y} = \boldsymbol{x}_c + \sigma\mathcal{N}(\boldsymbol{0}, \boldsymbol{C}, \tau)$ *normal distribution, m samples
$\mathcal{Y}' = \text{Sort}(f(\mathcal{Y}))$ * f is the single objective
$\boldsymbol{y}_w = \sum_{i=1}^{N} w_i \boldsymbol{y}_i$ *$\boldsymbol{y}_i \in \mathcal{Y}'$
$\boldsymbol{x}_c = \boldsymbol{x}_c + \sigma \boldsymbol{y}_w$
$n_w = 1/(\sum_{i=1}^{N} w_i^2)$
$\boldsymbol{C} = (1 - c_{cov})\boldsymbol{C} + c_{cov} n_w \boldsymbol{y}_w \boldsymbol{y}_w^T$
end
end

To update the mean and the covariance matrix, the last produced individuals are evaluated and the best N individuals are selected, by sorting according to the objective values. With those individuals, the mean and the covariance matrix can be updated as described in Algorithm 2.2.

The selection of the best individuals ensures, that the population as a whole moves towards an optimum. Other evolutionary strategy are summarized and evaluated by Kruse et Al. [11].

2.2.3 Differential Evolution

The Differential Evolution is presented with two different schemes by Storn et Al. in [12]. In this work, when referring to Differential Evolution, the DE Scheme 1 is meant. This algorithm selects three random individuals $(\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2}, \boldsymbol{x}_{r3})$ of the parent population and combines them as follows:

$$\boldsymbol{y}_{r1} = \boldsymbol{x}_{r1} + F \cdot (\boldsymbol{x}_{r3} - \boldsymbol{x}_{r2}) \tag{2.1}$$

F is a weight to determine the length of the vector that is added to the main individual. This part can be compared to the workwise of a crossover operator. Afterwards, $\boldsymbol{y}_{r1}$ is modified by randomly deciding, which genes should be taken from $\boldsymbol{y}_{r1}$ as calculated before and which should be taken from $\boldsymbol{x}_{r1}$. This part reminds of a mutation operator. For MOPs, $\boldsymbol{x}_{r1}$ is replaced by $\boldsymbol{y}_{r1}$, if $\boldsymbol{y}_{r1}$ dominates $\boldsymbol{x}_{r1}$. The process is repeated until an entire new generation is produced.

After these explanations of the basic concepts behind EAs the focus of this work, the crossover operators, are presented in detail. In the following section, the relevant operators and their properties are introduced.

2.3 Crossover Operators

This section starts with a brief description of the concept of crossover operators and the presentation of three important properties and ends with an introduction of all crossover operators, that are used in this work.

As previously mentioned, EAs consist of four components: Parent selection, crossover operation, mutation operation and environmental selection. The two selection components are usually defined by the algorithms, whereas crossover and mutation can be exchanged. In this work, the effects on exchanging the crossover operator is examined, and therefore the focus lies on those.

Crossover operators recombine the genes of the parent individuals to form new individuals, the offspring. Those recombinations can for example be selection strategies of value per gene, a mathematical formula to calculate a new value per gene or transformations of a parent based on the genes of the other parents. There are no limitations in numbers of parents per production, although many operators use two parents. Usually, it is desired to form as many new individuals as parents given, and thus the operation is repeated with randomized parents until the desired amount is reached, or the operator is designed in a way, that divides the parent population in parent groups and produce the same number of new individuals as the number of individuals in this parent group. The latter usually leads to a specific pattern of offspring, which is, in this work, also called the production scheme. Some properties of the crossover operators can be discovered by examining this production scheme.

In the next subsection, some properties of crossover operators that are interesting for this work are introduced. Afterwards, a selection of crossover operators is presented.

2.3.1 Related Work on Properties of Crossover Operators

Crossover operators produce solutions by mixing the genes of parent solutions. Thus, the properties of the offspring generation highly depend on the parents genes and on the workwise of the operator. The calculation of new genes for the offspring can cause different artefacts or specific behaviour that is mirrored to the solving process of the algorithm. We present three exemplary properties that change the quality of the solutions depending on the properties of the problem.

Rotationally Invariance

A rotated problem, is a problem where the objective functions do not only conflict but are also linked. This happens when the objective functions depend on multiple decision variables and those decision variables also appear on multiple objective functions. This is also called variable linkage. In those problems, the optimal solutions are not parallel to any decision variables axis in solutions space [13].

A crossover operator that is rotationally invariant is not affected by this rotation. Difficulties occur when the crossover operator produces offspring along the decision space axis. Those operators are not rotationally invariant and cannot process the information about the correlation. A well-known example for not rotationally invariant crossover operators is the Simulated Binary Crossover (SBX). Although it is one of the widest used operators, it performs very poorly on rotated problems when compared to operators that are rotationally invariant [14].

Self-Adaptiveness

Self-Adaptiveness is a specific adaptive behaviour, a crossover operator can display. It was first mentions by Beyer at Al. in [15]. It is abbreviated from evolutionary

strategies, that are optimization algorithms specified for numerical single objective problems. For those algorithms, the step size is crucial: When it is too high, the solutions could oscillate around the optimum without reaching it, if it is too low, it slows down the algorithms remarkably. Self-Adaptiveness, in this case, is the ability to let the algorithm adjust the step size during the processing.

This idea is transferred to EAs where the step size from evolutionary strategies is comparable with other parameters that determine the new solutions and their difference to the parent solutions. Therefore, a Self-Adaptive crossover operator can regulate itself and adjust its parameters so that the offspring do not differ too much from the parents, when near an optimum. However, the optima a rarely known, so the operator needs to use another method to determine when it should adapt its parameters. For example, when the parents get nearer to each other, it could indicate that there is in optimum in between them. Thus, one idea to add adaption is to consider the distance between the parents and create a dependency that shrinks the distance between the offspring and the parents when the distance between the parents is decreasing. One step further in this direction: instead of including the parent distance, include the distribution of the whole parent generation.

Considering adaptive behaviour, Beyer et Al. pointed out, that there are more conditions a crossover operator should meet, when it is Self-Adaptive. The movement of the population as a whole to a certain optimum is not the responsibility of a crossover operator. The movement arises, when the generation is evaluated and only good solutions are kept. The crossover operator should be used to search the current area, thus disperse new offspring around the search area or concentrate in the current point of interest. A Self-Adaptive crossover operator can differentiate between those two states, but it should still not move the population as a whole. Thus, another important feature of a crossover operator is the Centric Reproduction. By keeping the population centre the same during crossover operation, random movements are avoided. Beyer et Al. also mentions the importance of an increase in variance in each step. In this work, this is always given with a mutation operator, that is the same for all implementations and thus not crucial for our work.

Centric Reproduction

The last feature is the ability to keep the centre of the population at the same point. There are two different methods that were introduced by Deb et Al. in [16].

1. Parent Centric behaviour
2. Mean Centric behaviour

Parent Centric crossover operators create offspring symmetrically around the parents. The mean of these offspring is therefore the same as the mean of the parents. Thus, the mean of the whole population remains the same.

Mean Centric crossover operators create offspring with a subset of parents. Their position in solutions space is symmetrically around the subsets mean, and thus the mean remains the same and so does the mean of the whole population.

Those features are examples of crossover operator properties, that can be identified by analysing the method and by visualizing the production scheme. From those features, we could conclude whether an operator converges fast because it is adaptive, or if it performs better in the end because it utilizes more information about the problems features like the rotation.

2.3.2 Related Work on Crossover Operators

The modular structure of an EA implies that the components can simply be exchanged. Accordingly, there exists a wide number of operators. In the next section, the following crossover operator are described:

- Uniform Crossover
- Simulated Binary Crossover
- Rotated Simulated Binary Crossover
- Laplace Crossover
- Simplex Crossover

Uniform Crossover

The Uniform Crossover (UX) operator described by Syswerda et Al. [17] usually uses two parents, though multiple parents are possible as well. For each gene, one parent is randomly selected. The offspring copies the gene from this parent. This procedure is described in Algorithm 2.3.

Algorithm 2.3: Uniform Crossover.

```
input
    X,   * Parent Generation
    N,   * Number of individuals
    D    * Number of decision Variables
begin
  i = 0
  Y = {}
  foreach i in 1:N
    y_i = {} *new individual with no values in genes
    foreach j in 1:D
      u = GenerateRandomNumber()
      if(u <= 0.5)
          y_{i,j} = x_{i,j}
          y_{i+1,j} = x_{i+1,j}
      else
          y_{i,j} = x_{i+1,j}
          y_{i+1,j} = x_{i,j}
      end if
      j = j+1
    end loop
    Y = Y ∪ {y_i, y_{i+1}}
    i = i+2
```

```
    end loop
    return 𝒴
end
```

Simulated Binary Crossover

The SBX operator, introduced by Deb and Argwal et Al. [18], is one of the widest known and used crossover operator. It utilizes an equally distributed random number u to determine β, which, on the other hand, is used as a weight for the two parents. The calculation is described below in Algorithm 2.4.

Algorithm 2.4: Simulated Binary Crossover Operator

```
input
    𝒳,   * Parent Generation
    N,   * Number of individuals
    D    * Number of decision Variables
initialize
    i = 0
    𝒴 = {}
begin
    foreach i in 1:N
        j = 0
        foreach j in 1:D
            u = GenerateRandomNumber()
            if(u ≤ 0.5)
                β = (2u)^(1/(N+1))
            else
                β = (1/(2(1−u)))^(1/(N+1))
            end
            y_{i,j}   = 1/2[(1+β)x_{i,j} + (1−β)x_{i+1,j}]
            y_{i+1,j} = 1/2[(1−β)x_{i,j} + (1+β)x_{i+1,j}]
        end
        𝒴 = 𝒴 ⋃ {y_i, y_{i+1}}
    end
    return 𝒴
end
```

Rotation-Based Simulated Binary Crossover

The Rotation-Based Simulated Binary Crossover (RSBX) operator developed by Pan et Al. [14] is a workover of the SBX, which is known as not rotationally invariant. On rotated problems, a variable linkage is added so that the PO solutions are not distributed parallel to any decision variable. For the SBX, that produces offspring along the axis of the solution space, the optima are difficult to find. After each environmental selection, the solutions approximate the optima and thus, they find the rotation, but the SBX does not use this information and produces fewer offspring that improve and move nearer to the rotated solutions. The step of processing the rotation information is added for the RSBX. As described in the Algorithm 2.5, the

rotation needs to be calculated first by using the principle directions (eigenvectors of the covariance matrix) of the parent generation. The set of current individuals is then rotated so that it is parallel to the coordinate axis, and the SBX finds good solutions. In the end, this rotation is reversed.

Algorithm 2.5: Rotated Simulated Binary Crossover.

```
input
    X       * Parent Generation
begin
    m = Mean(X)          * Center of Parent Generation
    C = Covariance(X)
    V = Eigenvectors(C)

    X' = (X - m) · V
    Y = SBX(X')

    Y' = Y · V^-1 + m
    return Y'
end
```

Laplace Crossover

The Laplace Crossover (LX) operator, inspired by Deep and Thakur et Al. [19], uses approaches from Differential Evolution Crossover (DE) and SBX. Here, the two parents give a direction with their difference similar to the DE. The weight of this direction vector is calculated by a distribution function and a uniformly distributed random value u similar to the SBX, but here a Laplace distribution is used to calculate β. To make β follow the Laplace distribution, the distribution function needs to be inverted. The algorithm, proposed by Thakur et Al. [19] is described in Algorithm 2.6.

Algorithm 2.6: Laplace-Crossover

```
input
  X,     * Parent Generation
  N,     * Number of individuals
  D,     * Number of decision Variables
  b      * Distribution Parameter
initialize
  i = 0
  Y = {}
begin
  foreach i in 1:N
    j = 0
    foreach j in 1:D
      u = GenerateRandomNumber()
      if(u ≤ 0.5)
        β = b · ln(u)
      else
        β = -b · ln(u)
```

```
            end
            y_{i,j}   = x_{i,j} + |β(x_{i,j} − x_{i+1,j})|
            y_{i+1,j} = x_{i+1,j} + |β(x_{i,j} − x_{i+1,j})|
        end
        i = i+2
        𝒴 = 𝒴 ∪ {y_i, y_{i+1}}
    end
    return 𝒴
end
```

Simplex Crossover

The Simplex Crossover (SPX) operator introduced by Tsutsui et Al. in [20] utilizes the concept of the well-known Blend Crossover (BLX) operator, introduced by Eshelman et Al. [21]. With different variations of this operator [22, 23], it is possible to cope to problems with different levels of variable linkage. Tsutsui et Al. used these ideas to develop the SPX operator, which is therefore effective on problems with and without rotation. For N decision variable, up to N+1 parents are selected. Those build a simplex and limit an area, from where the offspring are uniformly picked. The procedure is described in Algorithm 2.7.

Algorithm 2.7: Simplex-Crossover

```
input:
    𝒳,   * Parent Population
    ε,   * Scaling parameter to increase simplex area
    N,   * Number of individuals
    n    * Number of parents per simplex
initialize:
    𝒴 = {} *Offspring Population
    𝒮 = {} *Simplex edge point set
begin
    while i ≠ N,
        𝒳' = 𝒳_{i:i+n}              * Subset of 𝒳 with the next n individuals
        m = (1/n) Σ_{j=1}^{n} x'_j  * x'_j as individuals in 𝒳'
        foreach e in 1:n
            s_e = (1+ε)(m − x'_e)   * s_e as edge points of simplex formed by 𝒮
            𝒮 = 𝒮 ∪ {s_e}
        end
        𝒴 = 𝒴 ∪ {PickFromSimplex(𝒮,n))} * n solutions, uniformly
        i = i+n
    end
    return 𝒴
end
```

In this section, the concept of crossover operators, their properties and example, that are relevant for this work are presented. The very different properties of those operators imply, that there are better operators for different problems. To measure the quality of those operators, the algorithm needs to be evaluated with different methods and metric. Those are introduced in the next section.

2.4 Evaluation of Evolutionary Algorithms

In this section, the methods of evaluating evolutionary algorithms are presented. To show the quality of Meta-Heuristics, Test Suits or Benchmark Problems are selected, that cover a range of properties. The choice of Benchmark Problems defines but also limits the analysis potential. Often, the global PO solutions are known, so that the difference between those and the solutions the algorithm provided is calculable. Besides this derivation to the global optima, there are more properties that indicate the quality of a EA.

- Robustness: ratio of solutions that fulfil a predefined satisfaction criterion to the total number of solutions
- Feasibility: ratio of feasible solutions, and hence solutions violate the constraints to the total number of solutions
- Performance: the role of computational time is shrinking as the number of kernels used in parallel programs is not an important cost factor. Thus, the performance is rather measured in function evaluations, hence the number of times a solution's objective values are calculated, before termination criterion was met
- Convergence Speed: rating of the difference each generation exhibits to its prior generation
- Diversity: rating of the spread of the algorithms solutions, to give a measurement on how well the possible optimal area is covered

There are different metrics to measure those properties. To get a significant result, a frequent repetition is necessary. It is recommended to use at least 13 runs per benchmark and algorithm.

In the following subsections, different Benchmark Problems and different metrics are presented.

2.4.1 Benchmark Problems

Benchmark problems for the evaluation of the quality of multi-objective evolutionary algorithms are mathematical problems developed to simulate different properties and structures to create a reproducible and comparable measurement. For those problems, the Pareto fronts are usually known, so that the difference between the algorithms solutions and the Pareto optimal solutions can be calculated. Examples for Benchmark Problems are DTLZ [24], WFG [25] and ZDT [26].

Rotated Test Suite

To evaluate a specific property, it is sometimes beneficial to modify another well-known benchmark problem set. This was for example used in [14] to measure the quality improvement of their operator. The RSBX crossover operator was designed to improve the quality of MOEAs on rotated problems. Thus, the test suit RM introduced by Qingfu Zhang et Al. [4] was used. The first four problems are

modifications of ZDT1, ZDT2, ZDT6 and DTLZ2. A variable linkage was added to the objective functions. Those cause the Pareto optimal solutions to be no longer distributed parallel to any decision variable.

Table 2.1: Listing of desirable multi-objective test problem recommendations and possible test problem features [3]

Recommendation (R) or Feature (F)	Comment
R1: No extremal parameter	Prevents exploitation by truncation-based correction operators
R2: No medial parameters	Prevents exploitation by intermediate recombination
R3: Scalable number of parameters	Increases flexibility, demands scalability
R4: Scalable number of objectives	Increases flexibility, demands scalability
R5: Dissimilar parameter domains	Encourage EAs to scale mutation strengths appropriately
R6: Dissimilar tradeoff ranges	Encourages normalization of objective values
R7: Pareto optima known	Facilitates the use of measures, analysis of results, in addition to other benefits
F1: Pareto optimal geometry	Convex, linear, concave, mixed, degenerate, disconnected, or some combination
F2: Parameter dependencies	Objectives can be separable or non-separable
F3: Bias	Substantially more solutions exist in some regions of fitness space than they do in others
F4: Many-to-one mappings	Pareto one-to-one/many-to-one, flat regions, isolated optima
F5: Modality	Uni-modal, or multi-modal (possibly deceptive multi-modality)

The selection of problems is crucial to give an overview of the quality and advantages and disadvantages of an algorithm. For this, Huband et Al. [3] presented recommendations and features of benchmarks in Table 2.1 and also gave an analysis of different benchmark problems and which recommendations they fulfil. The results of this analysis on ZDT, DTLZ and WFG can be found in Table 2.2. The results for RM are transferred from ZTD and DTLZ and confirmed with the statements from Qingfu Zhang et Al. [4]. Huband et Al. analysed different Test Suits in detail, so for more information refer to [3]

Table 2.2: Analysis of ZDT, DTLZ and WFG from [3], added RM1-4 with information from [4]

name	objectives	R3: # Parameteres	F2: Seperability	F5: Modality	R1: No Extremal	R2: No Medial	R5: Diss. Domains	R6: Diss. Ranges	R7: Optima Known	F1: Geometry	F3: Bias	F4a: Pareto Many-To-One	F4b: Flat Regions
ZDT1	f_1	1	S	U	×	✓	×	×	✓	convex	-	-	-
	f_2	✓	S	U									
ZDT2	f_1	1	S	U	×	✓	×	×	✓	concave	-	-	-
	f_2	✓	S	U									
ZDT3	f_1	1	S	U	×	✓	×	✓	✓	disconnected*	-	-	-
	f_2	✓	S	M									
ZDT4	f_1	1	S	U	✓	×	×*	×	✓	convex	-	-	-
	f_2	✓	S	M									
ZDT6	f_1	1	S	M	×	✓	×	✓	✓	concave	+	+	-
	f_2	✓	S	M									
DTLZ1	$f_{1:M}$	✓*	S*	M	✓	×	×	×	✓	linear	-	+	-
DTLZ2*	$f_{1:M}$	✓*	S*	U	✓	×	×	×	✓	concave	-	+	-
DTLZ3	$f_{1:M}$	✓*	S*	M	✓	×	×	×	✓	concave	-	+	-
DTLZ4	$f_{1:M}$	✓*	S*	U	✓	×	×	×	✓	concave	+	+	-
DTLZ5*	$f_{1:M}$	✓*	?	U	?	?	×	?	×	?	-	+	-
DTLZ6*	$f_{1:M}$	✓*	?	U	?	?	×	?	×	?	+	+	-
DTLZ7*	$f_{1:M-1}$	1	NA	U	×	✓	×	×*	✓	disconnected*	-	-	-
	f_M	✓	S	M									
RM1	f_1	1	S	U	×	✓	×	×	✓	convex, rotated	-	-	-
	f_2	✓	S	U									
RM2	f_1	1	S	U	×	✓	×	×	✓	concave, rotated	-	-	-
	f_2	✓	S	U									
RM3	f_1	1	S	M	×	✓	×	✓	✓	concave, rotated	+	+	-
	f_2	✓	S	M									
RM4	$f_{1:M}$	✓*	S*	U	✓	×	×	×	✓	concave, rotated	-	+	-
WFG1	$f_{1:M}$	✓	S	U	✓	✓	✓	✓	✓	convex, mixed	+	+	?
WFG2	$f_{1:M-1}$	✓	NS	U	✓	✓	✓	✓	✓	convex, disconnected	-	+	?
	f_M	✓	NS	M									
WFG3	$f_{1:M}$	✓	NS	U	✓	✓	✓	✓	✓	linear, degenerate	-	+	?
WFG4	$f_{1:M}$	✓	S	M	✓	✓	✓	✓	✓	concave	-	+	?
WFG5	$f_{1:M}$	✓	S	D	✓	✓	✓	✓	✓	concave	-	+	?
WFG6	$f_{1:M}$	✓	NS	U	✓	✓	✓	✓	✓	concave	-	+	?
WFG7	$f_{1:M}$	✓	S	U	✓	✓	✓	✓	✓	concave	+	+	?
WFG8	$f_{1:M}$	✓	NS	U	✓	✓	✓	✓	✓	concave	+	+	?
WFG9	$f_{1:M}$	✓	NS	M,D	✓	✓	✓	✓	✓	concave	+	+	?

2.4.2 Evaluation Metrics

Hyper Volume

Hyper Volume (HV) is a metric to measure the convergence and the diversity at once. It was introduced under the name *S-Metric* by Zitzler et Al. [27]. Each individual can form polytopes with edges along the axis of the objective space with a reference point. The space covered by those intersecting polytopes is the HV. Those polytopes are shown for a two-dimensional objective space in Figure 2.5.

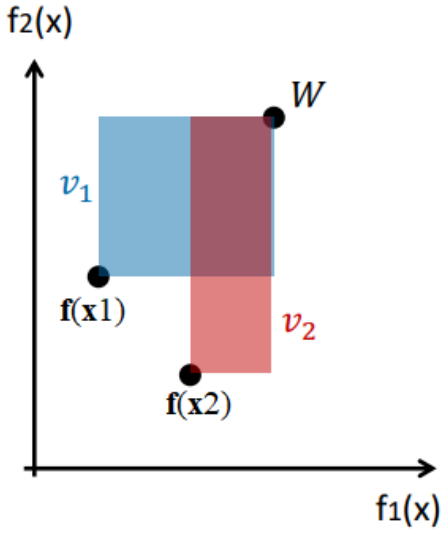


Figure 2.5: Example of a Hypervolume in two-dimensional objective space

Definition 2.5: Hypervolume

Each of the n individuals $\boldsymbol{x}_i \in \mathcal{X}$ forms a polytope P_i along the axis of the objective space with a reference point W.
The Hypervolume is then:

$$HV(\mathcal{X}) = \bigcup_{i=1}^{n} P_i$$

The HV can be calculated during the optimization process, to measure improvements over generations, or to make assumptions about the coverage of solutions. The larger the HV, the more space is covered, and the more diverse and the nearer to the optimum is the current solution set. To evaluate the final result, the HV of the last generation can be compared to the HV of the Pareto front.

Inverted Generational Distance

The idea of measuring the quality by calculating the distance from the calculated Pareto front to the real Pareto front, was firstly proposed by Van Veldhuizen [28] under the name *Generational Distance*. This metric calculates the average distance between the individuals in the current generation and their nearest PO individuals in objectives space. The IGD, as the name says, inverts this method and describes the average distance of the PO solutions and their nearest individual of the current generation in objective space. This idea was introduced by Czyżżak et Al. [29] under a different name, and later used again under the name *Inverted Generational Distance* by Coello Coello et Al. [30].

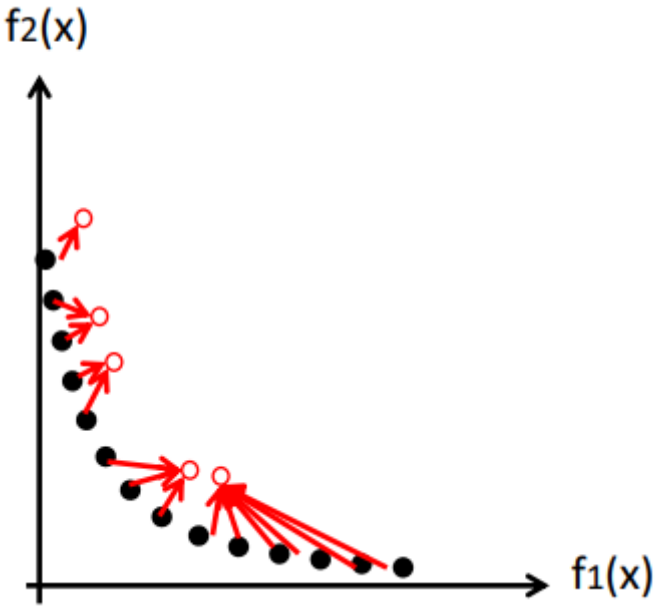


Figure 2.6: Example of IGD in two-dimensional objective space. Red circle: current generation, Black circles: PO Solutions. Red arrows show, between which PO solutions and which current solutions, the difference is calculated.

Definition 2.6: Inverted Generational Distance

With $\mathcal{P}$ as the set of PO solutions and $\mathcal{X}$ as the current generation

$$IGD(\mathcal{P},\mathcal{X}) = \frac{(\sum_{i=1}^{|\mathcal{P}|} d_i^q)^{\frac{1}{q}}}{|\mathcal{P}|}$$

$$\text{with } d_i = \min_{k\in|\mathcal{X}|} \sqrt{\sum_{j=1}^{m} (f_j(\boldsymbol{p}_i) - f_j(\boldsymbol{x}_k))^2}$$

so that d_i measures the euclidean distance in objective space between $\boldsymbol{p}_i \in \mathcal{P}$ and the nearest individual of $\mathcal{X}$.

Therefore, Generations with a small IGD value have a good diversity. An example of IGD for two-dimensional objective function is shown in Figure 2.6.

Those metrics presume that the optimal solutions are given. Thus, benchmark problems with known optima are used on algorithms and the metrics are measured to compare algorithms against each other. In the case, that a quality measurement in runtime on unknown problems are desired, R2 is a possible metric.

R2-Metric

The R-Metrics Family, introduced by Hansen et Al. [31], serves as an alternative to HV. It can as well be used to classify the diversity and the convergence of EAs. As HV has bad performance for an increasing number of objectives [32], it would not be recommended for online evaluations. Thus, the R-Metric is a possible alternative. The R-Metric Family uses parametrized utility functions $u_{\boldsymbol{\Lambda}}$, that could be the known

preference of the decision maker, or, as used here, a weighted sum function. Brockhoff et Al. [32] specified the R2 metric with the weighted Tchebycheff function for u as follows:

Definition 2.7: R2-Metric

The R2 Indicator I_{R2} of a solution set $\mathcal{X}$ with a utopian reference point z^* and a parameter set $\mathbf{\Lambda}$ is defined as:

$$I_{R2}(\mathcal{X}, \mathbf{\Lambda}, z^*) = \frac{1}{|\mathbf{\Lambda}|} \sum_{\lambda \in \mathbf{\Lambda}} \min_{x \in \mathcal{X}} (\max_{k \in \{1,\dots,l\}} (\lambda_k | z_k^* - f(x_k)|)$$

The Definition 2.7 says, that for each solution there is an objective, for which the weighted distance between the solution and the utopian point is maximized. From those maximized weighted distances for each solution, the smallest is selected. This is repeated for all weight vectors of $\mathbf{\Lambda}$ and summed up.

Zitzler et Al. [33] recommended to use numerous uniformly distributed normalized weight vectors. Brockhoff et Al. [32] concludes, furthermore, that the number of generated weights influence the accuracy and usability of the metric and states the number of weight vectors should be at least about 10 times the number of solutions.

In this section, methods to evaluate EAs are presented. These methods are used, to evaluate the quality of the algorithm after the run. They can also be used to measure the quality of generations during the execution of the algorithm, hence online. This information is utilized in Hyper-Heuristics. This concept is introduced in the following section.

2.5 Hyper-Heuristics

As already mentioned, Meta-Heuristics use a general approach to solve optimization problems independent and without knowledge about the problem. But the restriction of the use of these algorithms remains, as they perform differently well on different problem classes or families. To compensate weaknesses of Meta-Heuristics, the concept of Hyper-Heuristics arose. Hyper-Heuristics solve a superior problem, where the search space consists of Heuristics to solve a problem instead of solutions to a problem. Thus, they choose the Heuristic, that would likely solve the problem in the best way.

The No Free Lunch Theorem of Wolpert and Macready [9] for optimization problems states that all algorithms perform on average equally well over all existing problems. Meta-Heuristics with their general approach did not disprove the theorem so far, but Ross et Al. found out that it was not true for subsets of problems in cite [34]. Thus, specific Meta-Heuristics perform better on specific classes of problems than other Meta-Heuristic. Accordingly, there is probably the best choice of algorithm for each problem or each problem class. This choice is not trivial to make because problems are difficult to classify or even impossible when the environment of the search space or the geometry of the $\mathcal{PF}$ is unknown, as Ross et Al. [34] confirms. Therefore, Hyper-Heuristics as computational decision makers for the choice of the algorithms could be a helpful addition to Meta-Heuristics like MOEAs.

A well-known example of a problem that could benefit from the use of Hyper-Heuristics is job scheduling [34]. Although, it can be defined as one problem class because the solution spaces are alike, the properties and the complexity of these problems differ extremely and depend on the decision makers preferences. They can even be time dependent and change their properties while solving. In this case, it is nearly impossible to know which Meta-Heuristic would work best or even good enough. A Hyper-Heuristics could improve the situation by selecting the algorithm during the solving process, hence online [35]. If there are multiple similar problems, an offline learning Hyper-Heuristic [36], that build a knowledge base from a training with those problems, could be used.

Hyper-Heuristics in this work are selectors of algorithms or components. A selector needs a selection method and also a possibility to collect information on which the selection is founded. Thus, there are always two components that build the base of a Hyper-Heuristic: a Selection Heuristic and a Scoring Function, that summarize the gathered information. In the following subsections, different implementations of Hyper-Heuristics are presented.

2.5.1 Hyper-Heuristic on MOEAs: HHcMOEAD

The first Hyper-Heuristic algorithm presented here is the HHcMOEAD developed by Fritsche et Al. [35]. The Hyper-Heuristic swaps between seven whole MOEA. The probability of selecting a MOEA is updated in each generation, thus is this Hyper-Heuristic a probabilistic online learner. After every new generation, the quality is measured with the R2 Metric. Its improvement rate is included in the probability calculation.

The seven MOEAs include for example NSGA-II, which performs well on problems with $M \leq 3$ and NSGA-III that performs well on problems with $M > 3$. Thus, changing the number of objectives emphasizes the quality differences and should lead to a recognizable impact of the Hyper-Heuristic.

To show those differences, Fritsche et Al. executed HHcMOEAD on DTLZ1-2 and WFG1-4 with different number of objectives and their mirrored versions. The results are very promising as the HHcMOEAD delivers mostly the best results. There are still a few cases, where it was not the best. This could be caused by the learning process, where not the best algorithm was selected, and thus the convergence speed was lowered. Additionally, it is interesting, that the combination of the best performing algorithm with bad performing algorithms, is better than the best algorithm alone. This could mean, that the algorithm with the best result does not always perform best and at some points of the solving process another algorithm brings more and better progress.

The conclusions of this work for the design of a Hyper-Heuristic would be, that some algorithms work better in the beginning than in the end, and thus a fast learning is essential to adapt fast to change of preference. Also, reevaluations of longer not used algorithms are needed because of that change.

2.5.2 Hyper-Heuristic for Mutation Operator: Automatically Designed Mutation

The next Hyper-Heuristic is an offline learning mutation operator designer presented by Hong et Al. [36]. The base is a tailored mutation, that is altered during multiple training runs. Two different methods are implemented and compared. The first mutation operator is trained to solve a specific problem and the other one is trained to solve a problem family. The main difference is the training data set. In this example, the Hyper-Heuristic is not a selector but a modifier. To modify the parameters of the tailored mutation operator, Genetic Programming is used.

Both operators and a manually designed tailored mutation operator are tested on different benchmark problems. HV and IGD are measured and compared. As expected, the operator trained for a specific problem performs best on this problem, but it has a deficit on the other problems. The operator trained for the problem family performs well on the problems that belong to that family, although it is not necessarily the best. This emphasizes the major difficulty with offline learners. The selection of the training data. The specific operator is highly overfitted to these specific problems.

This implementation was done for single objective problems, but the idea can be transferred to MOPs. For MOPs it is harder to design a mutation operator manually because they combine different features. Here is a chance for the offline training because you can train with data that combines features and problems that have a specific feature. Nevertheless, finding the best trainings data and training procedure is hard and might be very different from case to case, so that finding a general mutation operator for MOPs could be a great obstacle.

2.5.3 Hyper-Heuristic for Crossover Operators: UNDX and UX

The next Hyper-Heuristic is again a for single objectives and decides whether the crossover operator should be an Unimodal Normal Distribution Crossover (UNDX) or a UX. The algorithm is developed by Ono et Al. in [37]. Ono et Al. included the Hyper-Heuristic in the Minimal Generation Gap (MGG) model, that check for every child, whether it is better than its parents. If it is better, it replaces one of the parents. This is then called a successful child solution. The UNDX crossover operator samples children with unimodal normal distribution on the line between its parents. Thus, it keeps the information about the correlation. Unfortunately, the exploratory behaviour is bad so that it is very difficult to find the optimum when it is near the boundaries of the search space. The UX works exactly vice versa, has good exploratory behaviour but looses the information about correlation.

The Hyper-Heuristic used in this work is probabilistic and updates the selection probability with the ratio of successful child solutions. The selection of the operator is then parent wise and not generational wise, and thus the parents may use different crossover operators in one generation. The probabilities for selecting one of the operators is summed up to one. One operator never has a zero or 100% chance to be picked. By that way, it is ensured, that both operators can be picked and updated in each generation. Thus, the algorithm adapts every time there is no further

movement possible for UNDX the UX operator would use its better exploration and have successful children until exploitation is more needed.

The focus in this work lies on the robustness. Because of the bad exploratory behaviour, there are problems, where the UNDX did not find any sufficient solutions. The test suit contains mainly such problems, and it is investigated whether the exchange to the UX operator help to solve those problems. Furthermore, the same applies to UX so other problems are added to the test suit, where UX could not find satisfying results. Furthermore, among the test functions are functions that were not properly solved by any of those two operators.

The results of that test show an increase in robustness with the adaptive operator. Not only is the chance higher to solve a problem, that one of those operators could solve, but also the chance to solve a problem that neither of them solved alone. This underlines again the experience, that the quality of solutions a crossover operator produces does not only depend on the problem, but on the parent generation, on the progress that was already made and the phase of the current run.

2.5.4 Hyper-Heuristic for Crossover Operators: SBX and RSBX

The last algorithm presented in this work is the Adaptive Simulated Binary Crossover developed by Pan et Al. [14]. This algorithm combines the SBX (see Algorithm 2.4) and the RSBX (see Algorithm 2.5). In this work, the RSBX was developed because the SBX is not rotational invariant and thus delivers bad results on a rotated problem. The RSBX uses the information about the rotation of the problem to reverse the rotation, use the SBX and rotate it back. The rotation of the whole population two times for every generation slows the algorithm extremely, especially when the problem is not rotated and these operations would not be necessary. Therefore, a Hyper-Heuristic is introduced, that swaps between SBX and RSBX. This is again a probabilistic online learner that also uses a success rate as a scoring function. Success in this case is the survival of an individual produced by one of the operators. The scores are converted to probabilities by using a sigmoid function that also depends on the complexity.

To evaluate the algorithm HV and IGD are measured on different benchmarks. The rotated problem designed by Qingfu et Al. in [4] are also used here. The experiments showed, that the combination of those two operators solved the difficulties described before. In this work, it is again very well visible, that there is not the only best algorithm for a problem. Whether SBX or RSBX is better depends on here again, on the problem, the parent generation and the phase of the problem.

Summary

There are already numerous concepts and implementations of Hyper-Heuristics, although it is not called by that name. It was predictable that the idea of generalization would not stop with Meta-Heuristics, and more general optimization algorithms would be investigated. All of these works about Hyper-Heuristic have the design of the concept in common. A Hyper-Heuristic solves an optimization problem, which is defined as finding the best algorithm or component of an algorithm to solve the

problem. To accomplish this, the Hyper-Heuristic can utilize online learning, offline learning or another optimization algorithm like the use of Genetic Programming.

Hyper-Heuristic Online Learners are mainly built up with two components. The selection heuristic can be probabilistic or a multi-armed bandit, for example. The scoring function can be anything that measures the quality. This measurement is then used to update the selection heuristic.

In this chapter, the basics of this work are clarified. EAs and MOPs are introduced, current works on crossover operators and evaluation metrics used in this paper are announced. The concept of Hyper-heuristics and different forms of implementations were described. In the next chapter, our characterization of the crossover operators and the application of this information as 4 different variants of Hyper-Heuristics are presented.

3. Proposed Methods

In this chapter, we propose our methods, to answer the question, whether Hyper-Heuristics are suitable as selectors of crossover operators and if these improve the quality of the solutions produced by MOEAs. To answer this, we implement different variants of Hyper-Heuristics Online Learner and evaluate their quality and analyse their behaviour, to show possible improvements on MOEAs.

As described in the prior chapter, Hyper-Heuristic Online Learner consist of three crucial components:

- Selection Heuristic
- Reward Function
- Selection Pool

In the following, we explain the design of our different algorithm variants. At the beginning, we give an overview about the extension of MOEAs with Hyper-Heuristics. Afterwards, we explain two different kinds of selection heuristic, followed by four reward functions. In the end, we describe the crossover operator, that are part of the selection pool.

3.1 Hyper-Heuristic Online Learner

The general structure of a MOEA with Hyper-Heuristic for the selection of the crossover operator is illustrated in Figure 3.1.

In this figure, instead of a crossover operator that takes the parent generation and produces the next generation, the Hyper-Heuristic takes over this task. For each generation, the Hyper-Heuristic uses its selection heuristic, to select one or more crossover operators from its selection pool These operators produce the child generation. The child generation is saved for the reward function. The stored and the current generation are compared in the reward function, and a corresponding score is stored as information for the selection heuristic in the next generation. By

updating the score for the different crossover operators in the selection pool, we create an online learning effect and teach the Hyper-Heuristic, which operators work best on the current problem.

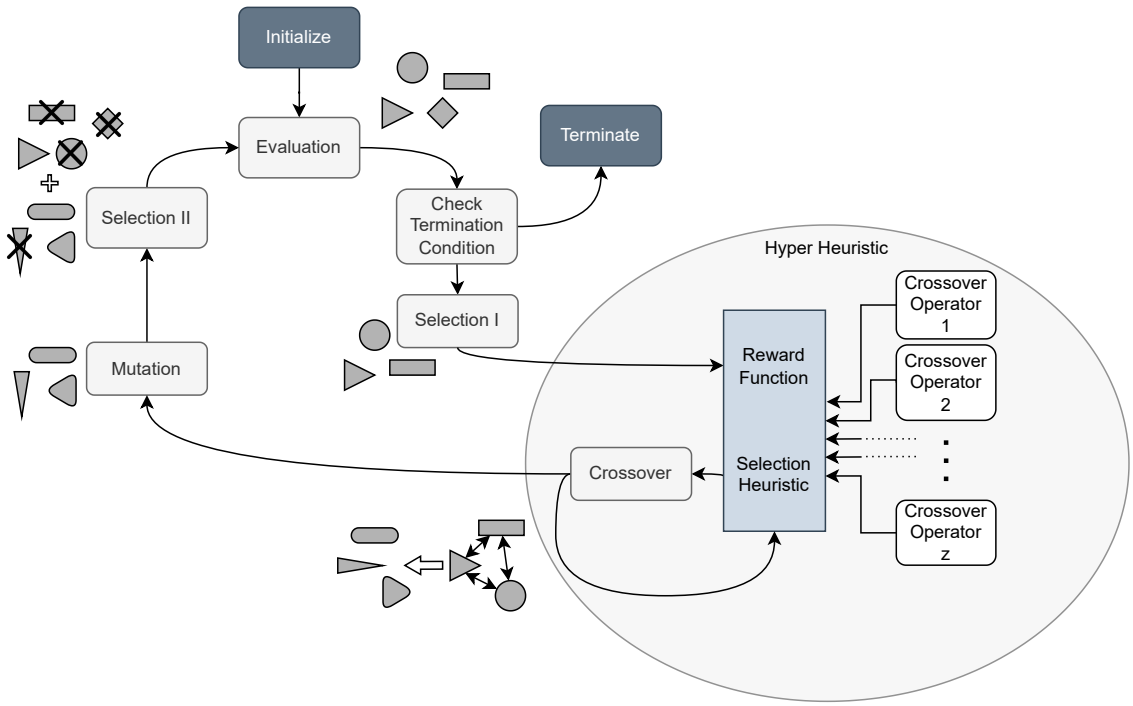


Figure 3.1: Visualization of the extended evolutionary algorithm. See original evolutionary algorithm in Figure 2.2 for comparison-

3.1.1 Selection Heuristics

We implement two different selection heuristics: The first variant utilizes the Roulette Wheel Selection and is used in hyper-heuristic selectors (HHSs). The second one distributes the current generation to all operators, depending on their score. The Hyper-Heuristics using this variant are called hyper-heuristic distributors (HHDs).

Algorithm 3.1: Hyper-Heuristic Selector.

```
initialize:
  O = {...}   * Set of Operators
  RR = 1, * Reward Rate
  MINP = 0, * Min. Probability
  EP = 3 * Explorations
input:
  X, * Parent Population
  Gen * Number of Generation
begin
  R = RewardFunction(X)
  Save(R)

  if(mod(Gen, RR) == 0) then
    UpdateScores(O, R, MINP)
  end

  if(Gen < EP × |O|)
    o = O(mod(Gen, |O|))
  else
    o = RouletteWheel(O)
  end

  Y = o.Cross(X)
  save(Y)

  return Y
end
```

Algorithm 3.2: Hyper-Heuristic Distributor.

```
initialize:
  O = {...}   * Set of Operators
  MINP = 0, * Min. Probability
input:
  X, *Parent Population*
begin
  R = RewardFunction(X)
  UpdateScores(O, R, MINP)
  foreach o in O do
    X_o = Distribute(X, o)
    Y_o = o.Cross(X_o)
    Y = Y ⋃ Y_o
  end
  return Y
end
```

The HHS, is described in Algorithm 3.1, and the HHD is described in Algorithm 3.2. Both firstly use the reward function to rate the current generation. This can be done by comparing it with a prior saved generation, but the current generation can also be rated alone. As only one operator is used in one generation of HHSs, the new reward values are saved for later. The HHSs needs more configuration in terms of the exploration phase as the HHDs. Those always use all operators, and thus can simply update the score in every generation. The HHS, on the other hand, needs a phase, where it can score every operator at least once because it can only update the rewards of one operator in each generation. Therefore, it would start with a random selection and this selected operator is either always or never again selected. The EP parameter gives the number of generations a crossover operator should have solved, before the scores are used. By default, this value is set to three. RR is the reward rate and is another parameter, that controls the exploration phase. This number states, how often the currently saved rewards should be updating the score. This is 1 by default, so the score is updated in every generation.

Both variants have the MINP parameter. This sets a minimum value, for the selection probability or the portion of the population. With this parameter set to

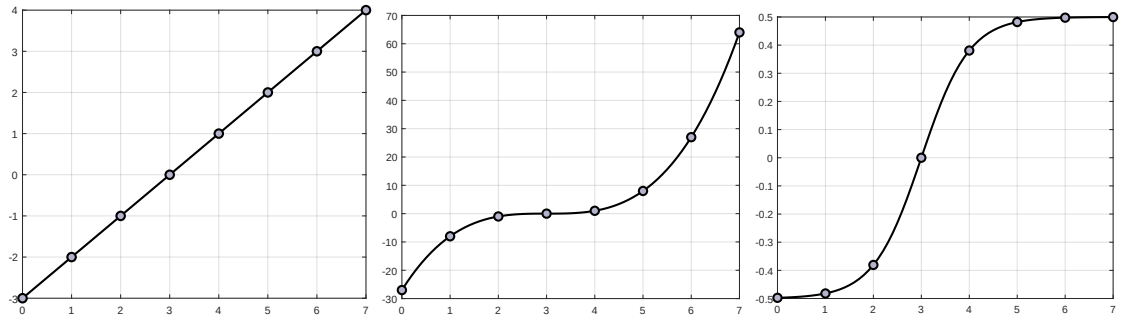


Figure 3.2: Linear, Cubic and Logistic Functions to map the rank to a Score value.

higher than zero, we force the HHDs always to use all operators. For the HHS it also can never completely exclude one operator, but it is still unlikely to be selected with a low selection probability. As the reward function can be anything, we do not know the value ranges, and thus, we decide to create a more general mapping and create a score. By this way, it is ensured, that the best operator will have a significant higher probability of being picked than the others, although the rewards do not differ that much. The difference needs to be emphasized as it could otherwise lead to only little differences in probability, which would mean, the selection is nearly random. The rewards are assigned to the operators and then ranked. The rank is then used in a scoring function. A few examples of scoring functions are shown in Figure 3.2.

The scoring function needs to have the following properties: around the middle of the ranked the scores should be near to zero, for the lowest ranks the score should be negative, and for the highest ranks the function should increase fast. The cubic Function 3.1 is the simplest that meets these criteria. With these criteria, it is secured, that the best operator will always have a higher probability of being picked.

$$s_{o_i}(x_{o_i}) = (x_{o_i} - x_{center})^3 \tag{3.1}$$

With x as a rank of the operator o_i, x_{center} as the middle rank, and s as the score of each operator.

The scoring update is adding the new calculated scores to the stored current score value for each operator. For the HHS, this means, the reward is calculated for the whole population for one operator and this operator is the only one updated in this generation. For this operator, the reward value is overwritten and saved, and scores are updated with all current rewards, if the RR allows it. For the HHD also the reward is calculated for the whole population, but regarding the operators that generated the individuals. Thus, all operators are updated in one generation.

After this update, the distribution of the scores is calculated as follows:

$$p_{o_i} = \frac{s_{o_i}}{\sum_i s_{o_i}} \tag{3.2}$$

This distribution is then for the HHS used as a selection probability and as a distribution for the HHD. This function needs to be modified, if a minimum probability is set.

For the last step in Algorithm 3.1 the HHS only needs to select a crossover operator according to the probability, and this operator will then generate the next generation. The HHD needs to distribute the population according to the distribution values. Thus, each operator generates one part of the population and those portions are in the end merged to one new generation of solutions.

Each of the two approaches of Hyper-Heuristic Online Learners are implemented with three different reward function and a fourth version with a uniform reward for comparison reasons. This means, the HHS always chooses randomly and the HHD always uses an equal distribution. The other three rewarding functions are presented in the following subsection.

3.1.2 Reward Functions

In this subsection, three reward functions are described. Those reward functions implement different approaches in measuring the quality of a solution set. As a reference set, they all use a previous generation. For those functions that use the objective value, it is important, that the generation is not stored after the crossover operators, but after the mutation, so that the objectives can be calculated and stored with the generation.

NC - Reward

The first reward function is the NC reward, that uses NDS as described in Algorithm 2.1 and CD as described in Definition 2.4. For both evaluations, the objective space is used. The function is described in Algorithm 3.3. The current parent generation and the former parent generation serve as input parameters.

Algorithm 3.3: NC.

input :
  $\mathcal{X}_{old}$,
  $\mathcal{X}_{new}$
begin
  $\mathcal{A} = \text{NDS}(\mathcal{X}_{old} \cup \mathcal{X}_{new})$
  $\mathcal{A}' = \text{FirstFront}(\mathcal{A})$
  $nd_{old} = \text{Count}(\mathcal{A}' \cap \mathcal{X}_{old})$
  $nd_{new} = \text{Count}(\mathcal{A}' \cap \mathcal{X}_{new})$

  $cd_{old} = \text{Median}(\text{CD}(\mathcal{X}_{old}))$
  $cd_{new} = \text{Median}(\text{CD}(\mathcal{X}_{new}))$

  $\text{bonus} = 1/4 \cdot cd_{new}/(cd_{new}+cd_{old}) + 3/4 \cdot nd_{new}/|\mathcal{X}_{new}|$
  $\text{malus} = 1/4 \cdot cd_{old}/(cd_{new}+cd_{old}) + 3/4 \cdot nd_{old}/|\mathcal{X}_{old}|$

  return bonus − malus
end

In the beginning, the old generation $\mathcal{X}_{old}$ and the newest generation $\mathcal{X}_{new}$ are compared in terms of domination. Thus, they are merged into one set of solutions and sorted according to the NDS. Then, the first front is examined: The individuals,

that were originally in the old generation are counted, as well as the individuals, that were originally in the new generation. Moreover, we calculate the median of the CDs of the old and the new generation. In the end, there is a score for the new generation, which is the bonus, and a score for the old generation, which is the malus, that is subtracted from the bonus. The remaining bonus is the reward. If the previous generation is better in those terms than the current generation, the reward could also be negative.

Survival Reward

This reward function uses a different approach. It needs the current parent generation and the previous child generation as an input. The reward counts the individuals that were produced in one iteration and survived a whole selection process. The procedure is described in Algorithm 3.4

Algorithm 3.4: Survival Reward.

```
input :
  X_i , * current parent generation
  Y_{i-1} * previous child generation
begin
    Y'_{i-1} = FromSameOperator(Y_{i-1})
    survived = |X_i|
    born = |Y'_{i-1}|
    all = |Y_{i-1}|
    return survived/all + survived/born
end
```

At the beginning, the previous child generation is filtered, so that only those individuals that belong to the current operator are regarded. Afterwards, the individuals that survived the selection, and are now part of $\mathcal{X}$, the previously born individuals from that operators, and all previously born individuals are counted. Two different survival rates are set in ratio. The first one, gives a statement about the number of individuals that survived compared to all individuals in one generation. This ratio alone would leave out the information about, how many individuals were produced. In the HHD for example, there could be an operator that was only allowed to produce two offspring, but they both survived the selection. In the same generation there could be an operator that produced 40 offspring but only 10 did survive. Both ratios should be considered, to assume about the quality of the operators.

R2-Reward

The last reward function, is the R2-Reward. We use it with the Definition 2.7 for the current parent generation and the last parent generation in objective space. Those values are summarized to an improvement rate, which is used as the reward. The procedure is described in Algorithm 3.5

Algorithm 3.5: R2-Reward.

```
input :  X_old ,  X_new
```

begin
utopian = $\min(\mathcal{X}_{old}, \mathcal{X}_{old})$

$R2_{old}$ = R2($\mathcal{X}_{old}$, utopian)
$R2_{new}$ = R2($\mathcal{X}_{new}$, utopian)

return $(R2_{old} - R2_{new})/R2_{new}$;
end

We create the Hyper-Heuristics variants by combining the two selection heuristics with the four reward functions. Thus, we receive eight different Hyper-Heuristics and achieve the second subgoal of this thesis.

- NCRXS: Non-Dominated Sorting + Crowding Distance Reward Selector
- NCRXD: Non-Dominated Sorting + Crowding Distance Reward Distributor
- SRXS: Survival Reward Selector
- SRXD: Survival Reward Distributor
- R2XS: R2-Reward Selector
- R2XD: R2-Reward Distributor
- URXS: Uniform Reward Selector
- URXD: Uniform Reward Distributor

Those Hyper-Heuristics are used to decide about the usage of the crossover operators for the next generation. Thus, they have a selection pool with crossover operators, from which they choose. This set of operators is introduced in the next section.

3.2 Crossover Operators

The selection pool is of great importance for the quality of the decision. In this work, the Hyper-Heuristic is used to decide on crossover operators, and thus the selection pool consists of crossover operators. The more diverse the set of operators, the more special features of a problem might be covered by one of them. Therefore, the preselection of those crossover operators can already make a big difference about the usability of this concept. In Section 2.3.2, different crossover operators and algorithms, from which some operators are derived, were introduced. The following operators are selected and analysed for this work:

- Uniform Crossover
- Simulated Binary Crossover
- Rotated Simulated Binary Crossover
- Laplace Crossover
- Differential Evolution Crossover
- Linear Combination Crossover

- Covariance Matrix Adaption Crossover

Each operators' production scheme is visualized to demonstrate their properties. We created those schemes by letting them produce 500 offspring from three parents.

Uniform Crossover

The first operator we use in this work is the UX operator described in Algorithm 2.3.

Since the offspring produced by this operator are always distributed around its parents, this operator is parent-centric. It does not adapt its scheme to the parent's properties, and thus it also is not self-adaptive.

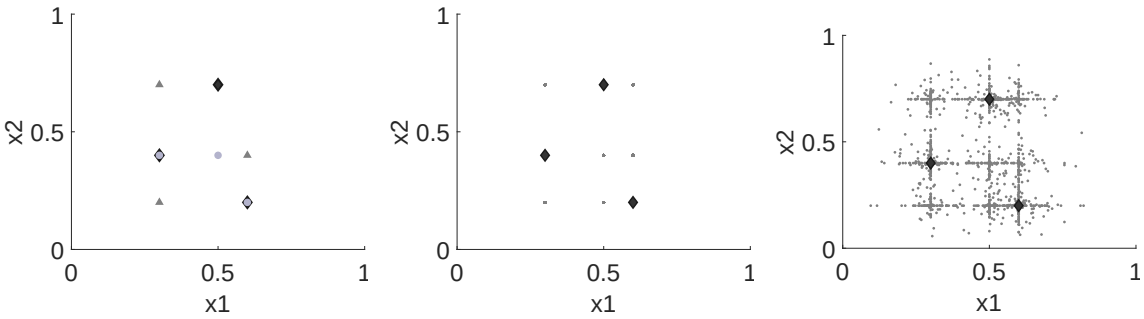


Figure 3.3: Production Scheme of UX operator, dark diamond Markers: Parent Individuals, Grey Markers: Offspring. Left: two Generations, Center: 500 Offspring to show pattern, Right: 500 Offspring with Polynomial Mutation

In Figure 3.3 it is visualized what the production scheme of offspring with an UX operator looks like. There are always 2^n, where n is the number of genes, variants of offspring per two parents, including an equal individual to one of the parents..

Simulated Binary Crossover

The next crossover operator is the SBX operator, as the most classical and widely known approach. The procedure is described in Algorithm 2.4.

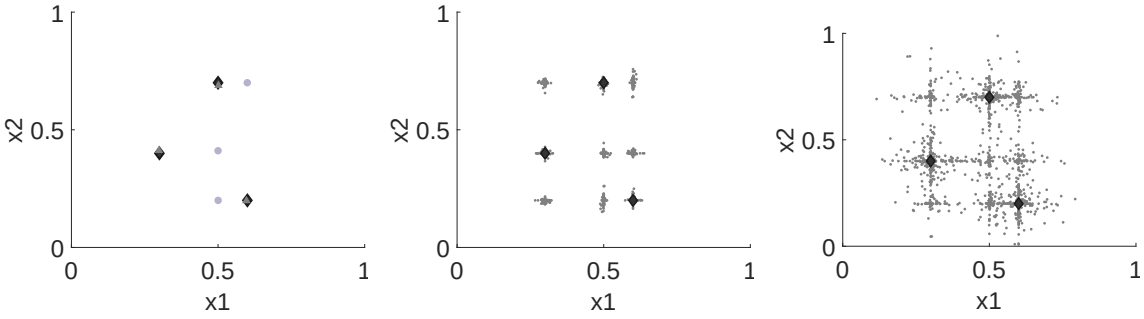


Figure 3.4: Production Scheme of SBX operator, dark diamond Markers: Parent Individuals, Grey Markers: Offspring. Left: two Generations, Center: 500 Offspring to show pattern, Right: 500 Offspring with Polynomial Mutation

The offspring produced by the SBX operator are visualized in Figure 3.4. The scheme is similar to the UX, but with more variation. Due to the distribution

described in the formula above of β, the produced offspring have a higher probability to be between the parents the closer the parents are to each other. Thus, the crossover is self-adaptive and also parent-centric. In Figure 3.4 it is also visible, that the offspring are produced parallel to the axis. Some problems feature is the rotation of the optimal solutions. The SBX has problems finding those solutions because of this not rotational invariant pattern.

Rotation based Simulated Binary Crossover

A further development of the SBX operator is the RSBX, described in Algorithm 2.5. This operator is very similar to the SBX operator, but claims to be rotational invariants. By comparing Figure 3.4 with Figure 3.5 it becomes clear, that the SBX creates solutions parallel to the axis of the search space and the RSBX utilizes the rotation that might arouse due to the optimization and creates solution along the rotated axis.

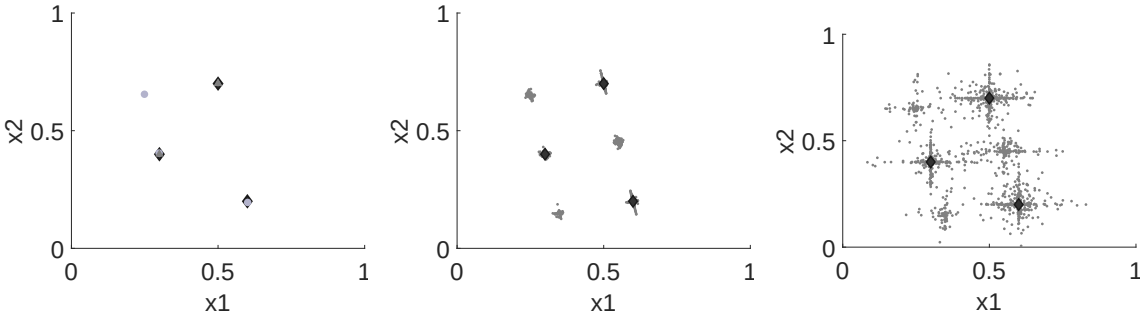


Figure 3.5: Production Scheme of RSBX operator, dark diamond Markers: Parent Individuals, Grey Markers: Offspring. Left: two Generations, Center: 500 Offspring to show pattern, Right: 500 Offspring with Polynomial Mutation

Despite the rotational invariance, the RSBX operators shares its properties with the SBX operator. Thus, it is parent-centric and self-adaptive. It is already clear, that the RSBX performs better than the SBX on rotated problem.

Differential Evolution Crossover

The DE operator is abbreviated from a part of Differential Evolution algorithm described in Section 2.2.3. We use the Equation 2.1 with a fixed F of 0.5. The procedure of creating offspring with DE is described in Algorithm 3.6.

Algorithm 3.6: Differential Evolution Crossover

```
input:
  𝒳,
  D,
  N
initialize:
  i = 0,
  𝒴 = {}
begin
  while(i ≠ N)
    foreach j in 1:D
```

$y_{i,j} \leftarrow x_{i,j} + 0.5 \cdot (x_{i+1,j} - x_{i+2,j})$
end
i $\leftarrow$ i+3
$\mathcal{Y} = \mathcal{Y} \bigcup \{\boldsymbol{y}_i\}$
end
return $\mathcal{Y}$
end

With this algorithm using three parents, offspring are generated and visualized in Figure 3.6.

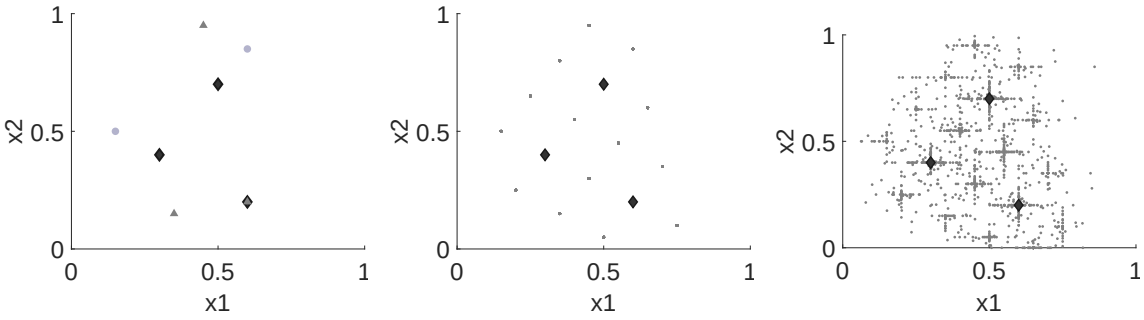


Figure 3.6: Production Scheme of DE operator, dark diamond Markers: Parent Individuals, Grey Markers: Offspring. Left: two Generations, Center: 500 Offspring to show pattern, Right: 500 Offspring with Polynomial Mutation

Here, it is visible, that the number of possible offspring is limited and with no variance in it. Thus, there is no reaction to the positions of the parents and there is no self-adaptive behaviour. We always want to produce the same number of offspring as the number of parents. Thus, for each parent in $\mathcal{X}$, two other parents are randomly selected, and only one offspring is generated. This is repeated for each parent in $\mathcal{X}$. The offspring markers in Figure 3.6 show all offspring, that could be produced with those three parents. They are always surrounding one of the parents, and thus the DE is a parent-centric crossover.

Covariance Matrix Adaption Crossover

The Covariance Matrix Adaption Crossover (CMAX) operator is based on Covariance Matrix Adaption [10], that is described in Algorithm 2.2. This algorithm describes a possibility to use the set of solution in its entirety to produce a complete new set of offspring. This idea is used as a crossover operator. Instead of updating the matrix in each step, it is calculated with each generation. New solutions are sampled with a multivariate normal distribution using the covariance matrix and the centre of the current generation. After the mutation and the two selection steps, a subset of the solutions forms the new generation and will approximate an optimum as an entirety. The crossover procedure is also described in Algorithm 3.7.

Algorithm 3.7: Covariance Matrix Adaption Crossover.

input :
$\mathcal{X}$,
σ

```
begin
    m = Mean(𝒳)
    C = Covariance(𝒳)
    𝒴 = m + σ · NormalDistribution(0, C)
    return 𝒴
end
```

A step size σ is given, that defines a spread factor. Increasing the value increases the distance between the offspring and the parent's centre. By tuning its value, an exploration phase to find a promising search area or an exploitation phase, where a smaller area is focused and searched, can be defined. An example of a generation produced by CMAX with different σ is visualized in Figure 3.7. In this work CMAX is exclusively used with $\sigma = 1$.

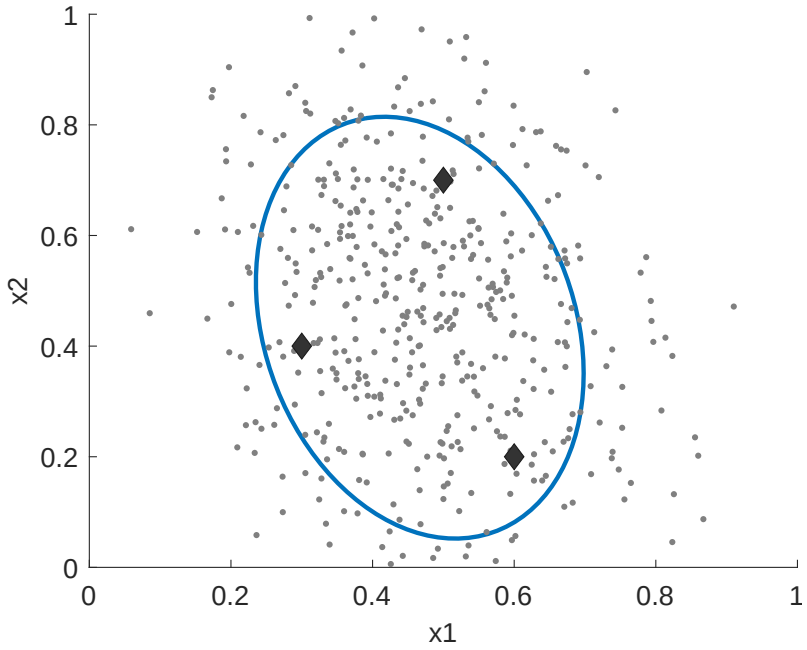


Figure 3.7: Production Scheme of CMAX operator, Dark Diamond Markers: Parent Individuals, Grey Markers: Offspring, Blue Ellipsoid: Covariance

The pattern of the offspring here is clearly mean-centric, since they are sampled around the mean of the population. This operator is also self-adaptive because the covariance matrix adapts itself to the properties of the parents. When they are closer to each other, the ellipsoid is smaller and the sampled offspring are more crowded.

Laplace Crossover

The next crossover operator is the LX operator. The procedure is described in Algorithm 2.6.

Some unwanted artefacts were visible in the visualization of the production scheme. Due to the fact, that the absolute value of the difference of the vectors is used, the information about the direction is lost. The idea of the Laplace crossover is to use a Laplace distribution to sample the offspring. For this, the distribution should be

inverted, but the equation given by Thakur et Al. [19] does not match the desired equation, which leads to gap on the sampling lines.

Due to those artefacts, which are not further discussed by Thakur et Al., we decided to modify this operator, to achieve our desired behaviour. As a first step, the Laplace distribution is inverted

$$F(x) = u = \begin{cases} \frac{1}{2}\exp(\frac{x-a}{b}), & x \leq a \\ 1 - \frac{1}{2}\exp(-\frac{x-a}{b}), & x > a \end{cases} \tag{3.3}$$

with a=0

$$\Leftrightarrow x = \beta = \begin{cases} b \cdot \ln(2u), & u \leq 0.5 \\ -b \cdot \ln(2(1-u)), & u > 0.5 \end{cases} \tag{3.4}$$

We also decided to leave out the absolute value, so that the direction information is used. With the Equation 3.4 as β, the algorithm for our Laplace crossover looks as follows.

Algorithm 3.8: Laplace-Crossover

```
input:
  𝒳,
  D,
  N
initialize: i = 0, 𝒴 = []
begin
  while(i ≠ N)
    foreach j in 1:D
      u = GenerateRandomNumber()
      if(u ≤ 0.5)
        β = b·ln(2u)
      else
        β = −b·ln(2(1−u))
      end
      y_{i,j} = x_{i,j} + β(x_{i,j} − x_{i+1,j})
      y_{i+1,j} = x_{i+1,j} − β(x_{i,j} − x_{i+1,j})
    end
    i = i+2
    𝒴 = 𝒴 ⋃ {y_i, y_{i+1}}
  end
  return 𝒴
end
```

In this work, we now refer to our Laplace crossover when we talk about LX. In Figure 3.8 the offspring generation is visualized. The lines that are drawn from the differences of the parents and the distribution of the offspring along those lines are very noticeable. Due to this distribution, a self-adaptive character very likely to the SBX can be seen. The pattern can be considered as parent-centric, as the offspring are produced around the parents.

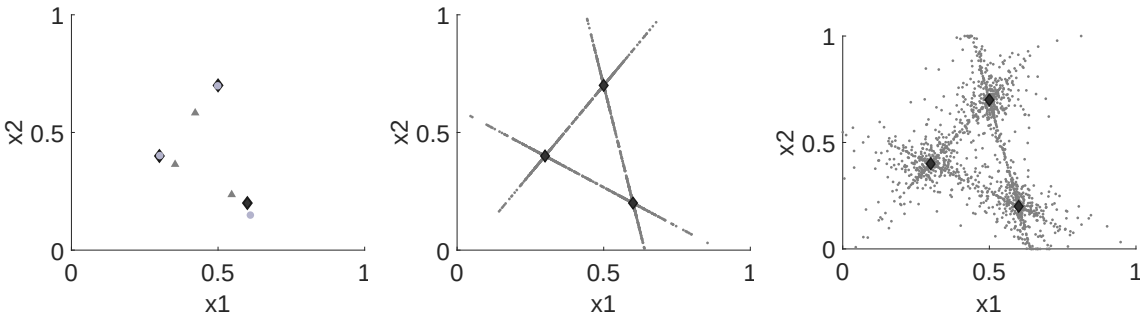


Figure 3.8: Production Scheme of LX operator, dark diamond Markers: Parent Individuals, Grey Markers: Offspring. Left: two Generations, Center: 500 Offspring to show pattern, Right: 500 Offspring with Polynomial Mutation

Linear Combination

The Linear Combination Crossover (LCX) operator makes use of linear combinations of n individuals. This idea is inspired by the SPX operator, described in Algorithm 2.7, that forms a coordinate system independent simplex out of n parents and samples solutions in this simplex. Here, also n parents form a simplex by a linear combination of the parent vectors. The production of offspring with the LCX operator is described in the following.

Algorithm 3.9: Linear Combination Crossover with 3 parents.

input:
  $\mathcal{X}$,
  D,
  N
initialize:
  $\mathcal{Y} = \{\}$
  j = 1
begin
  while(i $\neq$ N)
    foreach k **in** 1:3
      λ(k)=GenerateRandomNumber()
    end
    λ = Normalize(λ)
    foreach j **in** 1:D
      $y_{i,j} = \lambda_1 \cdot x_{i,j} + \lambda_2 \cdot x_{i+1,j} + \lambda_3 \cdot x_{i+2,j}$
      $y_{i+1,j} = \lambda_2 \cdot x_{i,j} + \lambda_3 \cdot x_{i+1,j} + \lambda_1 \cdot x_{i+2,j}$
      $y_{i+2,j} = \lambda_3 \cdot x_{i,j} + \lambda_1 \cdot x_{i+1,j} + \lambda_2 \cdot x_{i+2,j}$
    end
    i=i+3
  $\mathcal{Y} = \mathcal{Y} \bigcup \{\boldsymbol{y}_i, \boldsymbol{y}_{i+1}, \boldsymbol{y}_{i+2}\}$
  end
  return $\mathcal{Y}$
end

In the Figure 3.9 the simplex formed by those three parents is visible. The centre point of the population remains because the offspring are produced around it. Thus, the LCX is mean-centric. Self-adaptiveness is not given for this operator. Because of

the characteristic of the linear combination, the properties of the offspring do not change with the behaviour of the parents.

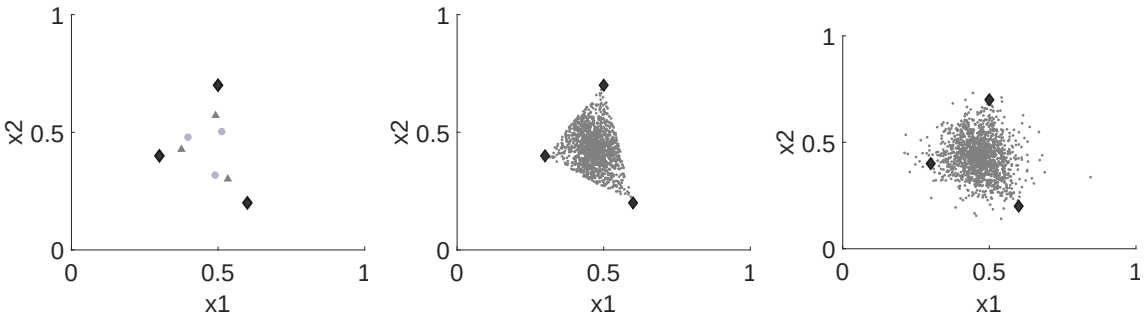


Figure 3.9: Production Scheme of LCX3 operator, dark diamond Markers: Parent Individuals, Grey Markers: Offspring. Left: two Generations, Center: 500 Offspring to show pattern, Right: 500 Offspring with Polynomial Mutation

In this chapter, all used algorithms and methods are described. In the following, those implementations are evaluated by using four different experiments, that shall answer the questions about the progresses made by using hyper-heuristics in evolutionary algorithms.

4. Evaluation

In this chapter, the algorithms presented in Chapter 3 are evaluated through experiments. Those experiments analyse the quality, the feature utilization and the adjustability of the proposed Hyper-Heuristics. First, those experiments are described and afterwards the results are presented and discussed.

4.1 Experimental Setup

In this section, our methods of validating our algorithms are described. We process 20 different benchmark problems with all proposed algorithms. We record the final and intermediate solutions, their quality regarding HV and IGD and additionally the picking behaviour of the Hyper-Heuristics.

So far, we achieved our research and design subgoals. With the experimental analyses, we reach the remaining:

- Perform an experimental evaluation with the selected crossover operators and identify their interplay with different problem's properties
- Perform an experimental evaluation with the proposed Hyper-Heuristics and identify the influence of the components on the result's quality
- Readjust parameters of selected Hyper-Heuristics and analyse the result's quality to show, that further advancements are possible
- Choose the best performing Hyper-Heuristics and compare them to NSGA-II with single crossover operators

We selected DTLZ1-7, RM1-4 and WFG1-9 for our experimental analysis. With these, we cover multiple variants of non-separability, modality and rotation on two and on three objectives. We use three different versions of those problems to examine the quality changes when the complexity is increased. Thus, we have the default configuration, D multiplied with four and D multiplied with six. Another way, to

increase the complexity is by scaling the number of objectives, but as we use only NSGA-II, and it is optimized for $M \leq 3$, this would be pointless for our case.

We let each single crossover operator and each hyper-heuristic crossover operator implemented in NSGA-II solve all benchmark problems, on all difficulty levels, 31 times. By choosing an odd number of runs, we can ensure to have one run being the median run. We record 20 intermediate solutions and therefore 20 intermediate IGD and HV values. We decided to record HV and IGD as metrics to track the convergence and the diversity of the data. As both metrics are not completely independent of each other, we lay the focus on IGD and confirm the data with HV. Additionally, we record the selection probabilities of all HHS and the distribution of all HHD in every generation, as well as the number of offspring per operator in each generation to analyse the picking behaviour and the learning process. With a detailed comparison, our objective is to identify the influence of the different components of the Hyper-Heuristics.

After this first experiment, we analyse the data and select the best HHS and the best HHD. The best algorithms are those that surpass the original NSGA-II using SBX most often. As a second criterion, the number of times the algorithm belongs to the best performing algorithms for one problem is counted. Those algorithms are then adjusted in terms of their different parameters. We create 15 new versions of the Hyper-Heuristic NSGA-II and let them solve the 20 Benchmarks once again.

All experiments and the data collection are implemented for and executed with the PlatEMO Framework [38].

Framework PlatEMO

The PlatEMO Framework is developed by BIMK (Institute of Bioinspired Intelligence and Mining Knowledge) of Anhui University and NICE (Nature Inspired Computing and Engineering Group) of University of Surrey for Matlab [38]. It contains all basic and many additional algorithms, benchmark problems and quality metrics. We utilize the extendability of this framework to implement our algorithms with all their variations as inheritance of the ALGORITHM class and additional benchmark problems inheritance of the PROBLEM class for our intentions. To record the additional data from our Hyper-Heuristics, we extended the SOLUTION class, that is responsible for the processing of generations as well as saving and loading of metrics and solution data.

We furthermore use the feature of data illustration with tables to present all the information in a compact way. In PlatEMO, it is possible to load data from multiple algorithms on multiple benchmarks to compare them with a significance test. We decide to use the rank sum test and highlight all algorithms that are considered as the best performing algorithms on one benchmark. The data visualized in the cells are the median results of the IGD or the HV measurement. Furthermore, the rank sum test is used in a one-to-all comparison, which results are symbolized with markers to show, whether the algorithm performed significantly better than the compare algorithm ($+$), significantly worse ($-$) or approximately equal ($\approx$). We decided to evaluate our data by using the median of the resulting IGD and HV values and the corresponding interquartile range (IQR) values.

In the following section, the results are presented and the different analysation steps conducted.

4.2 Results and Discussion

In this section, the data recorded with the prior described methods are presented and analysed. In the first part in Section 4.2.1, we compare the crossover operators with each other using the median results of the IGD metric, including the IQR. In the second part in Section 4.2.2, the same analysis is done for the four HHDs and the other four HHSs. In the third part in Section 4.2.3, the single crossover operators are compared to the hyper-heuristic operators by regarding IGD over function evaluations on different selected benchmark problems. To get an overview about the behaviour of the Hyper-Heuristics, we compared the trend of the single objective operators to the trend of the selection probabilities or the distribution, respectively. In the fourth part of the evaluation in Section 4.2.4, we utilize the insights from the first experiments to adjust the parameters of the Hyper-Heuristics. By this way, we can give an outlook on possible improvements and refinements. To finalize, the best performing algorithms from the parameter tuning are compared again to the single crossover operators.

By doing those analyses, we answer the questions about improvements and impairments using Hyper-Heuristic online selectors and distributors on crossover operators in MOEA. We furthermore state the behaviour of the operators, that is evoked by the different components of each operator. With the last experiments, we can estimate the potential of Hyper-Heuristics in this scenario and give an outlook on further usages.

4.2.1 Comparison of different Crossover Operators

In this section, the IGD results of the NSGA-II algorithms with single crossover operators are reviewed and analysed. We divided the data into three sets. Each set contains the data of all 20 benchmarks, the seven algorithms and one complexity configuration. Those data are presented in Tables 4.1, 4.2, 4.3 generated by PlatEMO. The corresponding HV results can be found in Appendix A.1 in Tables A.1, A.2, A.3. As explained before, the table shows the median results including their IQR, a marker for the comparison with the original NSGA-II, SBX+NSGA-II, and a highlighting, if the algorithm delivered the best results after the pairwise comparison with the other algorithms.

With this data, we accomplish our third subgoal, to show the behaviour of the different crossover operator and compare them with the research results of their property. We are expecting to see averagely good results from SBX+NSGA-II, as it is widely used for many problems. However, there might be a better option among the other algorithms. We also expect that there is not only one single best operator for all problems. Especially on rotated problems, the RSBX operator, which is specifically for those designed, should care for a better performance than the SBX and the UX operator as both are not rotational invariant. There are other operators that are rotational variant, like the CMAX or the DE operator, that should also perform better on rotated problems than UX and SBX.

In the following, the three tables are described by firstly pointing out the results of the pairwise comparison and afterwards presenting the results of the comparison to the SBX+NSGA-II. After those descriptions, the results of the algorithms with single crossover operators are reasoned and conclusions are drawn.

Default Configuration

Table 4.1 shows the median IGD of all NSGA-II variants with single crossover operators on the 20 selected benchmark problems with default configuration after 10,000 function evaluations (FEs).

Table 4.1: Inverted Generational Distance of NSGA-II with different crossover operators on DTLZ, RM and WFG with their default configurations.

Problem	M	D	CMAX	DE	LCX3	LX	RSBX	UX	SBX
DTLZ1	3	7	1.5969e+1 (7.95e+0) –	2.8847e+0 (2.40e+0) –	8.7530e+0 (4.10e+0) –	1.2019e+1 (7.91e+0) –	5.3914e+0 (3.23e+0) –	1.2288e-1 (2.36e-1) ≈	8.4665e-2 (3.29e-1)
DTLZ2	3	12	7.0360e-2 (4.07e-3) –	7.9601e-2 (4.64e-3) –	2.0318e-1 (5.29e-2) –	9.1055e-2 (1.31e-2) –	7.1440e-2 (4.08e-3) –	6.6472e-2 (2.78e-3) +	6.8717e-2 (3.35e-3)
DTLZ3	3	12	1.8363e+2 (1.01e+2) –	2.2369e+1 (4.09e+1) –	2.1001e+2 (6.84e+1) –	9.3897e+1 (6.22e+1) –	1.0102e+2 (5.07e+1) –	6.9522e+0 (4.14e+0) ≈	6.8909e+0 (5.92e+0)
DTLZ4	3	12	2.4596e-1 (3.29e-1) –	8.6094e-2 (2.81e-2) –	8.7921e-1 (1.44e-1) –	2.7164e-1 (8.30e-2) –	2.5768e-1 (1.71e-1) –	6.8740e-2 (4.76e-1) ≈	6.8306e-2 (7.41e-3)
DTLZ5	3	12	5.7549e-3 (3.12e-4) +	8.2860e-3 (1.36e-3) –	1.3275e-2 (4.78e-3) –	5.9477e-3 (8.17e-4) ≈	5.9568e-3 (4.25e-4) ≈	6.6502e-3 (3.87e-4) –	6.0446e-3 (4.25e-4)
DTLZ6	3	12	3.0560e+0 (1.55e+0) –	6.4816e-1 (9.64e-1) –	8.8431e+0 (1.73e-1) –	1.0927e+0 (7.90e-1) –	4.0312e+0 (1.17e+0) –	2.0490e+0 (3.21e-1) –	5.7703e-3 (5.12e-4)
DTLZ7	3	22	1.7450e+0 (1.03e+0) –	2.0700e+0 (1.23e+0) –	8.6756e+0 (8.46e-1) –	2.6308e+0 (1.33e+0) –	1.8225e+0 (8.80e-1) –	7.7099e-2 (8.95e-3) +	9.6454e-2 (8.99e-3)
RM1	2	30	6.0210e-3 (3.70e-3) +	5.9832e-3 (5.68e-4) +	1.9281e-1 (2.20e-2) +	2.8555e-2 (3.93e-2) +	3.1334e-2 (3.62e-2) +	1.8640e-1 (6.15e-2) +	2.1232e-1 (3.61e-2)
RM2	2	30	5.8398e-3 (9.43e-4) +	6.1630e-3 (6.89e-4) +	2.9883e-1 (2.26e-2) +	1.1966e-2 (5.83e-2) +	1.0868e-1 (6.98e-2) +	3.8181e-1 (1.27e-1) +	4.4772e-1 (1.32e-1)
RM3	2	10	3.1782e-1 (6.24e-2) +	1.9153e-1 (2.12e-1) +	1.0885e+0 (2.88e-1) –	2.0594e-1 (6.40e-2) +	3.9720e-1 (1.73e-1) +	6.2006e-1 (1.60e-1) +	7.0012e-1 (1.25e-1)
RM4	3	12	6.5508e-2 (3.08e-3) +	7.1711e-2 (3.35e-3) +	4.7921e-1 (5.17e-2) –	7.5436e-2 (5.20e-3) +	7.7460e-2 (4.42e-3) +	1.2111e-1 (1.02e-2) –	1.0781e-1 (9.43e-3)
WFG1	3	12	1.9327e+0 (6.89e-2) –	1.6573e+0 (3.38e-2) –	1.9558e+0 (1.32e-1) –	1.5085e+0 (1.21e-1) –	1.5679e+0 (6.92e-2) –	1.1560e+0 (3.19e-1) –	5.5028e-1 (1.18e-1)
WFG2	3	12	2.4955e-1 (1.86e-2) –	2.6647e-1 (2.49e-2) –	4.3423e-1 (1.54e-1) –	2.9625e-1 (4.38e-2) –	2.6186e-1 (2.09e-2) –	2.1135e-1 (1.36e-2) +	2.2702e-1 (1.64e-2)
WFG3	3	12	2.0386e-1 (3.17e-2) –	2.4317e-1 (4.33e-2) –	2.6964e-1 (5.53e-2) –	2.0554e-1 (4.82e-2) –	1.5041e-1 (4.01e-2) –	1.1366e-1 (2.27e-2) +	1.2305e-1 (1.44e-2)
WFG4	3	12	3.0869e-1 (1.12e-2) –	3.3760e-1 (1.87e-2) –	4.3865e-1 (6.80e-2) –	3.3507e-1 (2.23e-2) –	3.1546e-1 (1.31e-2) –	2.5711e-1 (1.45e-2) +	2.7893e-1 (8.19e-3)
WFG5	3	12	3.2482e-1 (4.92e-2) –	2.8678e-1 (2.29e-2) ≈	8.6407e-1 (4.33e-2) –	3.0364e-1 (2.10e-2) –	3.5222e-1 (5.04e-2) –	2.7358e-1 (1.07e-2) +	2.8534e-1 (1.13e-2)
WFG6	3	12	3.1367e-1 (2.97e-2) +	3.7666e-1 (2.54e-2) –	6.4768e-1 (7.94e-2) –	4.3976e-1 (3.69e-2) –	3.3310e-1 (2.00e-2) ≈	3.0709e-1 (2.34e-2) +	3.2699e-1 (2.16e-2)
WFG7	3	12	3.0234e-1 (2.09e-2) –	3.4257e-1 (1.56e-2) –	5.3792e-1 (5.39e-2) –	3.7802e-1 (4.90e-2) –	3.0996e-1 (1.94e-2) –	2.7633e-1 (9.41e-3) ≈	2.8099e-1 (1.44e-2)
WFG8	3	12	4.4703e-1 (3.76e-2) –	4.6373e-1 (2.06e-2) –	8.3457e-1 (1.06e-1) –	5.0655e-1 (4.63e-2) –	4.6527e-1 (2.56e-2) –	3.5677e-1 (1.34e-2) +	3.7599e-1 (1.02e-2)
WFG9	3	12	2.7123e-1 (1.32e-2) +	3.9243e-1 (8.65e-2) –	3.6619e-1 (5.40e-2) –	3.2107e-1 (2.91e-2) –	2.7636e-1 (1.86e-2) +	2.7682e-1 (1.49e-2) +	2.8317e-1 (1.17e-2)
+/−/≈			7/13/0	4/15/1	2/18/0	4/15/1	5/13/2	12/4/4	

In the pairwise comparison, the UX+NSGA-II is visibly superior. It belonged to the best performing algorithms for 13 out of 20 problems. The SBX+NSGA-II and the CMAX+NSGA-II belongs six times to the best performing algorithms, NSGA-II with DE operator three times and with RSBX one time. The combinations with LCX3 and LX are never part of the best performing algorithms in terms of IGD. The corresponding table for the HV results can be found in Appendix A.1 in Table A.1. The HV results confirm these descriptions, except for DTLZ3, where no algorithm can deliver satisfying HV results. Therefore, all algorithms are marked as equally performing.

Although the SBX+NSGA-II cannot compete with the UX+NSGA-II, it delivers noticeable better results than most of the other operators on most problems. As already noted, the SBX is not rotational invariant and thus, it has trouble finding good solutions to the rotated problems, RM1-4. The HV results confirm this experience and furthermore shows, that the combinations with the DE and the SBX operators are the only algorithms to find satisfying results on DTLZ6. Similar happens on DTLZ7, where LX+NSGA-II and LCX3+NSGA-II do not deliver sufficient results and again on RM3 with the combinations with LCX3, UX and SBX. UX+NSGA-II is either better or equally well as the SBX+NSGA-II, except for DTLZ5, DTLZ6, RM4 and WFG1. Besides the rotated problems, the combination with SBX is also surpassed by CMAX+NSGA-II on DTLZ5, WFG6 and WFG9 and by RSBX+NSGA-II on WFG9. The algorithm using the LXC3 operators performs even worse than the SBX+NSGA-II on RM3 and RM4. Furthermore, UX+NSGA-II undercut SBX+NSGA-II on RM4. Over all problems, the UX+NSGA-II is 12 times significantly better, four times significantly worse and four times approximately equally good as the SBX+NSGA-II. The combination with the RSBX operator is only on the rotated problem and on WFG9 significantly better and 13 times, the SBX+NSGA-II is better. The usage of the LX operator lead to significantly worse results for almost all not rotated problems. Only on DTLZ5, it performs nearly equally to SBX+NSGA-II. The LCX3+NSGA-II is only on two rotated problems, RM1 and RM2, better than the SBX+NSGA-II. On the remaining 18 problems, SBX+NSGA-II performs significantly better. The combination with the DE operator lead to similar results as the LX operator. It surpassed the SBX+NSGA-II on the rotated problems and performs approximately equal on WFG5. CMAX+NSGA-II is the second best regarding the pairwise comparison, as well as the SBX+NSGA-II, and is also in direct comparison to the SBX+NSGA-II the second best. Besides the four rotated problems, the result quality is significantly higher on DTLZ5, WFG6 and WFG9.

Number of Decision Variables multiplied by four

Table 4.2 shows the median IGD of all NSGA-II variants with single crossover operators on the 20 selected benchmark problems, with all D multiplied by 4 after 10,000 FEs.

In the pairwise comparison, again, UX+NSGA-II outperforms the other algorithms multiple times. It is superior on 12 out of 20 problems. Surprisingly, one of these 12 problems is RM3 as rotated problem. Although, the IGD values are comparably high for all algorithms on that problem. The HV results, in the corresponding Table A.2, show that no algorithm can find sufficient solutions on RM3 as well as on

Table 4.2: Inverted Generational Distance of NSGA-II with different Hyper-Heuristic crossover operators on DTLZ, RM and WFG with a number of decision variables multiplied by 4.

Problem	M	D	CMAX	DE	LCX3	LX	RSBX	UX	SBX
DTLZ1	3	28	1.3051e+2 (1.54e+2) −	1.3657e+1 (3.56e+1) +	4.5758e+2 (5.23e+1) −	8.7842e+1 (3.46e+1) −	4.0550e+2 (5.89e+1) −	2.8021e+1 (6.82e+0) +	4.1689e+1 (1.11e+1)
DTLZ2	3	48	2.5952e-1 (4.08e-2) −	3.5053e-1 (4.91e-2) −	6.5261e-1 (2.23e-1) −	5.1957e-1 (7.57e-2) −	1.7191e-1 (3.90e-2) −	8.4655e-2 (6.87e-3) +	1.0340e-1 (1.02e-2)
DTLZ3	3	48	2.1849e+2 (2.24e+2) +	1.2427e+2 (3.16e+2) +	2.8606e+3 (2.35e+2) −	5.3459e+2 (1.66e+2) −	2.6124e+3 (2.82e+2) −	2.6012e+2 (4.50e+1) +	3.8932e+2 (9.09e+1)
DTLZ4	3	48	9.7893e-1 (4.31e-2) −	7.0977e-1 (1.63e-1) −	1.0381e+0 (4.45e-2) −	1.0583e+0 (1.96e-1) −	7.7333e-1 (1.31e-1) −	9.1171e-2 (4.66e-1) +	1.2286e-1 (4.34e-1)
DTLZ5	3	48	7.4963e-2 (3.53e-2) −	1.8644e-1 (4.41e-2) −	1.5515e-1 (2.72e-2) −	2.3802e-1 (9.42e-2) −	5.9278e-2 (2.33e-2) −	2.9694e-2 (9.06e-3) +	3.8987e-2 (1.16e-2)
DTLZ6	3	48	2.0908e+1 (2.47e+0) −	1.6154e+1 (2.54e+0) ≈	4.1300e+1 (4.17e-1) −	1.5146e+1 (1.62e+0) +	2.5873e+1 (4.28e+0) −	2.7700e+1 (9.95e-1) −	1.7075e+1 (2.07e+0)
DTLZ7	3	88	7.9289e+0 (8.57e-1) −	7.3604e+0 (8.31e-1) −	1.0457e+1 (8.11e-1) −	8.4172e+0 (1.04e+0) −	7.8837e+0 (6.76e-1) −	6.3786e-1 (6.39e-2) +	8.4442e-1 (1.85e-1)
RM1	2	120	1.8591e-1 (1.08e-2) +	1.9627e-1 (1.12e-2) +	2.2002e-1 (3.04e-2) +	1.9694e-1 (9.63e-3) +	1.8536e-1 (9.50e-3) +	3.1063e-1 (5.89e-2) ≈	3.2392e-1 (6.06e-2)
RM2	2	120	2.9658e-1 (1.55e-2) +	3.0784e-1 (1.10e-2) +	3.3557e-1 (5.60e-2) +	3.1735e-1 (1.73e-2) +	2.9888e-1 (1.39e-2) +	5.0495e-1 (2.20e-2) +	5.1121e-1 (1.62e-2)
RM3	2	40	3.2367e+0 (2.35e-1) −	3.8133e+0 (3.38e-1) −	2.8294e+0 (2.80e-1) ≈	3.0036e+0 (3.68e-1) −	2.3261e+0 (3.52e-1) +	2.1805e+0 (3.55e-1) +	2.6871e+0 (3.79e-1)
RM4	3	48	6.2730e-1 (1.21e-1) −	3.2677e-1 (1.59e-1) +	5.9346e-1 (1.91e-1) −	8.1436e-1 (1.72e-1) −	8.5435e-1 (1.91e-1) −	5.4186e-1 (6.55e-2) ≈	5.4905e-1 (7.04e-2)
WFG1	3	48	1.9930e+0 (3.73e-2) −	1.6952e+0 (2.95e-2) −	2.1803e+0 (1.04e-1) −	1.5957e+0 (4.56e-2) −	1.6317e+0 (7.44e-2) −	1.5766e+0 (7.86e-2) −	1.2488e+0 (7.84e-2)
WFG2	3	48	3.9027e-1 (3.06e-2) −	4.2026e-1 (4.11e-2) −	6.7758e-1 (1.72e-1) −	4.5944e-1 (4.19e-2) −	4.2226e-1 (3.80e-2) −	2.4771e-1 (1.92e-2) +	2.9229e-1 (3.70e-2)
WFG3	3	48	4.0666e-1 (3.88e-2) −	4.9583e-1 (3.71e-2) −	4.4644e-1 (3.89e-2) −	4.2654e-1 (5.13e-2) −	3.9635e-1 (6.35e-2) −	2.5306e-1 (3.79e-2) +	3.2445e-1 (2.96e-2)
WFG4	3	48	3.4066e-1 (1.29e-2) −	3.9480e-1 (1.85e-2) −	5.5746e-1 (8.43e-2) −	3.8044e-1 (2.31e-2) −	3.5379e-1 (1.65e-2) −	2.6542e-1 (1.47e-2) +	3.1997e-1 (1.29e-2)
WFG5	3	48	4.0457e-1 (1.10e-1) −	3.0355e-1 (1.90e-2) +	9.7587e-1 (5.43e-2) −	3.1485e-1 (2.22e-2) +	5.3270e-1 (4.56e-2) −	3.0396e-1 (1.24e-2) +	3.4070e-1 (2.12e-2)
WFG6	3	48	4.9574e-1 (3.36e-2) −	3.2251e-1 (4.17e-2) +	7.9809e-1 (5.57e-2) −	5.1765e-1 (1.47e-1) −	5.4631e-1 (2.82e-2) −	3.2196e-1 (2.16e-2) +	3.7961e-1 (2.32e-2)
WFG7	3	48	4.2011e-1 (2.67e-2) ≈	4.9870e-1 (2.38e-2) −	6.3657e-1 (4.81e-2) −	4.8521e-1 (3.10e-2) −	4.4220e-1 (2.95e-2) ≈	4.1445e-1 (9.14e-2) ≈	4.3422e-1 (8.50e-2)
WFG8	3	48	4.9337e-1 (3.06e-2) −	5.6502e-1 (3.10e-2) −	8.1349e-1 (9.89e-2) −	5.7695e-1 (3.85e-2) −	5.1354e-1 (1.87e-2) −	3.3915e-1 (1.88e-2) +	3.8080e-1 (1.28e-2)
WFG9	3	48	3.5803e-1 (2.62e-2) +	3.4441e-1 (3.63e-2) +	4.3033e-1 (6.82e-2) −	3.7866e-1 (4.83e-2) +	3.7190e-1 (2.50e-2) +	3.8706e-1 (4.16e-2) ≈	4.0673e-1 (3.55e-2)
+/−/≈			4/15/1	8/11/1	2/17/1	5/15/0	4/15/1	14/2/4	

DTLZ1, DTZL3 and DTLZ6. The distribution of the markers for the best performing algorithms is visibly more distributed than in Table 4.1. SBX+NSGA-II belongs only twice to the best performing algorithms. The combinations with RSBX and CMAX are three times significantly better than all other algorithms, thus the CMAX has greater loss in quality, than the RSBX. They perform nearly equally on RM1, RM2 and WFG7. The quality reached with the DE operator seems to suffer less from the increase of D than the other algorithms, as it surpasses the others six times. This time the combination with LX also belongs once to the best performing algorithms. LCX3+NSGA-II remains with no belongings to the best performing algorithms.

The original NSGA-II using SBX performs also in this setting significantly better than most algorithms on most problems, excluding UX+NSGA-II. Exceptions are again the rotated problems, RM1-4, and WFG9, where only the LCX3+NSGA-II is worse. UX+NSGA-II outperforms SBX+NSGA-II 14 times, and performs worse on WFG1 and DTLZ6. RSBX+NSGA-II has a decrease in quality in RM4, compared to the easier setting, so that the SBX+NSGA-II performs significantly better. It still beats the original NSGA-II in RM1-3 and WFG9. LX+NSGA-II performs better than SBX+NSGA-II on DTLZ6, RM1, RM2, WFG6 and WFG9. LCX3+NSGA-II has similar results as for the default settings. It is mostly outperformed by SBX+NSGA-II. The combination with the DE operator seem to tolerate the increase of D more than the other algorithms. In the other setting, DE+NSGA-II performed better than the original NSGA-II on all rotated problems. In this setting, the algorithm sacrifices the superior performance on RM3, but therefore, accomplishes better results than SBX+NSGA-II on DTLZ1, DTLZ3, WFG5, WFG6 and WFG9. The CMAX+NSGA-II on the other hand loses more to the increase of D and in contrast to the previous setting performs worse than SBX on RM3, RM4, WFG6 and DTLZ5. It performs better than SBX+NSGA-II only on 4 problems: DTLZ3, RM1, RM2 and WFG9.

Number of Decision Variables multiplied by six

Table 4.3 shows the median IGD of all NSGA-II variants with single crossover operators on the 20 selected benchmark problems with all D multiplied by 6 after 10,000 FEs.

In the pairwise comparison, the original NSGA-II belongs only twice to the best performing algorithms. It remains the best on WFG1, but loses inn WFG7, where the UX+NSGA-II is the solely best performing algorithm. The SBX+NSGA-II is together with the UX+NSGA-II the best performing algorithm on DTLZ4. The UX+NSGA-II looses more of its superiority and surpasses the other algorithms in only 9 out of 20 problems. It still belongs more often to the best performing algorithms than the others. It is followed by CMAX+NSGA-II and DE+NSGA-II, with 6 out of 20 problems. After these, the combination with RSBX follows with three problems, then LX and SBX with two and LCX3 with one. The comparison with the corresponding Table A.3 shows that all algorithms have problems finding sufficient results on DTLZ1, DTLZ3, DTLZ4, DTLZ6, DTLZ7, RM3 and RM4. On DTLZ4 and DTLZ7, only SBX+NSGA-II and UX+NSGA-II found results, for which the HV values are calculable. On RM4, the combinations with CMAX, DE and LCX3 found solutions. This confirms the IGD results, as also here, the combinations with CMAX, DE and LCX3 performed better than other algorithms.

UX+NSGA-II is also in this setting superior to SBX+NSGA-II, as it outperforms the original algorithm in 15 cases. It is only worse on DTLZ6 and WFG1, where SBX+NSGA-II is the solely best performing algorithm. The combination with RSBX performs better than SBX+NSGA-II on three rotated problems, RM1-3, WFG7 and WFG9. On all other problems, the original algorithm is better. Although, the LX+NSGA-II rarely belongs to the best performing algorithms, it achieves higher IGD values than SBX+NSGA-II on seven problems and is approximately equal on two problems. The combination with LCX3 is the only algorithm that is consequently

Table 4.3: Inverted Generational Distance of NSGA-II with different Hyper-Heuristic crossover operators on DTLZ, RM and WFG with a number of decision variables multiplied by 6.

Problem	M	D	CMAX	DE	LCX3	LX	RSBX	UX	SBX
DTLZ1	3	42	2.5311e+2 (2.48e+2) −	6.0695e+1 (9.88e+1) +	8.6499e+2 (8.57e+1) −	1.5372e+2 (5.34e+1) ≈	7.5703e+2 (7.44e+1) −	6.5606e+1 (1.37e+1) +	1.3129e+2 (1.38e+1)
DTLZ2	3	72	3.8767e-1 (9.02e-2) −	5.2294e-1 (6.21e-2) −	6.9902e-1 (2.97e-1) −	8.4427e-1 (1.88e-1) −	3.2205e-1 (5.11e-2) −	1.3484e-1 (1.76e-2) +	1.8607e-1 (3.19e-2)
DTLZ3	3	72	2.4204e+2 (3.16e+2) +	1.0462e+2 (7.19e+2) +	4.9989e+3 (2.40e+2) −	8.3723e+2 (2.94e+2) +	4.5281e+3 (1.83e+2) −	5.2021e+2 (5.75e+1) +	9.2333e+2 (1.48e+2)
DTLZ4	3	72	1.0291e+0 (5.19e-2) −	1.0719e+0 (1.52e-1) −	1.1110e+0 (3.58e-2) −	1.2644e+0 (1.62e-1) −	1.0315e+0 (1.29e-1) −	5.5125e-1 (4.32e-1) ≈	2.2814e-1 (3.54e-1)
DTLZ5	3	72	1.7056e-1 (6.63e-2) −	3.3032e-1 (5.37e-2) −	2.4868e-1 (6.27e-2) −	4.9986e-1 (1.63e-1) −	1.3549e-1 (3.41e-2) −	8.3417e-2 (2.00e-2) +	1.2424e-1 (2.08e-2)
DTLZ6	3	72	3.3940e+1 (3.72e+0) +	2.7987e+1 (3.89e+0) +	6.3021e+1 (8.36e-1) −	2.5363e+1 (3.95e+0) +	4.2308e+1 (4.79e+0) −	4.6531e+1 (1.42e+0) −	3.6696e+1 (2.51e+0)
DTLZ7	3	132	8.5585e+0 (7.92e-1) −	8.0677e+0 (7.84e-1) −	1.0869e+1 (6.03e-1) −	9.3632e+0 (7.00e-1) −	8.6089e+0 (6.06e-1) −	9.6698e-1 (1.19e-1) +	1.2437e+0 (1.83e-1)
RM1	2	180	1.9062e-1 (9.06e-3) +	2.0112e-1 (1.42e-2) +	2.4027e-1 (4.90e-2) +	2.0280e-1 (7.08e-3) +	1.8989e-1 (9.35e-3) +	3.2459e-1 (3.08e-2) +	3.6452e-1 (5.11e-2)
RM2	2	180	2.9752e-1 (1.12e-2) +	3.1932e-1 (1.82e-2) +	3.7034e-1 (7.10e-2) +	3.1774e-1 (1.27e-2) +	3.0044e-1 (9.20e-3) +	5.0850e-1 (1.32e-2) +	5.2538e-1 (2.54e-2)
RM3	2	60	3.7888e+0 (3.06e-1) ≈	4.6489e+0 (3.68e-1) −	3.2932e+0 (2.48e-1) +	3.9711e+0 (4.69e-1) ≈	3.1537e+0 (3.01e-1) +	3.3132e+0 (5.16e-1) +	3.8126e+0 (3.05e-1)
RM4	3	72	7.3778e-1 (1.69e-1) +	6.7975e-1 (2.04e-1) +	6.5791e-1 (2.06e-1) +	1.3421e+0 (5.54e-1) −	1.6739e+0 (7.40e-1) −	9.8500e-1 (1.58e-1) ≈	1.0207e+0 (2.28e-1)
WFG1	3	72	2.0093e+0 (4.98e-2) −	1.6781e+0 (4.73e-2) −	2.2534e+0 (1.10e-1) −	1.5956e+0 (3.35e-2) −	1.6407e+0 (6.66e-2) −	1.6602e+0 (7.06e-2) −	1.3664e+0 (2.40e-2)
WFG2	3	72	4.0896e-1 (2.50e-2) −	4.3610e-1 (2.67e-2) −	6.4541e-1 (1.85e-1) −	4.8117e-1 (3.07e-2) −	4.3960e-1 (3.79e-2) −	2.7664e-1 (2.57e-2) +	3.3539e-1 (3.55e-2)
WFG3	3	72	4.3099e-1 (1.73e-2) −	5.1395e-1 (5.18e-2) −	4.8663e-1 (4.35e-2) −	4.7177e-1 (3.57e-2) −	4.3179e-1 (4.52e-2) −	3.1537e-1 (5.66e-2) +	4.0474e-1 (2.58e-2)
WFG4	3	72	3.4769e-1 (9.83e-3) −	4.0360e-1 (1.56e-2) −	5.8176e-1 (9.75e-2) −	3.8644e-1 (2.52e-2) −	3.5680e-1 (1.81e-2) −	2.7976e-1 (8.76e-3) +	3.3633e-1 (1.91e-2)
WFG5	3	72	4.1172e-1 (6.95e-2) −	3.0500e-1 (2.80e-2) +	9.8791e-1 (4.17e-2) −	3.2809e-1 (2.16e-2) +	5.5608e-1 (7.51e-2) −	3.3180e-1 (2.10e-2) +	3.8296e-1 (1.87e-2)
WFG6	3	72	5.2914e-1 (2.36e-2) −	3.1023e-1 (3.76e-2) +	8.3696e-1 (9.16e-2) −	5.1919e-1 (1.78e-1) −	5.6856e-1 (2.71e-2) −	3.5795e-1 (1.74e-2) +	4.2604e-1 (2.14e-2)
WFG7	3	72	4.4485e-1 (1.46e-2) +	5.0979e-1 (1.91e-2) ≈	6.6984e-1 (7.01e-2) −	4.9724e-1 (2.21e-2) +	4.6081e-1 (1.61e-2) +	4.9388e-1 (1.90e-2) +	5.0923e-1 (2.79e-2)
WFG8	3	72	4.9581e-1 (1.99e-2) −	5.6732e-1 (2.59e-2) −	7.5757e-1 (6.56e-2) −	5.7623e-1 (3.48e-2) −	5.1657e-1 (2.08e-2) −	3.4894e-1 (1.37e-2) +	4.0378e-1 (1.93e-2)
WFG9	3	72	3.7477e-1 (2.84e-2) +	3.7228e-1 (3.31e-2) +	4.3836e-1 (6.96e-2) ≈	3.7268e-1 (2.83e-2) +	3.9616e-1 (2.96e-2) +	4.4567e-1 (4.52e-2) ≈	4.5999e-1 (4.19e-2)
+/−/≈			7/12/1	9/10/1	4/15/1	7/11/2	5/15/0	15/2/3	

superior to the original algorithm on all four rotated problems. It cannot keep up with the other algorithms on the other problems. The combinations with DE and CMAX operators suffer less from the increase of D, and thus they perform better than the original algorithm on more problems. The CMAX+NSGA-II outperforms SBX+NSGA-II seven times, the DE+NSGA-II outperforms the SBX+NSGA-II nine times. This trend is confirmed by the HV results in Table A.3.

Evaluation

At the beginning of this section, we stated different expectations on the results. The first expectation arose from the researches we made on rotated problems. We

anticipate that the rotational invariant problems perform better on rotated problem compared to UX and SBX, which are not rotational variant. As explained before, the RSBX outperforms the SBX on RM1-3 on all levels of complexity. On RM4, the SBX performs better than the RSBX on higher complexities. The corresponding work from Pan et Al. [14] receives similar results, and thus we can confirm it. We know from our researches that the SBX converges faster than the RSBX. Therefore, on RM3 and RM4, the solutions in objective space were far enough from the $\mathcal{PF}$, that the rotation had not yet influence on the result. This is confirmed by the high HV measurements for RM3 and RM4 on higher complexities. Thus, we assume, if we use more FEs for the problems RM3 and RM4, the results would be more unambiguous.

Furthermore, the other rotational invariant operators, CMAX and DE, performed better than the SBX operator and mostly better than the RSBX on the rotated problems. This meets our expectation, that the RSBX is not always the best choice for rotated problems.

Additionally, we expected that the SBX operator, as a well-known and widely used crossover operator, would perform average well on most problems. This is confirmed, as the SBX outperforms most operators on most problems. Nevertheless, it is mostly topped by the UX operator, which is the superior operator in this experimental setting. Both of these operators are known to have good exploratory behaviour. The SBX operator is self-adaptive, which offers a good balance between exploration and exploitation. The UX operator, on the other hand, is not self-adaptive, and nonetheless performs better. This draws particularly high attention to the multimodal problems DTLZ1, DTLZ3, WFG2, WFG4 and WFG9. The UX operator can be used on multimodal problems as it spreads well, even when it is converging to an optimum. By this way, it is not as fast stuck in local optima as the SBX operator, that converges faster into an optimum due to its self-adaptive behaviour. Therefore, those results can be a hint, that self-adaptive and other fast converging operators have problems solving a multimodal problem and well spreading operators would find better solutions. The self-adaptive operators, LX, RSBX and SBX, have problems solving the five multimodal problems. Additionally, LCX limits the space for offspring very strictly so that it converges from the beginning and has nearly no exploration behaviour. Thus, finding good solutions to any problem is an enormous difficulty for the LCX3. In comparison, the SPX operator, which inspired the design of the LCX operators, adds an offset to the simplex and thus supports a spreading of offspring. This advantage was lost in the simplification of the SPX, which explains the overall bad results of the LCX3. Another special case is the CMAX operator, that moves the population as a whole through the solutions space. Thus, around the centre of the population the offspring are well distributed, but if the whole population found a local optimum, it would converge fast. Therefore, the CMAX operator has also difficulties on multimodal problems, but as it has a good exploration behaviour in the beginning, it still performs averagely well.

It is also interesting, which trends are visible with an increasing $\boldsymbol{D}$. First, with more decision variables, more solutions are possible. Thus, the solution space grows with increasing $\boldsymbol{D}$. With a growing space, a good exploration phase is indispensable. Of course, the UX fulfils this condition, as mentioned before. But also the CMAX can profit from its wider spread of solutions. Moreover, self-adaptiveness with a strong

exploration can be an advantage. Thus, the LX and the RSBX performed better than the SBX on several problems. The increase in complexity was expected to offer more visible differences between the operator features. This expectation is met, as we can see that more operators belong to the best performing algorithms. We can also see, that not all operators exploit their features at the same rate. The SBX, for example, falls behind more operators than in the prior experiments. We also discovered that many problems, especially multimodal problems, are too difficult with an increased D, as we did not receive measurements of the HV.

The most interesting results are found in the medium setting, with D multiplied by four, as the feature's influence on the quality becomes more visible. As well as this influence becomes more visible, the effects of the complexity of the problems is noticeable. Therefore, many problems are too difficult with a D multiplied by six, and thus, the meaningfulness of the data collected from these experiments is decreased.

With these results, we achieved our third subgoal in identifying the interplays between crossover operator features and problem properties. In the next section, we analyse the results of the HHSs and the HHDs.

4.2.2 Comparison of different Hyper-Heuristic Selectors and Distributors

In this section, the IGD results of the NSGA-II algorithms with Hyper-Heuristic crossover operators are reviewed and analysed. We divided the data, as before, into three sets. Each set contains the data of all 20 Benchmarks, the eight algorithms and one complexity configuration. Those data are presented in Tables 4.4, 4.5, 4.6 generated by PlatEMO. The corresponding HV results can be found in Appendix A.1 in Tables A.4, A.5, A.6. Those tables show again the median results including their IQR, a marker for the comparison with the original NSGA-II, and a highlighting, if the algorithm delivered the best results after the pairwise comparison with the other algorithms.

With this data, we accomplish the first part of the fourth subgoal, to show the behaviour of the different Hyper-Heuristics and give an overview about the quality. We expect to see, that for most problems, Hyper-Heuristics perform better than the original algorithm because they find a better alternative. They are not all always better, especially the HHDs because they always use all operators, including those, that do not work well with the problem. Both URX Hyper-Heuristic operators are working randomly, so there will be problems where they perform well on, but we expect that they are mostly on average.

In the following, the three tables are described by firstly pointing out the results of the pairwise comparison and afterwards presenting the results of the comparison to the SBX+NSGA-II. After those descriptions, the results of the algorithms with Hyper-Heuristic crossover operators are reasoned and conclusions are drawn.

Default Configurations

Table 4.4 shows the median IGD of all NSGA-II variants with Hyper-Heuristic crossover operators on the 20 selected benchmark problems with default configurations after 10,000 FEs.

Table 4.4: Inverted Generational Distance of NSGA-II with different Hyper-Heuristic crossover operators on DTLZ, RM and WFG with their default configurations.

Problem	*M*	*D*	NCRXD	NCRXS	R2XD	R2XS	SRXD	SRXS	URXD	URXS	SBX
DTLZ1	3	7	5.3279e-1 (5.49e-1) −	3.5961e-2 (3.42e-2) +	1.1541e-1 (2.76e-1) ≈	2.9744e-1 (3.07e-1) ≈	3.6487e-2 (2.45e-1) +	9.7908e-1 (9.44e-1) −	3.8643e-1 (3.55e-1) −	3.3312e-1 (3.17e-1) −	8.4665e-2 (3.29e-1)
DTLZ2	3	12	6.8147e-2 (3.21e-3) ≈	6.6719e-2 (2.42e-3) +	6.7472e-2 (3.65e-3) ≈	6.8394e-2 (4.10e-3) ≈	6.7571e-2 (3.72e-3) +	6.7789e-2 (3.00e-3) ≈	6.7044e-2 (3.40e-3) +	6.7132e-2 (3.42e-3) +	6.8717e-2 (3.35e-3)
DTLZ3	3	12	2.2724e+1 (1.77e+1) −	4.9280e+0 (3.62e+0) ≈	7.8037e+0 (6.57e+0) ≈	9.1254e+0 (1.11e+1) −	4.0146e+0 (5.22e+0) +	5.0475e+1 (4.24e+1) −	1.5547e+1 (9.22e+0) −	2.2042e+1 (1.62e+1) −	6.8909e+0 (5.92e+0)
DTLZ4	3	12	6.6668e-2 (3.41e-3) ≈	6.5580e-2 (2.41e-3) +	6.7106e-2 (3.58e-3) ≈	6.7423e-2 (3.13e-3) ≈	6.6730e-2 (3.57e-3) ≈	6.9251e-2 (3.89e-3) ≈	6.6807e-2 (3.46e-3) ≈	6.7573e-2 (3.31e-3) ≈	6.8306e-2 (7.41e-3)
DTLZ5	3	12	5.7501e-3 (3.43e-4) +	5.5693e-3 (4.30e-4) +	5.5727e-3 (3.52e-4) +	5.6369e-3 (5.42e-4) +	5.3826e-3 (3.85e-4) +	5.9537e-3 (7.33e-4) ≈	5.5694e-3 (3.17e-4) +	5.5614e-3 (3.03e-4) +	6.0446e-3 (4.25e-4)
DTLZ6	3	12	5.7066e-3 (3.65e-4) +	5.5891e-3 (9.84e-4) ≈	5.5667e-3 (3.45e-4) +	5.6422e-3 (6.79e-4) ≈	5.4727e-3 (4.01e-4) +	5.7241e-3 (9.30e-4) ≈	5.4928e-3 (4.06e-4) +	5.4605e-3 (6.18e-4) +	5.7703e-3 (5.12e-4)
DTLZ7	3	22	1.4960e-1 (3.99e-2) −	1.2464e-1 (6.96e-2) −	8.7013e-2 (1.25e-2) +	9.3072e-2 (2.14e-2) ≈	9.3170e-2 (1.64e-2) ≈	3.7957e-1 (2.77e-1) −	1.5312e-1 (4.58e-2) −	2.0829e-1 (5.47e-2) −	9.6454e-2 (8.99e-3)
RM1	2	30	5.9525e-3 (2.35e-3) +	5.6636e-3 (5.88e-3) +	1.5268e-2 (3.15e-2) +	9.0645e-3 (2.61e-2) +	3.4713e-2 (3.10e-2) +	5.8917e-3 (2.90e-3) +	5.2268e-3 (1.29e-3) +	5.8673e-3 (2.94e-3) +	2.1232e-1 (3.61e-2)
RM2	2	30	8.9447e-3 (7.67e-3) +	5.8975e-3 (7.18e-3) +	1.2162e-2 (2.02e-2) +	6.6419e-3 (1.55e-2) +	1.6821e-2 (4.32e-2) +	6.6641e-3 (5.08e-3) +	5.2772e-3 (5.19e-4) +	6.0367e-3 (2.92e-3) +	4.4772e-1 (1.32e-1)
RM3	2	10	1.7615e-1 (3.68e-2) +	1.6141e-1 (4.75e-2) +	1.7197e-1 (1.38e-1) +	1.9142e-1 (7.42e-2) +	1.4871e-1 (2.82e-2) +	2.1140e-1 (4.68e-2) +	1.7859e-1 (3.86e-2) +	1.8215e-1 (4.06e-2) +	7.0012e-1 (1.25e-1)
RM4	3	12	7.1591e-2 (2.90e-3) +	6.9104e-2 (4.57e-3) +	6.8742e-2 (2.68e-3) +	6.7736e-2 (4.66e-3) +	6.8487e-2 (5.14e-3) +	7.0849e-2 (4.66e-3) +	6.9112e-2 (1.76e-3) +	6.8995e-2 (2.22e-3) +	1.0781e-1 (9.43e-3)
WFG1	3	12	9.1365e-1 (2.24e-1) −	5.0493e-1 (1.60e-1) ≈	5.9522e-1 (1.97e-1) −	7.2207e-1 (1.70e-1) −	5.0569e-1 (1.34e-1) ≈	1.1594e+0 (2.34e-1) −	9.0120e-1 (1.79e-1) −	9.8796e-1 (1.50e-1) −	5.5028e-1 (1.18e-1)
WFG2	3	12	2.1721e-1 (1.20e-2) +	2.1336e-1 (1.09e-2) +	2.1542e-1 (1.33e-2) +	2.1806e-1 (1.65e-2) ≈	2.1508e-1 (1.20e-2) +	2.2213e-1 (1.31e-2) ≈	2.1851e-1 (1.05e-2) +	2.1412e-1 (1.59e-2) +	2.2702e-1 (1.64e-2)
WFG3	3	12	1.3038e-1 (2.57e-2) ≈	1.2149e-1 (2.52e-2) ≈	1.1818e-1 (3.68e-2) ≈	1.2665e-1 (2.76e-2) ≈	1.2351e-1 (3.31e-2) ≈	1.2978e-1 (2.05e-2) ≈	1.1983e-1 (2.86e-2) ≈	1.2840e-1 (2.03e-2) ≈	1.2305e-1 (1.44e-2)
WFG4	3	12	2.9686e-1 (9.08e-3) −	2.6741e-1 (1.42e-2) +	2.9347e-1 (2.10e-2) −	2.9610e-1 (1.44e-2) −	2.8934e-1 (1.48e-2) −	2.9927e-1 (1.58e-2) −	2.9421e-1 (1.10e-2) −	2.9815e-1 (1.37e-2) −	2.7893e-1 (8.19e-3)
WFG5	3	12	3.0864e-1 (1.61e-2) −	2.7148e-1 (7.52e-3) +	2.8459e-1 (1.38e-2) ≈	2.9327e-1 (2.58e-2) −	2.7888e-1 (1.22e-2) +	3.0534e-1 (2.56e-2) −	2.9318e-1 (1.24e-2) −	2.9321e-1 (2.00e-2) −	2.8534e-1 (1.13e-2)
WFG6	3	12	3.0074e-1 (1.23e-2) +	2.8743e-1 (2.02e-2) +	3.0355e-1 (2.74e-2) +	3.0677e-1 (1.87e-2) +	3.0057e-1 (1.50e-2) +	3.0107e-1 (1.70e-2) +	2.9880e-1 (1.30e-2) +	2.9524e-1 (2.87e-2) +	3.2699e-1 (2.16e-2)
WFG7	3	12	2.7894e-1 (1.47e-2) ≈	2.7704e-1 (1.11e-2) ≈	2.7938e-1 (2.11e-2) ≈	2.7798e-1 (1.17e-2) ≈	2.7358e-1 (1.43e-2) +	2.8248e-1 (1.73e-2) ≈	2.7587e-1 (1.29e-2) ≈	2.8032e-1 (1.64e-2) ≈	2.8099e-1 (1.44e-2)
WFG8	3	12	3.9317e-1 (1.49e-2) −	3.7810e-1 (1.40e-2) ≈	3.9709e-1 (2.83e-2) −	3.9884e-1 (2.52e-2) −	4.2745e-1 (2.74e-2) −	3.9779e-1 (1.90e-2) −	3.9448e-1 (2.11e-2) −	3.9320e-1 (2.36e-2) −	3.7599e-1 (1.02e-2)
WFG9	3	12	2.7156e-1 (9.00e-3) +	2.7170e-1 (1.49e-2) +	2.7723e-1 (1.33e-2) +	2.7528e-1 (1.16e-2) +	2.7570e-1 (1.77e-2) +	2.6560e-1 (1.17e-2) +	2.7080e-1 (1.54e-2) +	2.7502e-1 (1.40e-2) +	2.8317e-1 (1.17e-2)
+/−/≈			9/7/4	13/1/6	10/3/7	7/5/8	14/2/4	6/7/7	10/7/3	10/7/3	

It is conspicuous, that the number of marked cells is immensely higher than in the prior experiments. This is, on the one hand, a sign for more frequent similar results, but also for more frequent good performing results. Only in three cases is the original algorithm part of the best performing algorithms in the pairwise comparison. The most often best performing algorithms are NCRXS+NSGA-II with 15 times and SRXD+NSGA-II with 12 times. Furthermore, it is interesting that the URXD+NSGA-II as well as URXS+NSGA-II belong multiple times to the best performing algorithms, although they both behave nearly random in the operator selection. The variants using the R2-Reward function and the NCR+NSGA-II algorithm make up the centre field in terms of frequency of belonging to the best performing algorithms. At the end of the line are SRXS+NSGA-II and the original algorithm.

The SBX+NSGA-II is outperformed on nearly any problem by at least one algorithm. On WFG1, it is approximately equal to NCRX+NSGA-II and SRXD+NSGA-II. On WFG3, all algorithms are approximately equal and on WFG8 only the NCRXS+NSGA-II can keep up with the SBX+NSGA-II. Regarding the corresponding HV results in Table A.4, the SBX+NSGA-II seems to perform a lot better, as it additionally belongs to the best performing algorithms on DTLZ3, DTLZ6 and WFG7. On DTLZ3, no sufficient HV results are calculable, and thus all algorithms are marked as equal. On DTLZ7, the values of the HV are seemingly very near to each other, and hence they are approximately equal except for SRXD+NSGA-II and URXD+NSGA-II that reached a sufficient significant better result in the pairwise comparison. The algorithm outperforming the original algorithm most often is the SRXD+NSGA-II with 14 times, followed by the NCRXS+NSGA-II with 13 times. Regarding the HV, the SRXD+NSGA-II is only second best against SBX+NSGA-II and NCRXS+NSGA-II performs most often better than SBX+NSGA-II. For the IGD results, the variants using R2XD, URXD and URXS are on the third place with 10 times each. NCRXD+NSGA-II and R2XS+NSGA-II make up the centre field again. SRXS+NSGA-II is on the last place as the only Hyper-Heuristic algorithm that is more often outperformed by SBX+NSGA-II as vice versa. It is also remarkable, that both variants of URX, the distribution variant of NCRX and the selection variant of SRX are seven times outperformed by SBX+NSGA-II, which is the highest frequency of all algorithms in this setting. On the rotated problems, RM1-4, is not any algorithm that performs worse than or equal to the SBX+NSGA-II algorithm.

Number of Decision Variables multiplied by four

Table 4.5 shows the median IGD of all NSGA-II variants with Hyper-Heuristic crossover operators on the 20 selected benchmark with all D multiplied by 4 after 10,000 FEs.

Table 4.5: Inverted Generational Distance of NSGA-II with different Hyper-Heuristic crossover operators on DTLZ, RM and WFG with a number of decision variables multiplied by 4.

Problem	*M*	*D*	NCRXD	NCRXS	R2XD	R2XS	SRXD	SRXS	URXD	URXS	SBX
DTLZ1	3	28	1.2050e+2 (2.55e+1) −	3.7672e+1 (1.29e+1) +	5.5126e+1 (3.55e+1) −	6.1571e+1 (5.13e+1) −	3.9246e+1 (1.63e+1) ≈	1.5025e+2 (5.10e+1) −	1.0456e+2 (2.08e+1) −	1.1396e+2 (2.01e+1) −	4.1689e+1 (1.11e+1)
DTLZ2	3	48	8.5599e-2 (8.12e-3) +	7.9526e-2 (5.47e-3) +	9.5740e-2 (1.91e-2) ≈	9.9104e-2 (2.96e-2) ≈	1.1532e-1 (2.88e-2) −	8.6645e-2 (1.30e-2) +	9.2745e-2 (1.09e-2) +	8.7561e-2 (5.92e-3) +	1.0340e-1 (1.02e-2)
DTLZ3	3	48	7.9849e+2 (1.25e+2) −	4.4765e+2 (1.40e+2) −	2.9244e+2 (2.88e+2) ≈	5.5474e+2 (1.98e+2) −	4.4071e+2 (1.30e+2) ≈	1.0507e+3 (2.74e+2) −	7.8607e+2 (1.03e+2) −	8.6484e+2 (7.99e+1) −	3.8932e+2 (9.09e+1)
DTLZ4	3	48	1.1122e-1 (2.99e-2) +	8.4983e-2 (1.22e-2) +	9.5014e-2 (1.49e-2) +	8.7813e-2 (2.21e-2) +	1.1410e-1 (3.69e-2) ≈	1.0429e-1 (2.38e-2) +	1.0531e-1 (1.99e-2) +	9.8769e-2 (2.26e-2) +	1.2286e-1 (4.34e-1)
DTLZ5	3	48	2.0929e-2 (5.24e-3) +	1.9011e-2 (7.19e-3) +	2.6802e-2 (8.50e-3) +	3.0024e-2 (1.99e-2) +	3.2488e-2 (1.31e-2) +	2.0807e-2 (8.70e-3) +	2.2549e-2 (8.34e-3) +	1.8740e-2 (4.44e-3) +	3.8987e-2 (1.16e-2)
DTLZ6	3	48	1.0762e-2 (9.74e-1) +	6.2667e-3 (1.26e-3) +	1.8607e+0 (4.19e+0) +	9.6668e-1 (2.01e+0) +	5.9681e-3 (8.12e-1) +	8.1625e+0 (6.39e+0) +	1.0063e+0 (1.00e+0) +	4.4455e+0 (4.20e+0) +	1.7075e+1 (2.07e+0)
DTLZ7	3	88	1.6081e+0 (5.84e-1) −	1.4459e+0 (4.00e-1) −	8.0357e-1 (3.99e-1) ≈	1.1139e+0 (7.24e-1) −	9.0483e-1 (2.08e-1) ≈	4.0269e+0 (2.02e+0) −	1.9848e+0 (3.62e-1) −	2.6768e+0 (6.40e-1) −	8.4442e-1 (1.85e-1)
RM1	2	120	1.6973e-1 (7.78e-3) +	1.8027e-1 (9.06e-3) +	1.8268e-1 (8.07e-3) +	1.7941e-1 (9.36e-3) +	1.8299e-1 (5.90e-3) +	1.7186e-1 (1.06e-2) +	1.7327e-1 (8.54e-3) +	1.7427e-1 (9.81e-3) +	3.2392e-1 (6.06e-2)
RM2	2	120	2.7839e-1 (1.59e-2) +	2.9241e-1 (8.49e-3) +	2.8870e-1 (1.22e-2) +	2.9289e-1 (1.52e-2) +	2.9097e-1 (1.14e-2) +	2.8671e-1 (1.04e-2) +	2.8521e-1 (1.50e-2) +	2.8460e-1 (1.02e-2) +	5.1121e-1 (1.62e-2)
RM3	2	40	2.5437e+0 (4.14e-1) ≈	2.4102e+0 (3.75e-1) +	2.3380e+0 (3.07e-1) +	2.6413e+0 (4.65e-1) ≈	2.1076e+0 (2.04e-1) +	2.6314e+0 (3.79e-1) ≈	2.5248e+0 (3.50e-1) +	2.5303e+0 (3.27e-1) +	2.6871e+0 (3.79e-1)
RM4	3	48	3.9606e-1 (7.79e-2) +	4.3664e-1 (7.89e-2) +	5.8442e-1 (1.45e-1) ≈	5.3974e-1 (1.03e-1) ≈	5.1534e-1 (1.25e-1) +	3.5003e-1 (1.86e-1) +	4.0663e-1 (6.21e-2) +	3.9017e-1 (1.15e-1) +	5.4905e-1 (7.04e-2)
WFG1	3	48	1.4907e+0 (4.28e-2) −	1.1623e+0 (8.29e-2) +	1.3029e+0 (1.01e-1) −	1.3824e+0 (7.94e-2) −	1.2133e+0 (7.62e-2) +	1.5220e+0 (1.16e-1) −	1.4321e+0 (8.32e-2) −	1.4104e+0 (9.98e-2) −	1.2488e+0 (7.84e-2)
WFG2	3	48	2.8276e-1 (2.54e-2) ≈	2.6356e-1 (1.62e-2) +	2.7682e-1 (3.36e-2) ≈	2.8851e-1 (2.98e-2) ≈	2.7290e-1 (2.63e-2) +	2.8346e-1 (3.32e-2) ≈	2.7911e-1 (1.54e-2) ≈	2.8350e-1 (2.14e-2) ≈	2.9229e-1 (3.70e-2)
WFG3	3	48	3.1003e-1 (3.52e-2) +	2.7838e-1 (2.45e-2) +	3.2442e-1 (3.98e-2) ≈	3.0536e-1 (4.27e-2) +	3.0443e-1 (4.89e-2) +	2.8848e-1 (4.36e-2) +	2.8593e-1 (3.17e-2) +	3.0793e-1 (4.33e-2) +	3.2445e-1 (2.96e-2)
WFG4	3	48	3.1790e-1 (1.20e-2) ≈	2.8513e-1 (1.29e-2) +	3.1663e-1 (1.71e-2) ≈	3.2011e-1 (1.21e-2) ≈	3.1999e-1 (1.39e-2) ≈	3.2444e-1 (2.87e-2) ≈	3.1646e-1 (1.45e-2) ≈	3.1869e-1 (1.30e-2) ≈	3.1997e-1 (1.29e-2)
WFG5	3	48	3.7776e-1 (2.88e-2) −	2.7827e-1 (9.68e-3) +	3.0026e-1 (2.15e-2) +	3.3107e-1 (5.08e-2) ≈	2.8813e-1 (2.31e-2) +	3.7140e-1 (5.33e-2) −	3.2527e-1 (2.95e-2) +	3.4196e-1 (2.98e-2) ≈	3.4070e-1 (2.12e-2)
WFG6	3	48	3.5357e-1 (2.30e-2) +	3.0952e-1 (1.37e-2) +	3.4706e-1 (6.38e-2) +	3.4927e-1 (4.03e-2) +	3.6689e-1 (3.18e-2) +	3.7144e-1 (4.38e-2) ≈	3.4791e-1 (2.32e-2) +	3.5550e-1 (1.70e-2) +	3.7961e-1 (2.32e-2)
WFG7	3	48	3.4879e-1 (1.68e-2) +	3.4115e-1 (4.07e-2) +	3.3373e-1 (2.19e-2) +	3.4067e-1 (1.80e-2) +	3.7327e-1 (3.30e-2) +	3.4578e-1 (2.15e-2) +	3.4082e-1 (2.52e-2) +	3.3219e-1 (3.33e-2) +	4.3422e-1 (8.50e-2)
WFG8	3	48	4.3019e-1 (1.52e-2) −	3.9996e-1 (1.64e-2) −	4.3361e-1 (2.41e-2) −	4.3058e-1 (2.12e-2) −	4.4796e-1 (2.35e-2) −	4.3219e-1 (2.30e-2) −	4.2884e-1 (1.93e-2) −	4.3304e-1 (1.67e-2) −	3.8080e-1 (1.28e-2)
WFG9	3	48	3.1170e-1 (1.60e-2) +	3.0821e-1 (2.03e-2) +	3.1676e-1 (2.64e-2) +	3.1994e-1 (2.61e-2) +	3.1753e-1 (2.92e-2) +	3.0920e-1 (2.56e-2) +	3.1171e-1 (1.93e-2) +	3.1293e-1 (2.07e-2) +	4.0673e-1 (3.55e-2)
+/−/≈			11/6/3	17/3/0	10/3/7	9/5/6	13/2/5	10/6/4	13/5/2	12/5/3	

In the pairwise comparison underlines the dominance of the NCRXS+NSGA-II in this setting. It belongs to best performing algorithms on 13 problems. This is confirmed by the corresponding HV results in Table A.5, where it belonged to the best performing algorithms on two additional problems. It is necessary to note, that 3 of those problems, namely DTLZ1, DTLZ3 and RM3, are not sufficiently solved by any algorithm in terms of HV so that they are all marked as equal. Both variants using the R2-Reward function are on the same level as SBX+NSGA-II regarding the pairwise comparison. But also the NCRXD+NSGA-II and both URX variants belong only four times to the best performing algorithms. On this more complex setting, both SRX variants belong six times to the best performing algorithms. In comparison to the default setting, the number of algorithms belonging to the best algorithm on each problem is decreased.

In comparison to the SBX+NSGA-II, most algorithms achieve significantly better results. Especially on WFG1, which is the best problem of SBX+NSGA-II in the single operator experiment, the original NSGA-II performs very successfully, only topped by the NCRXS+NSGA-II. The results are similar for DTLZ1, DTLZ3, DTLZ7 and WFG8, noting that DTLZ1 and DTLZ3 are not sufficiently solved by any algorithm regarding the HV. Overall, the number of times the SBX+NSGA-II is surpassed in terms of the median IGD is noteworthy high. The NCRXS+NSGA-II performed 17 times significantly better than the original NSGA-II. The SRXD+NSGA-II as well as the URXD+NSGA-II reaches a better result 13 times. URXS+NSGA-II outperformed the SBX+NSGA-II 12 times, NCRXD+NSGA-II 11 times and SRXS+NSGA-II and R2XD+NSGA-II 10 times. Those numbers are different to those from the HV measurement. This is caused by a higher number of approximately equal results, that shrinks the number of times, where another algorithm is significantly better or worse than the original NSGA-II.

Number of Decision Variables multiplied by six

Table 4.6 shows the median IGD of all NSGA-II variants with Hyper-Heuristic crossover operators on the 20 selected benchmark with all D multiplied by 6 after 10,000 FEs.

Table 4.6: Inverted Generational Distance of NSGA-II with different Hyper-Heuristic crossover operators on DTLZ, RM and WFG with a number of decision variables multiplied by 6.

Problem	*M*	*D*	NCRXD	NCRXS	R2XD	R2XS	SRXD	SRXS	URXD	URXS	SBX
DTLZ1	3	42	2.3607e+2 (3.88e+1) −	9.9898e+1 (4.26e+1) +	1.4250e+2 (8.20e+1) ≈	1.6650e+2 (8.56e+1) −	1.1778e+2 (2.46e+1) ≈	3.2310e+2 (1.03e+2) −	2.2284e+2 (2.29e+1) −	2.4585e+2 (3.22e+1) −	1.2793e+2 (2.67e+1)
DTLZ2	3	72	1.3159e-1 (2.90e-2) +	1.1146e-1 (2.09e-2) +	1.8131e-1 (4.57e-2) ≈	1.7459e-1 (6.88e-2) ≈	1.7790e-1 (5.03e-2) ≈	1.2899e-1 (2.56e-2) +	1.4205e-1 (3.43e-2) +	1.2608e-1 (1.84e-2) +	1.8915e-1 (2.72e-2)
DTLZ3	3	72	1.2439e+3 (2.93e+2) −	9.9067e+2 (1.99e+2) ≈	4.9084e+2 (3.49e+2) +	1.1165e+3 (3.62e+2) −	8.5450e+2 (2.02e+2) ≈	1.7709e+3 (2.36e+2) −	1.2996e+3 (1.81e+2) −	1.5139e+3 (1.39e+2) −	9.0732e+2 (1.35e+2)
DTLZ4	3	72	2.0310e-1 (5.21e-2) +	1.4235e-1 (3.25e-2) +	1.5817e-1 (3.62e-2) +	1.2799e-1 (4.17e-2) +	1.8270e-1 (3.76e-2) +	1.7780e-1 (5.31e-2) +	1.9413e-1 (4.32e-2) +	1.6457e-1 (3.88e-2) +	2.5734e-1 (3.41e-1)
DTLZ5	3	72	7.0813e-2 (2.28e-2) +	5.4675e-2 (2.34e-2) +	8.1720e-2 (4.54e-2) +	8.5981e-2 (3.55e-2) +	8.7901e-2 (3.60e-2) +	5.4779e-2 (3.13e-2) +	7.2331e-2 (3.00e-2) +	6.2985e-2 (1.42e-2) +	1.3972e-1 (2.65e-2)
DTLZ6	3	72	7.5898e+0 (6.30e+0) +	9.9076e-1 (9.97e-1) +	1.0312e+1 (1.19e+1) +	4.4037e+0 (7.25e+0) +	9.9727e-1 (1.96e+0) +	1.7420e+1 (1.12e+1) +	7.4490e+0 (2.74e+0) +	1.3654e+1 (5.15e+0) +	3.6812e+1 (1.88e+0)
DTLZ7	3	132	2.9377e+0 (7.46e-1) −	2.2116e+0 (5.70e-1) −	1.4069e+0 (7.86e-1) −	2.1239e+0 (1.05e+0) −	1.5453e+0 (4.64e-1) −	4.6187e+0 (1.55e+0) −	3.1874e+0 (6.04e-1) −	4.1046e+0 (5.89e-1) −	1.1564e+0 (1.92e-1)
RM1	2	180	1.7818e-1 (6.12e-3) +	1.9099e-1 (8.74e-3) +	1.8618e-1 (4.47e-3) +	1.8987e-1 (8.31e-3) +	1.8936e-1 (7.33e-3) +	1.8257e-1 (7.84e-3) +	1.8046e-1 (5.28e-3) +	1.8534e-1 (6.19e-3) +	3.6452e-1 (5.11e-2)
RM2	2	180	2.8586e-1 (1.06e-2) +	2.9891e-1 (1.08e-2) +	2.9258e-1 (9.92e-3) +	2.9908e-1 (8.50e-3) +	2.9098e-1 (8.72e-3) +	2.9754e-1 (1.46e-2) +	2.9391e-1 (1.01e-2) +	2.9410e-1 (8.13e-3) +	5.2538e-1 (2.54e-2)
RM3	2	60	3.2629e+0 (3.04e-1) +	3.0637e+0 (2.40e-1) +	3.0655e+0 (2.22e-1) +	3.1969e+0 (3.28e-1) +	2.8171e+0 (2.93e-1) +	3.2805e+0 (3.78e-1) +	3.2457e+0 (3.87e-1) +	3.2125e+0 (2.79e-1) +	3.8126e+0 (3.05e-1)
RM4	3	72	6.1934e-1 (1.09e-1) +	6.5004e-1 (1.28e-1) +	7.9123e-1 (2.12e-1) +	6.9029e-1 (1.65e-1) +	7.1880e-1 (1.09e-1) +	5.9647e-1 (1.21e-1) +	6.2085e-1 (1.26e-1) +	6.1614e-1 (1.33e-1) +	1.0348e+0 (1.59e-1)
WFG1	3	72	1.5496e+0 (5.64e-2) −	1.2946e+0 (6.49e-2) +	1.4168e+0 (1.05e-1) −	1.4649e+0 (8.25e-2) −	1.3482e+0 (6.77e-2) ≈	1.5598e+0 (7.27e-2) −	1.4942e+0 (4.39e-2) −	1.4984e+0 (5.35e-2) −	1.3609e+0 (4.40e-2)
WFG2	3	72	3.1489e-1 (2.69e-2) +	2.8439e-1 (3.01e-2) +	3.0946e-1 (4.22e-2) +	3.1940e-1 (4.12e-2) +	2.9782e-1 (2.44e-2) +	3.1340e-1 (2.51e-2) +	3.1207e-1 (2.51e-2) +	3.1195e-1 (2.37e-2) +	3.4039e-1 (3.54e-2)
WFG3	3	72	3.5704e-1 (2.93e-2) +	3.1165e-1 (2.05e-2) +	3.5678e-1 (4.58e-2) +	3.6745e-1 (3.83e-2) +	3.5293e-1 (2.44e-2) +	3.4139e-1 (2.80e-2) +	3.4746e-1 (3.78e-2) +	3.3750e-1 (2.71e-2) +	4.0978e-1 (3.57e-2)
WFG4	3	72	3.2416e-1 (1.40e-2) +	2.9954e-1 (2.03e-2) +	3.2452e-1 (1.36e-2) +	3.3210e-1 (1.70e-2) +	3.2746e-1 (1.03e-2) +	3.3091e-1 (1.69e-2) +	3.2082e-1 (1.18e-2) +	3.2361e-1 (1.63e-2) +	3.3662e-1 (1.41e-2)
WFG5	3	72	3.8492e-1 (3.33e-2) ≈	2.8527e-1 (1.41e-2) +	3.1112e-1 (3.46e-2) +	3.5034e-1 (3.11e-2) +	3.0142e-1 (1.74e-2) +	3.9913e-1 (8.36e-2) ≈	3.3138e-1 (2.74e-2) +	3.5704e-1 (3.21e-2) +	3.8375e-1 (1.54e-2)
WFG6	3	72	3.7449e-1 (2.00e-2) +	3.2448e-1 (1.94e-2) +	3.7017e-1 (4.02e-2) +	3.7139e-1 (3.11e-2) +	3.9612e-1 (2.88e-2) +	4.0330e-1 (4.31e-2) +	3.6991e-1 (2.21e-2) +	3.7434e-1 (2.81e-2) +	4.3717e-1 (1.95e-2)
WFG7	3	72	3.7199e-1 (2.53e-2) +	3.7682e-1 (5.80e-2) +	3.8541e-1 (3.31e-2) +	3.7089e-1 (3.51e-2) +	3.9760e-1 (3.60e-2) +	3.7440e-1 (2.74e-2) +	3.7176e-1 (1.88e-2) +	3.6999e-1 (2.46e-2) +	4.9716e-1 (2.41e-2)
WFG8	3	72	4.3737e-1 (2.70e-2) −	4.1054e-1 (1.21e-2) −	4.4184e-1 (2.25e-2) −	4.4099e-1 (2.56e-2) −	4.4620e-1 (2.09e-2) −	4.4151e-1 (1.92e-2) −	4.4620e-1 (1.46e-2) −	4.4563e-1 (1.69e-2) −	4.0371e-1 (1.33e-2)
WFG9	3	72	3.1330e-1 (1.76e-2) +	3.1167e-1 (1.71e-2) +	3.2179e-1 (1.66e-2) +	3.2016e-1 (1.52e-2) +	3.2361e-1 (2.27e-2) +	3.3003e-1 (2.95e-2) +	3.1898e-1 (1.75e-2) +	3.0998e-1 (1.83e-2) +	4.7615e-1 (6.44e-2)
+/−/≈			14/5/1	17/2/1	15/3/2	14/5/1	14/2/4	14/5/1	15/5/0	15/5/0	

In the pairwise comparison, it is visible that the trend of a decreasing number of grey cells continues. It underlines the idea of getting more diverse results by increasing the complexity of the problems. The same is noticeable in the corresponding HV Table A.6. Again, the NCRXS+NSGA-II is the most dominating algorithm with 13 out of 20 cases, where it belongs to the best performing algorithms. The other algorithms are similarly often part of the best performing algorithms, with three to five times. An exception is R2XD+NSGA-II with only one mark, which is one less than the original algorithm.

In the comparison to SBX+NSGA-II, the NCRXD+NSGA-II outperforms SBX+NSGA-II 17 times, the algorithms using R2XD, URXD and URXS 15 times and the algorithms using NCRXD, R2XS, SRXD and SRXS 14 times. Therefore, the differences between the quality of Hyper-Heuristics MOEAs extends with increasing D.

Evaluation

At the beginning of this section, we stated different expectations on the results. The first expectation, the superiority on most problems over the original NSGA-II is already mentioned in the description of the data. It is visible, that the Hyper-Heuristic selection, improves the result quality on most problems in all three settings, although the improvements are emphasized by the increase of D. This is reasoned by the additional effort of the Hyper-Heuristics in the exploring of the best crossover operator. When all operators perform nearly equally well, the selection of the best operator does not compensate the computational costs, which are limited by the number of FEs. This happens in the default configuration on the problems DTLZ5 and WFG9. Although the Hyper-Heuristics surpass the SBX+NSGA-II, the results are very close to each other. Another case that shrinks the benefits from using Hyper-Heuristics, is when one single crossover operator is superior to the other operators. Especially HHDs have difficulties in those cases, because they are forced to use at least a part of the generation for operators, that do not deliver good results. This happens on WFG1 with all configurations, as SBX+NSGA-II outperforms all other operators. Only NCRXS+NSGA-II and SRXS+NSGA-II can keep up to the SBX+NSGA-II, especially in the higher complexity settings because they can use good exploration of UX or CMAX to compensate the weaknesses of SBX with higher D. Those cases demonstrate the importance of a preselection of crossover operators, as the Hyper-Heuristics can only improve the single operator usage if there is a compensation between the selected operators. Besides those two cases, the Hyper-Heuristics dominate the original NSGA-II mostly.

We also expected to see differences in quality between the different reward functions. As there are multiple well-working operators on most problems, all Hyper-Heuristics that consider those operators, are performing better than the single SBX operator. Even the URXD and URXS, that act mostly random, can outperform the single crossover operators, as visible here in numerous instances on all configurations. The best performing Hyper-Heuristic is the NCRXS. This Hyper-Heuristic implements a concept, which we already know, works well. The non-dominated sorting argument combined with crowding distance is also part of the environment selection of the NSGA-II. We save in every generation, which operators are responsible for those offspring that are rejected for the next generation, and punish those operators. The

Survival Reward Hyper-Heuristics, SRXD and SRXS, use a similar reward, as they calculate the ratio between produced and survived individuals. When we already implement NDS and CD as an environmental selection, the rating of the number of survivors and rating the rank after the NDS should lead to a similar selection behaviour. Thus, the Hyper-Heuristics using Survival Rewards are the second best. The use of the R2-Reward seems to cause the least successful results.

Our last expectation, that the HHSs would overall perform better than HHDs, was partially confirmed. The best Hyper-Heuristic operator is visibly the NCRXS, but the SRXD also performed well and the R2XD performed better than the R2XS.

These results show, that the usage of a Hyper-Heuristic as a selector for crossover operators, improve the quality of the results. We also showed, that the impact of the Reward function and the selection heuristic is present, although it is not yet completely reasoned, what the impact is and how these components affect the selection behaviour. It is furthermore not yet shown if Hyper-Heuristics perform better than all single operators. As noted in the previous section, the CMAX or the UX operators often performed better, than the SBX operator. In this analysation, we already touched on the topic of the influence of the operator selection. In the next section, we use visualizations to show the operator behaviour over FEs. We categorize the single operator behaviour to show whether the Hyper-Heuristics behave accordingly.

As in both of the previous analyzations, the medium complexity configurations delivered the most interesting results, the following analyzations continue with exclusively this configuration.

4.2.3 Comparison of Single Crossover Operators and Hyper-Heuristics of FEs

In this section, the behaviour of the Hyper-Heuristics over FEs are analysed. This is done by comparing first the IGD trends of single crossover operators with those of the Hyper-Heuristics and, afterwards, interpreting the selection and distribution behaviour of the Hyper-Heuristics.

We selected four benchmark problems, each to represent one category of behaviour of the single crossover operators so that we can compare these to the processes of the Hyper-Heuristics. The categories are:

1. crossover operators behave alike
2. one or two crossover operators perform better than the other
3. one or two crossover operators perform worse than the other
4. crossover operators cover a wider range of quality levels

We sort the benchmark problems to those categories by eye and choose the most extreme problems as examples. In Appendix A.1.2 are all benchmark problems displayed. By analysing those categories, we achieve the fourth subgoal, to identify the influence of the components of Hyper-Heuristics by reasoning their behaviour.

With this more detailed comparison, we confirm again our results from the prior section and extend the insights on the quality of the HHSs and the HHDs. We already know that the differences in the behaviour of the single crossover operators have a big influence on the Hyper-Heuristics, and thus the categories were chosen to examine these cases in more detail.

For the comparison, we use two line graphs, the first one representing the single crossover operators and the second one representing the Hyper-Heuristics and the best single crossover operators. Additionally, we visualize interim and the end IGD values, including the IQR, of all algorithms in a bar chart to give a better overview about the pairwise comparisons. For a more in-depth analysis of the selection behaviour of the Hyper-Heuristics, we selected the best HHS and the best HHD, according to the number of times they surpassed the original NSGA-II. Thus, we decided to show the behaviour of NCRXS and SRXD. We use a bar chart to visualize the change in selection probability, for NCRXS, or distribution, for SRXD, and the cumulative number of offspring produced by each single crossover operator in a line graph. The exemplary data, shown in the graphs of this section, are recorded in those runs, that belong to the median IGD results in Table 4.2 and 4.5. As the prior evaluations already concluded, the medium complexity configurations delivered the most interesting results, and thus we continue using only the problems with D multiplied by four.

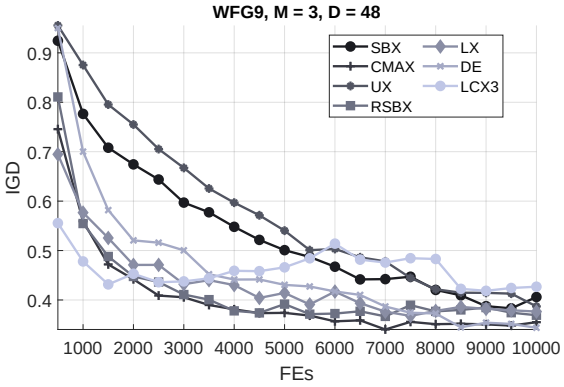
In the following, we will first explain what we are expecting to see in each category, we describe the example and then evaluate the graphs and discuss whether the expectations were met and what concludes from those.

1.) Category: crossover operators behave alike

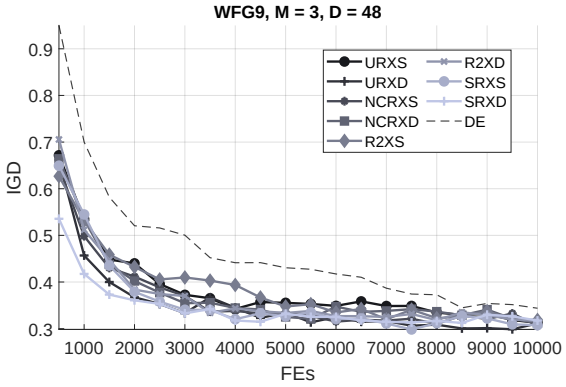
This category gathers all benchmark problems, where the single crossover operators have similar IGD curves, that ends up near to the others. The benchmark problems belonging to this category are DTLZ2, DTLZ5, RM3, RM4, and WFG9. As the exemplary problem, we chose WFG9. When all single crossover operators act similar, we expect that not all variants of the Hyper-Heuristic operators would outperform the best single crossover operator. Due to the similarity, the HHSs would select the operator nearly randomly, like the URXS. As we have seen in the prior section, it would still perform better than the original NSGA-II in most cases, but not in all. The HHD would distribute the generation approximately equally to the operators, like the URXD. When all operators are equally good, this would not have any negative effects. Of course, they are not completely alike, but with only small differences HHSs have difficulties exploiting the best operator.

We first describe the quality comparison over FEs and afterwards the behaviour of Hyper-Heuristics. At the end of this paragraph, we conclude our findings.

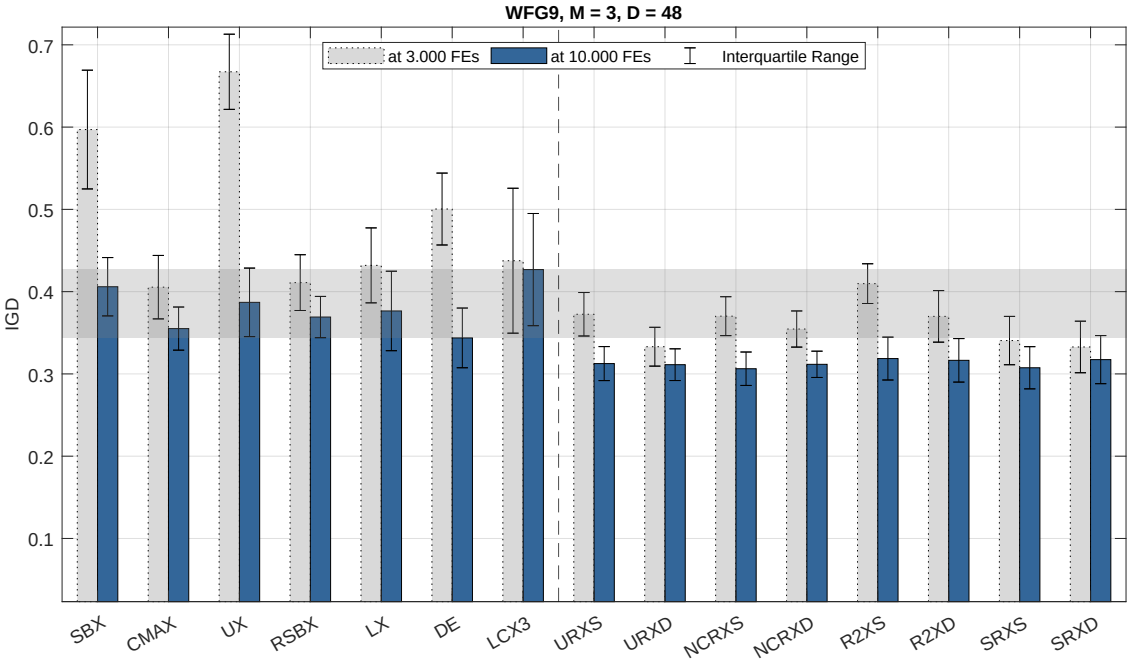
The first graph in Figure 4.1 shows the IGD values of the single crossover operators on the y-axis and the FEs on the x-axis. Each graph shows the quality trend of one single crossover operator. As we can see, the SBX and the UX both converge slower than the others, as their curves are flatter than the others. Interesting is the behaviour of the LCX3, as it starts already with a well IGD value, but it varies a lot and even increases again after about 5000 FEs. It ends up with the highest



(a) IGD of NSGA-II with single crossover operators.



(b) IGD of NSGA-II with hyper-heuristic crossover operators compared to DE+NSGA-II

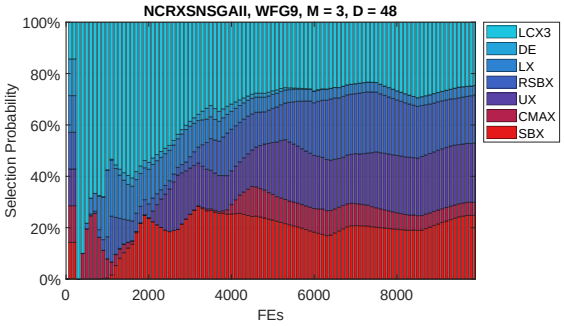


(c) IGD after 3000 FEs and after 10.000 FEs.

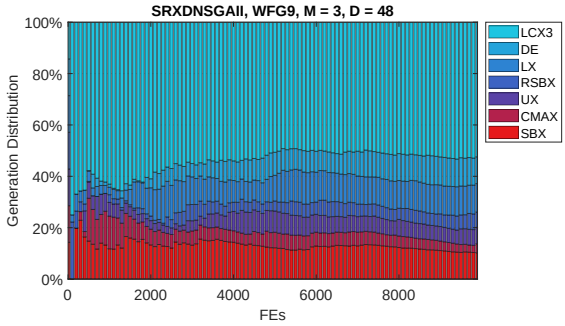
Figure 4.1: IGD measured on WFG9. The number of the decision variables is multiplied by 4.

end IGD. The remaining operators behave similar, with a steep descendant at the beginning and a flatter end. The DE operator, starting between the other operators, is continuously decreasing, even when the others are already flattening. Thus, its end IGD is the best in this comparison.

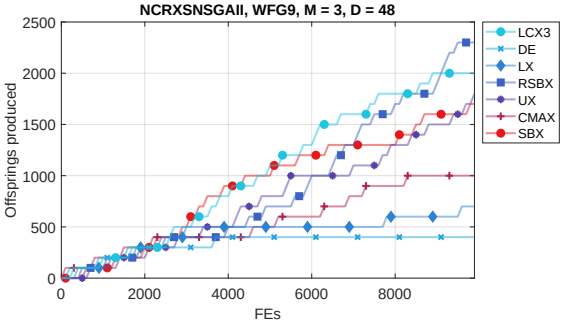
The second graph in Figure 4.1 shows the IGD values of the Hyper-Heuristic crossover operators on the y-axis and the FEs on the x-axis. Additionally, the DE operator is chosen for the comparison and is symbolized with a dashed line. It is remarkable that all Hyper-Heuristics behave a like, starting already with a low IGD, having a steep descendant, and ending with a flatter part after about 4.000 FEs. For the whole process, all Hyper-Heuristics achieve an IGD below the value of the DE operator. A visible difference is given by the graph for the R2XS, that has a small variety at about 3.000 FEs. This algorithm also ends up with the highest IGD of the Hyper-Heuristic operators.



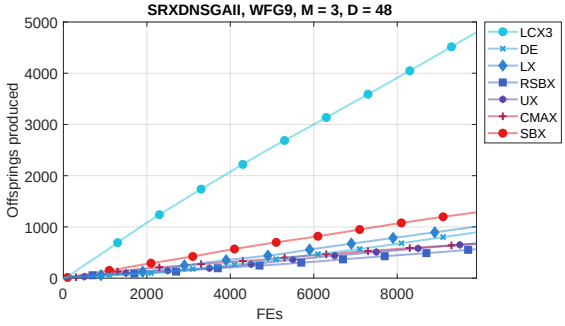
(a) Selection probabilities of crossover operators by NCRXS over the FEs.



(b) Distribution of the generations to the crossover operators by SRXD over the FEs.



(c) Cumulative number of individuals produced by each crossover operator inn NCRXS.



(d) Cumulative number of individuals produced by each crossover operator in SRXD.

Figure 4.2: Selection and distributions behaviour of NCRXS and SRXD on DTLZ7 with the number of the decision variables is multiplied by 4.

The third graph in Figure 4.1 is a bar chart with, again, IGD values on the y-axis. The graph shows the interim results as lighter bars after 3.000 FEs and the end results as darker bars after 10.000 FEs. Both bars demonstrate the values of the median run and include an error bar symbolizing the IQR. On the left side of the dashed line in the centre are the results of the single crossover operators, on the right side are the results of the Hyper-Heuristics crossover operators. The darker horizontal bar highlights the interval of the worst median IGD result of the single crossover operators to the best. This interval is small in this case because single crossover operators end up near to each other. All Hyper-Heuristics are visibly below this interval. Partially, even the interim values are already below the best single crossover operator after 10.000 FEs. The variety of R2XS is also visible in this bar chart, as the interim results is comparably high to the other values.

In Figure 4.2 the selection behaviour of NCRXS, left column, and SRXD, right column, are visualized. In the upper row, the change of selection probability respectively the generation distribution of each single crossover operator is illustrated. The y-axis is here the percentage of the operator, the x-axis are the FEs. We choose a stacked bar chart, that always adds up to 100%, to emphasize the ratios between the crossover operators. In the lower row, we use line graphs to show, how many new individuals the operators produced cumulatively over FEs. Accordingly, the y-axis is the number of offspring and x-axis the FEs.

For the HHSs the exploration phase of the crossover operators are visible. For the first generations, the selection probabilities are equal so that each operator is selected

five times. The score is afterwards updated after each generation. This is shown in Figure 4.2 (a). After the exploration with NCRXS on WFG9, the LCX3 operator has the highest selection probability. It begins to compete with the CMAX for the first 1.000 FEs. The SBX increased its probability after 2.000 FEs and stays with few variations at about 20%. After 4.000 FEs, the RSBX and the UX both raised their probability to 10%. The CMAX and the LX converge to probability of about 5%, whereas the DE operators vanishes completely after 6.000 FEs. This is also recognizable in Figure 4.2 (c). The LCX3 generate whole generations frequently, as long as its selection probability is high. It is in the last FEs passed by the RSBX. Moreover, the SBX and the UX operators are often selected. The vanishing of the selection probability of DE is visible, as its production curve stagnate after 4.000 FEs. The LX and the CMAX operators, whose selection probabilities are mostly low, are also only responsible for few generations.

The distribution data of SRXD is less varying. It also starts to value the LCX3 highly, with a portion of generation over 50%. After many variations in the first 4.000 FEs between the other crossover operators, the curves stagnate at a distribution of about 50% LCX3, 12% DE, 12% LX, 12% SBX, 5% RSBX, 5% UX and 4% CMAX. This fast stagnating distribution is visible in the production plot in Figure 4.2 (d), as the line graphs have linear forms. The LCX3 operator with the highest portion produces the highest number of offspring. The production graphs of the other operators are similar, as their generation portions are similar too.

Both Hyper-Heuristics valued the LCX3 operator highly in the beginning. Whereas the NCRXS brought more varying operators into the game, the SRXD converges quickly to one distribution. The distribution uses all operators, although the operator receives the majority of the population as parents. The selection, on the other hand, does not focus on one but four operators and selects alternately between those.

Comparing those results again to the single crossover operator behaviour in Figure 4.1 (a), it becomes clear, why the LCX3 is preferred in that rate by both algorithms. The start values are by far the best, although it is passed already after 2.500 FEs by CMAX, RSBX and LX. The NCRXS adapts this and shrinks the probability of the LCX3, and compensates this by raising the probability of UX, LX and SBX. Regarding the Hyper-Heuristics curve in Figure 4.1 (b) the NCRXS reaches at 3.000 FEs an IGD value of below 0.4. The single crossover operators reach this point at about 7.000 FEs. At this point of the process, the UX and SBX do also deliver good solutions, which explains, why the NCRXS rated the UX and SBX better than the CMAX and RSBX. That the probability of the LCX3 does not shrink more, is either an indicator for the quality improvements of the LCX3 with other circumstances or the learning rate of NCRXS is not fast enough. We cannot exclude, that operators that do not perform well when used solely would perform well when combined with another operator that may compensate its weaknesses. This idea is indicated, by the NCRXS as well as the SRXD, as both perform best in the pairwise comparison and none of them utilizes the best operator in this setting, the DE operator, and even prefer the worst operator for this setting, the LCX3 operator, according to the pairwise comparison in Table 4.2. Even the progress graph in Figure 4.1 (a) confirms, that the LCX3 performs bad over all FEs, except the very beginning, and the DE

performs best over nearly all FEs. Nevertheless, both Hyper-Heuristics surpass all single crossover operator, with usage focused on the seemingly worst of them.

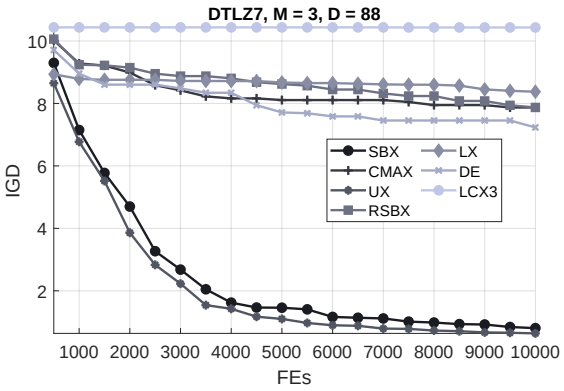
At the beginning of this paragraph, we phrased our expectations on the outcome of this category. We expected, that not all variants of the Hyper-Heuristic operators would outperform the best single crossover operator. We assumed, that the selection probabilities and the distributions would be nearly equal and cause therefore problems for the exploitation of the best option. None of these expectations are met. We assumed, that we could directly conclude from the behaviour of the single crossover operators, to the selection behaviour of the Hyper-Heuristics. This is not true because of two reasons: First, the combination of operators changes their ability to find good solutions dramatically. The operators can compensate their weaknesses so that the quality can be improved by any combination of operators. We anticipated, that the selections and the distributions are more difficult to navigate, when all operators behave seemingly equal. This brings up the second reason, why our expectation is not met. There is no overall quality measurement for solutions on MOPs. The IGD is a well-known metric, that gives an overview on the quality, but this does not necessarily correlate with our online quality measurements in the Hyper-Heuristics Reward functions. Moreover, the Rewards depend on the quality improvement, thus instead of regarding the IGD values, the steepest descends are maybe more interesting. Furthermore, the form of the population depends on the crossover operator that created the individuals. When the LCX3 performs bad on structures it created itself, it might perform better on a structure produced by the SBX or the DE operator. Thus, interesting is not only what is selected mostly, but also when and in which order. Those properties are nearly impossible to predict, as they only come up during the solving process. Furthermore, the number of possibilities of combining the crossover operators differently for every generation is extremely high. This underlines, that Hyper-Heuristics can decide on components of an algorithm with online learned knowledge we would never have before solving a problem.

2.) Category: one or two crossover operators perform better than the other

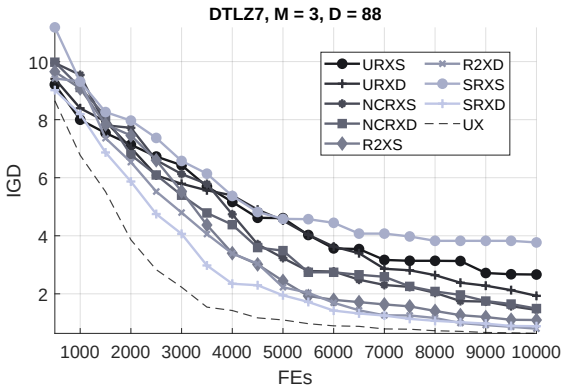
This category gathers all benchmark problems, where the single crossover operators have mostly similar IGD curves, except one or two that perform exceedingly better. The benchmark problems belonging to this category are DTLZ4 and DTLZ7. As the exemplary problem, we chose DTLZ7. With at least one operator, that performs clearly better than the others, the Hyper-Heuristic operators should not have any problems finding this operator and exploit it. Thus, we expect, that the Hyper-Heuristic operators perform mostly better than the single crossover operators. Furthermore, we expect that HHSs perform better than HHDs because the selection is better suited to exploit one good operator, since the HHDs are forced to use always all operators.

We first describe the quality comparison over FEs and afterwards the behaviour of Hyper-Heuristics. At the end of this paragraph, we conclude our findings.

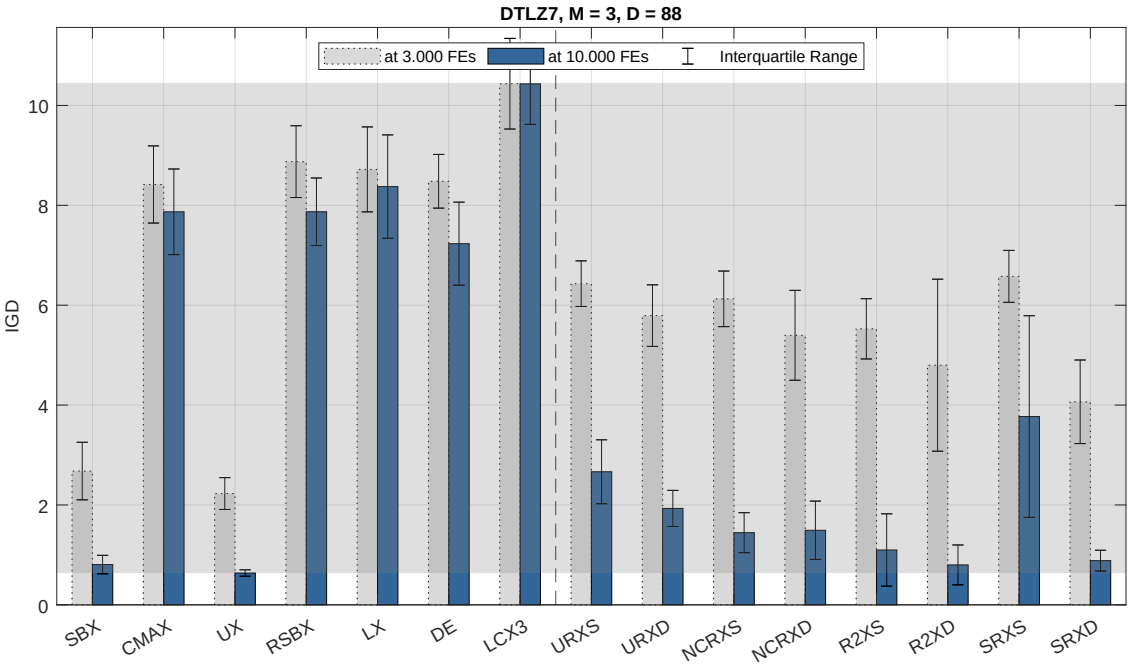
The first graph in Figure 4.3 shows the IGD values of the single crossover operators on the y-axis and the FEs on the x-axis. Each graph shows the quality trend of one single crossover operator. We selected DTLZ7 because two crossover operators,



(a) IGD of NSGA-II with single crossover operators.



(b) IGD of NSGA-II with hyper-heuristic crossover operators compared to UX+NSGA-II

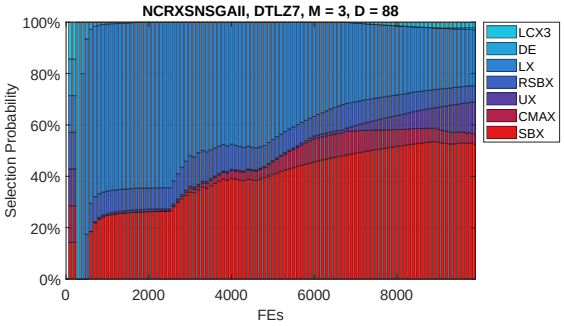


(c) IGD after 3000 FEs and after 10.000 FEs.

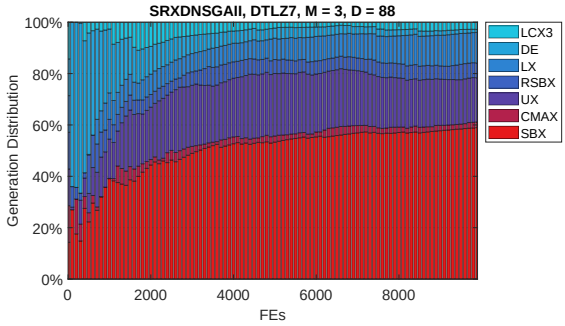
Figure 4.3: IGD measured on DTLZ7. The number of the decision variables is multiplied by 4.

SBX and UX, behave outstanding. Both of these operators have a steep descendant and flatten on a very low IGD value compared to the other operators' IGD values after roughly 5.000 FEs. The LCX3 operator, on the other hand, is constantly on an extremely high IGD value. The remaining operators are descending slowly, but they stay on a high value. The best performing operator is the UX.

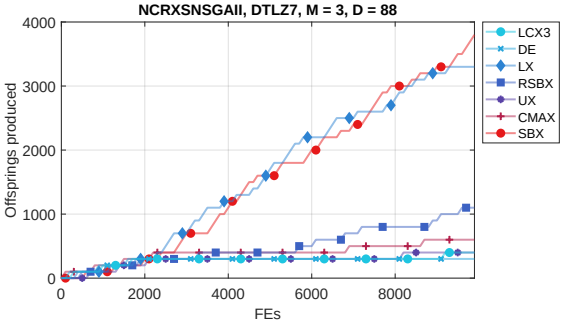
The second graph in Figure 4.3 shows the IGD values of the Hyper-Heuristic crossover operators on the y-axis and the FEs on the x-axis. Additionally, the UX operator is chosen for the comparison and is symbolized with a dashed line. For this benchmark, no Hyper-Heuristic operator performs better than the UX operator. All of them start on a very high IGD value above 10 and descend gradually until they flatten between 5.000 and 6.000 FEs. Although they all behave similarly, they end up wider distributed. The SRXS stagnates at an IGD of about 4, whereas the SRXD and R2XD approximate the UX curve.



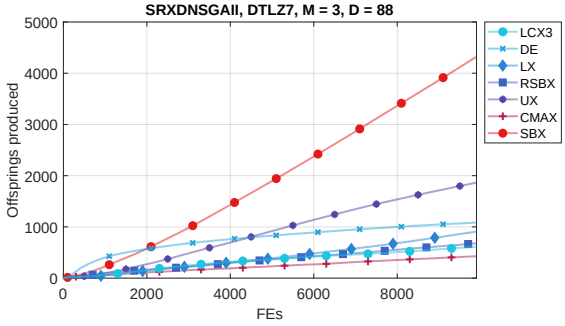
(a) Selection probabilities of crossover operators by NCRXS over the FEs.



(b) Distribution of the generations to the crossover operators by SRXD over the FEs.



(c) Cumulative number of individuals produced by each crossover operator in NCRXS.



(d) Cumulative number of individuals produced by each crossover operator in SRXD.

Figure 4.4: Selection and distributions behaviour of NCRXS and SRXD on DTLZ7 with the number of the decision variables is multiplied by 4.

The third graph in Figure 4.3 is a bar chart with, again, IGD values on the y-axis. The graph shows the interim results as lighter bars after 3.000 FEs and the end results as darker bars after 10.000 FEs. Both bars demonstrate the values of the median run and include an error bar symbolizing the IQR. On the left side of the dashed line in the centre are the results of the single crossover operators, on the right side are the results of the Hyper-Heuristics crossover operators. The darker horizontal bar highlights the interval of the worst median IGD result of the single crossover operators to the best. The bar chart emphasis, again, that no Hyper-Heuristic can keep up with the single UX crossover operator. It is interesting, that in this case the URX variants both delivered bad results. Furthermore, the SRXD delivers the worst result, whereas its selection variant, SRXS, delivers one of the best results on this setup.

In Figure 4.4 the selection behaviour of NCRXS, left column, and SRXD, right column, are visualized. In the upper row, the change of selection probability respectively the generation distribution of each single crossover operator is illustrated. The y-axis is here the percentage of the operator, the x-axis are the FEs. We choose a stacked bar chart, that always adds up to 100%, to emphasize the ratios between the crossover operators. In the lower row, we use line graphs to show, how many new individuals the operators produced cumulatively over FEs. Accordingly, the y-axis is the number of offspring and x-axis the FEs.

In Figure 4.4 (a) it is visible that the NCRXS focused the LX operator in the beginning. At about 500 FEs the selection probability for LX is about 70%, for

RSBX it is 10%, and for the SBX it is about 20%. The importance of the LX for the NCRXS decreases after 4.500 FEs as the selection probability of the SBX rises to 40% and the CMAX more attention draws. At 8.000 FEs, the selection probability of the CMAX operator shrinks again and the probability of the UX rises. The SBX operator reaches a selection probability of about 50%, and LX operators' probability decreases to 20%. The corresponding production curve in Figure 4.4 (c) confirms, that NCRXS mainly utilizes the SBX and the LX operator. After 6.000 FEs, the RSBX is also selected a few times, but the vast majority of all solutions are produced by the LX and the SBX operators.

In Figure 4.4 (b) it is visible, that the SRXD estimated the quality of the SBX operator similarly. In the beginning, about 70% of the generation is distributed to the DE operator, the rest to SBX and RSBX. Over the FEs, the importance of the DE shrink quickly and the value of the SBX and UX rise. Similar to the previous category, the HHD reached a distribution that stay nearly the same. At about 2.000 FEs 50% of the population are distributed to the SBX operator, about 25% to the UX operator, 15% to the LX, 5% to the RSBX and the other 5% are distributed to LCX3, DE and CMAX. The production graph in Figure 4.4 (c) also shows, that the in the beginning the DE operator produced many offspring, but the curve quickly takes on an asymptotic shape. The other graphs are again nearly linear, as the distribution of the generation does not change a lot after 2.000 FEs. The most offspring are produced by UX and SBX operators.

In this case, the Hyper-Heuristics had only two valuable options. The SBX or the UX operator. The other operators stagnate on very high IGD values, which probably indicates, that they cannot find any sufficient solution to this problem. The SRXD, as one of the better performing Hyper-Heuristics on this problem, utilized both operators, but still had the problem, that it is forced to use the other operators, too. Thus, despite the seemingly working reward function, the SRXD does not perform better than the UX or the SBX operator. Surprisingly, it seems like the NCRXS did not even consider the UX operator, which could be caused be the updating conditions of the HHSs. The UX operator did probably not perform well in the first FEs, hence the exploration. If its probability is set to a low level from the beginning, it very unlikely that it is selected so that it cannot update its probability. This is the exact reason, why the HHSs have an exploration phase. Unfortunately, the length of that phase, is not long enough for that problem, so that the initial probabilities are already a problem for the exploitation phase. This is an indicator for the importance of the length of the exploration phase. The HHDs do not need an exploration, as all operators are always considered.

We stated our expectations at the beginning of this paragraph, that the Hyper-Heuristics are mostly better, especially the HHS because they have a well exploitation behaviour. This expectation is again not met. Regarding the trends for the problem of DTLZ4 in Appendix A.1.2, the Hyper-Heuristic operators perform at least in the same range as the single crossover operators. But here also, our expectation is not met. As explained above, the biggest problem for the Hyper-Heuristics are the update rates and methods and their exploratory behaviour. For the HHSs this means, the first generations are only used, to examine each of the seven operator. Regarding the graph in Figure 4.3 (a), at the first FEs we can see, that ranking of

the operators is not meaningful for the whole process. However, the probability of the most important operator in this case, the UX, is set to such a low level, that it is not selected for production and cannot update its values. This applies to all HHSs, which is why most of them did not perform well. Nevertheless, the Hyper-Heuristics used combinations of operators and surpassed most of the single crossover operators. With a better configuration of exploration and exploitation behaviour, this difficulty can be minimized.

3.) Category: one or two crossover operators perform worse than the other

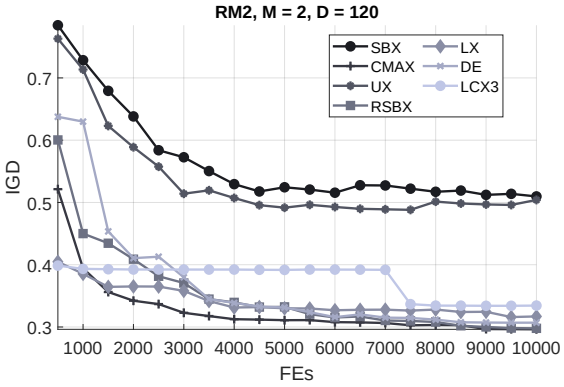
This category gathers all benchmark problems, where the single crossover operators have mostly similar IGD curves, except one or two that perform exceedingly worse. The benchmark problems belonging to this category are DTLZ3, RM1, RM2, WFG4 and WFG5. As the exemplary problem, we chose RM2. We expect the approximately same quality as the better single crossover operators, with some improvements on some Hyper-Heuristics. Most Hyper-Heuristics should be able to identify which operators do not work well, and thus, the quality should at least be in the same range as the better single crossover operators. Excluding bad performing operators is less complex than exploiting the well performing operators, as the probability of selection shrinks fast and the operator is most likely not selected again. Even HHD can minimize the influence of the bad performing operators.

We first describe the quality comparison over FEs and afterwards the behaviour of Hyper-Heuristics. At the end of this paragraph, we conclude our findings.

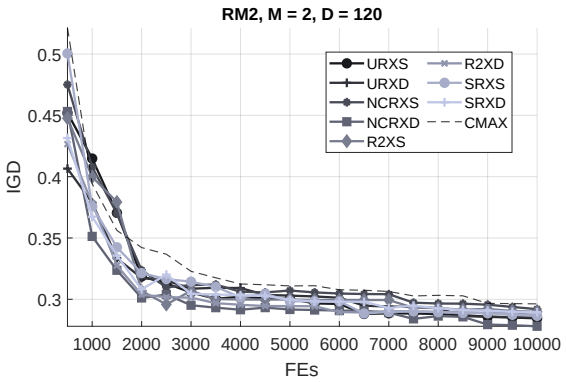
The first graph in Figure 4.5 shows the IGD values of the single crossover operators on the y-axis and the FEs on the x-axis. Each graph shows the quality trend of one single crossover operator. We chose this example because it firstly shows a clear separation, between the well and the bad performing operators, and secondly it is a rotated problem, which we want to pay special attention to. As we already know, the SBX and the UX operators have difficulties creating good solutions to rotated problems. In this graph we can see again, how the trend is good in the beginning, but flattens soon after 4.000 FEs on a high IGD value, compared to other operators. The LCX3 operator produces again an interesting IGD curve, where it starts already on a low value and does not change for about 7.000 FEs, and then takes a step downwards and stays nearly constant again. The remaining crossover operators start as usual with a steep descent and a flattening of the curve at about 4.000 FEs, converging to a low IGD value. The best performing operator here is the CMAX.

The second graph in Figure 4.5 shows the IGD values of the Hyper-Heuristic crossover operators on the y-axis and the FEs on the x-axis. Additionally, the CMAX operator is chosen for the comparison and is symbolized with a dashed line. All Hyper-Heuristics act again very similar. The all have a very steep descent and seem to flatten already after 2.500 FEs. They all end up in a slightly better value than the CMAX.

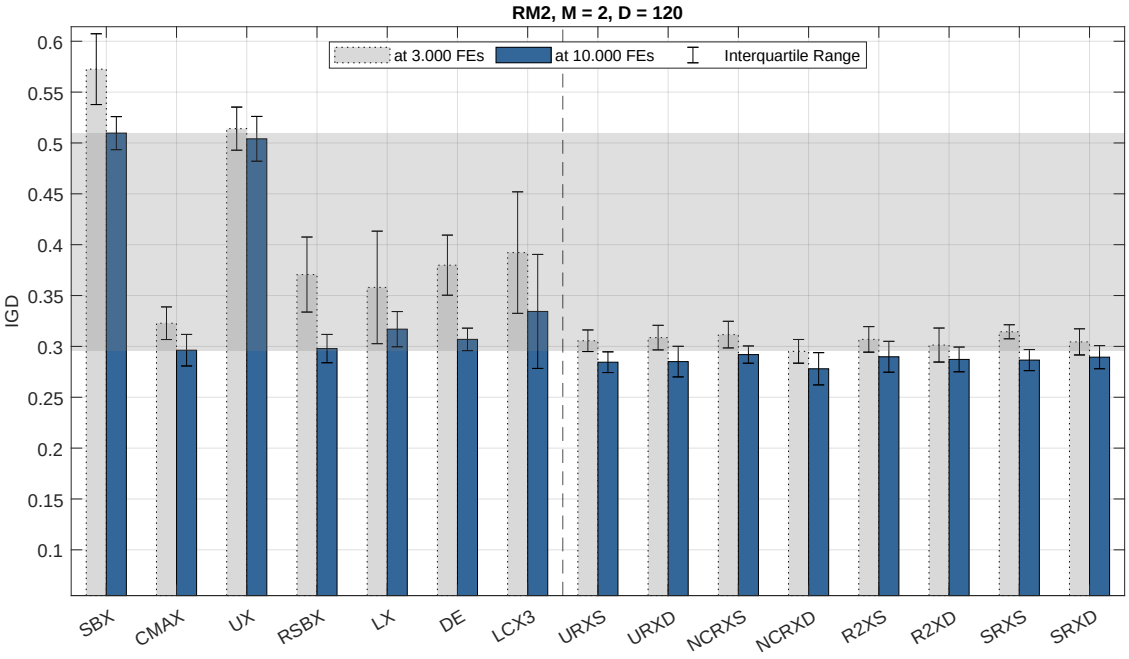
The third graph in Figure 4.5 is a bar chart with, again, IGD values on the y-axis. The graph shows the interim results as lighter bars after 3.000 FEs and the end results as darker bars after 10.000 FEs. Both bars demonstrate the values of the median run and include an error bar symbolizing the IQR. On the left side of the



(a) IGD of NSGA-II with single crossover operators.



(b) IGD of NSGA-II with hyper-heuristic crossover operators compared to CMAX+NSGA-II

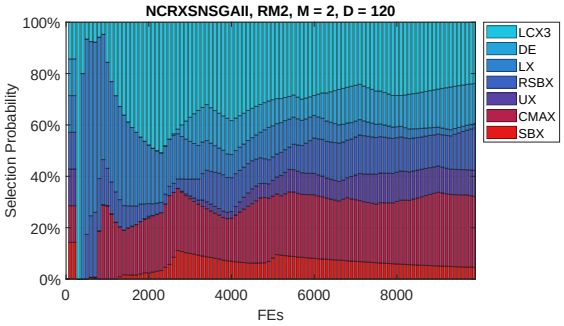


(c) IGD after 3000 FEs and after 10.000 FEs.

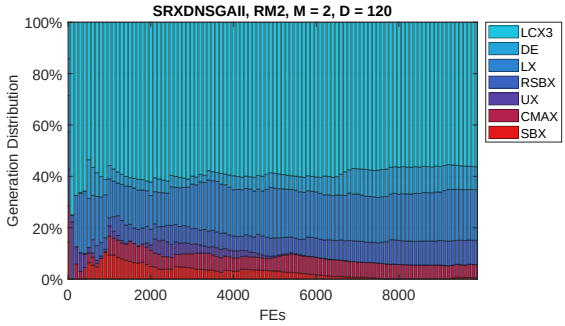
Figure 4.5: IGD measured on WFG9. The number of the decision variables is multiplied by 4.

dashed line in the centre are the results of the single crossover operators, on the right side are the results of the Hyper-Heuristics crossover operators. The darker horizontal bar highlights the interval of the worst median IGD result of the single crossover operators to the best. Similar to the first example in this section, all Hyper-Heuristics are below the quality interval of the single crossover operators. The early convergence of the IGD curves are also visible in this chart, as the interim results are very close to the end IGD values.

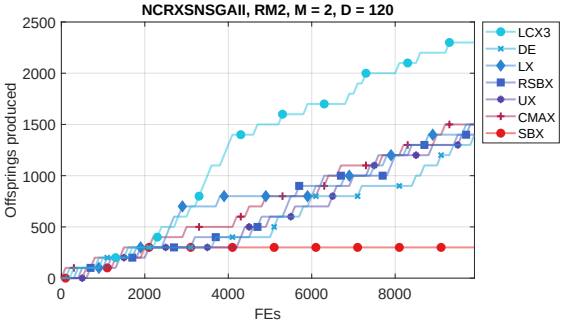
In Figure 4.6 the selection behaviour of NCRXS, left column, and SRXD, right column, are visualized. In the upper row, the change of selection probability respectively the generation distribution of each single crossover operator is illustrated. The y-axis is here the percentage of the operator, the x-axis are the FEs. We choose a stacked bar chart, that always adds up to 100%, to emphasize the ratios between the crossover operators. In the lower row, we use line graphs to show, how many new individuals



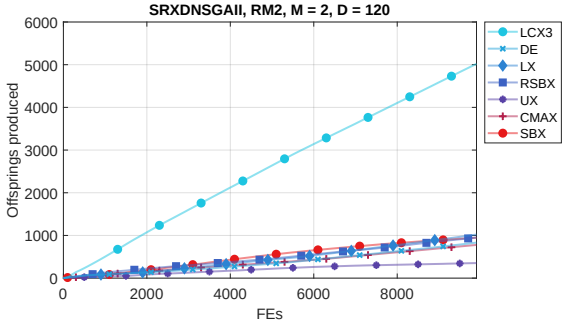
(a) Selection probabilities of crossover operators by NCRXS over the FEs.



(b) Distribution of the generations to the crossover operators by SRXD over the FEs.



(c) Cumulative number of individuals produced by each crossover operator in NCRXS.



(d) Cumulative number of individuals produced by each crossover operator inn SRXD.

Figure 4.6: Selection and distributions behaviour of NCRXS and SRXD on DTLZ7 with the number of the decision variables is multiplied by 4.

the operators produced cumulatively over FEs. Accordingly, the y-axis is the number of offspring and x-axis the FEs.

In Figure 4.6 (a) and (b) it is visible, that the NCRXS and the SRXD both avoided the SBX and the UX operators most of the time. The NCRXS starts with high probabilities for the RSBX, the LX and the LCX3 operator. The CMAX increases its probability and probability of the SBX is gradually rising. After 6.000 FEs, the probabilities do not change much. The focus of the selection are the CMAX and the LCX3 operators, with about 25% selection probability. The UX, RSBX and the DE operator are also higher valued as their selection probability is about 15%. Only the LX and the SBX are of vanishing importance. The production graph in Figure 4.6 (c) shows, that the LCX3 produced the most offspring in this process, although the frequency of selecting the LCX3 is decreasing after 4.000 FEs. The SBX is the only crossover operator, that is nearly not selected. The other operators are used alternately, which makes sense when the probabilities are similar.

The SRXD finds a constant distribution after about 6.000 FEs. The distribution itself stay remarkably free of variances for the whole process. The SRXD prefers the LCX3 operator, with nearly constantly 60% of all populations. The only other important operator is the LX, with about 20%. The SBX and the UX distribution are here vanishing as well. This is confirmed by the production graph in Figure 4.6 (d). The LCX3 produces the vast majority of all populations. The other operators are used nearly equally, except the UX, which is minimally utilized.

Again, both operators have in common, that they prefer to use the LCX3 operator. In this case, the operator did also perform well in the single crossover experiments.

At the beginning of the paragraph, we stated the expectation, that the Hyper-Heuristics perform approximately equal as the good single crossover operators because they can identify the bad performing operators and minimize their influence. This expectation is partially met, as the Hyper-Heuristics show in this example, that they can successfully exclude bad performing operators from the solution generation as far as possible. But this does not always apply regarding the other benchmark problems, which belong to this category (s. Appendix A.1.4). On DTLZ3 and WFG4, all Hyper-Heuristics are surpassed by a single crossover operator. On WFG5, only NCRXS, R2XD and SRXD perform better than the best single crossover operator. Solely on RM3, we can observe a similar result than on RM2. Thus, this might be connected to the rotation property. Interestingly, DTLZ3, WFG4 and WFG5 are all multimodal. Most likely, the interplay of the rotation property and the crossover operators is more unambiguous than the interplay between the multimodal property and the crossover operators. As we already know from the prior experiments, that it is not clear, which operators work best on multimodal problem. The UX operator delivered mostly good results, but that is not enough for solving it. Furthermore, by switching between different operators, it would be possible to be stuck in local optima even sooner, than a single crossover operator. Further investigations on explicitly multimodal problems and their interplay with Hyper-Heuristics would be necessary.

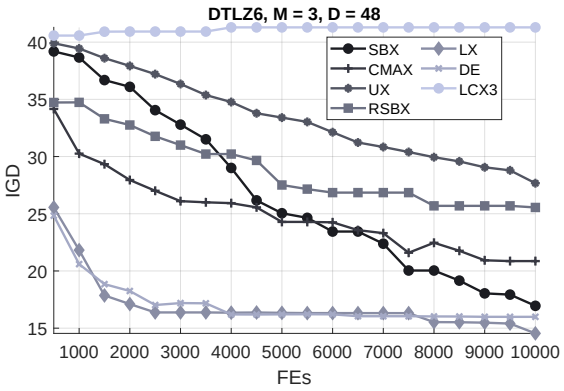
4.) Category: crossover operators cover a wider range of quality levels

This category gathers all benchmark problems, where the single crossover operators have different IGD curves so that they represent a wide quality range. The benchmark problems belonging to this category are DTLZ1, DTLZ6, WFG1, WFG2, WFG6, WFG7, WFG8. As the exemplary problem, we chose DTLZ6. We expect that

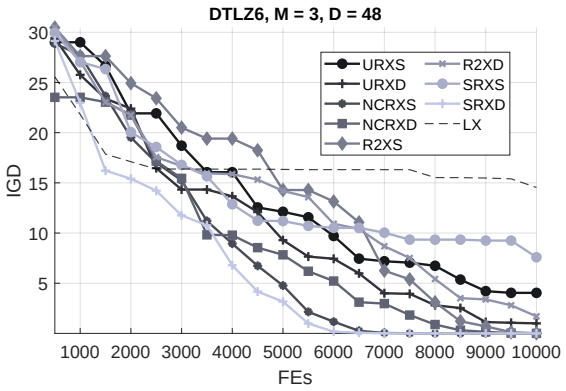
We first describe the quality comparison over FEs and afterwards the behaviour of Hyper-Heuristics. At the end of this paragraph, we conclude our findings.

The first graph in Figure 4.7 shows the IGD values of the single crossover operators on the y-axis and the FEs on the x-axis. Each graph shows the quality trend of one single crossover operator. We selected this example because all trends look differently and end up in very distant IGD values. The LCX3, for instance, starts at the very high IGD of 40, and stagnates there. The UX and the SBX also start that high. The UX has a nearly linear descent, whereas the SBX has steeper parts and thus end up and lower IGD value of about 17. The RSBX and the CMAX start at a lower IGD of 35. CMAX start with strong decrease, but then gradually flattens. The RSBX has from the beginning on only a moderate descent. Both end up higher than the SBX, that started with a much higher value. The LX and the DE operator act alike, starting lower than the other operators and flattening after 2.500 FEs above an IGD of 15.

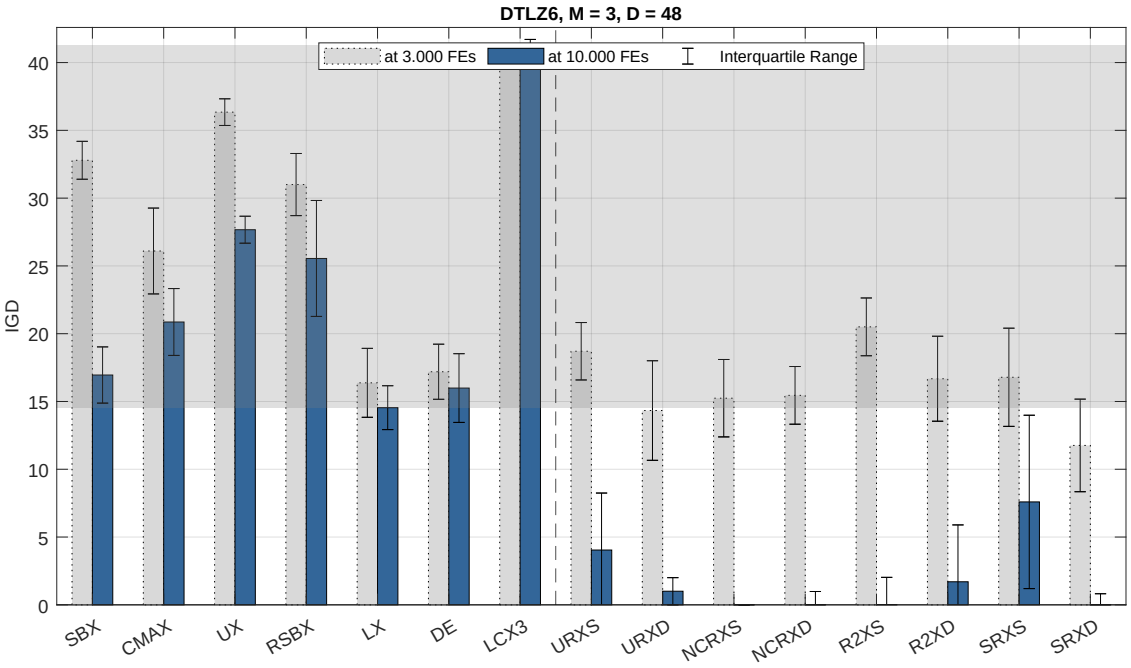
The second graph in Figure 4.7 shows the IGD values of the Hyper-Heuristic crossover operators on the y-axis and the FEs on the x-axis. Additionally, the LX operator is chosen for the comparison and is symbolized with a dashed line. In this case, the IGD curves of the Hyper-Heuristic operators act differently from each other. Foremost,



(a) IGD of NSGA-II with single crossover operators.



(b) IGD of NSGA-II with hyper-heuristic crossover operators compared to LX+NSGA-II

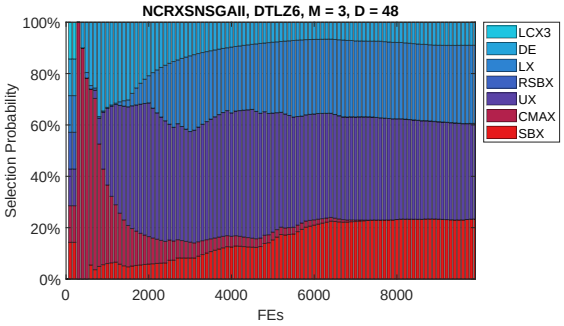


(c) IGD after 3000 FEs and after 10.000 FEs.

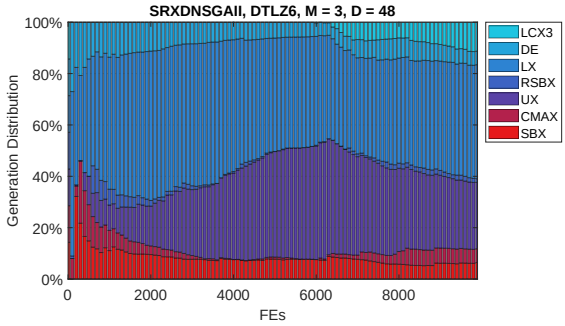
Figure 4.7: IGD measured on WFG9. The number of the decision variables is multiplied by .

they all end up very far below the best single crossover operator. They start with a steep descent and begin to flatten at about 7.000 FEs. Only the R2XS seem to decrease the value until the end. The Hyper-Heuristic with the steepest descent is the SRXD, followed by NCRXS. The SRXS flattens a lot sooner after 4.500 FEs at a higher IGD value. The R2XS has a lot of variation in the steepness, but also reaches a very low IGD value after 10.000 FEs.

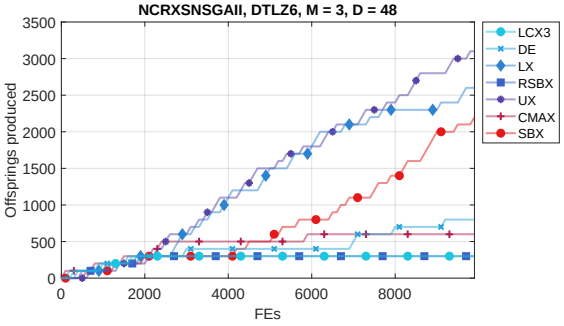
The third graph in Figure 4.7 is a bar chart with, again, IGD values on the y-axis. The graph shows the interim results as lighter bars after 3.000 FEs and the end results as darker bars after 10.000 FEs. Both bars demonstrate the values of the median run and include an error bar symbolizing the IQR. On the left side of the dashed line in the centre are the results of the single crossover operators, on the right side are the results of the Hyper-Heuristics crossover operators. The darker horizontal bar highlights the interval of the worst median IGD result of the single



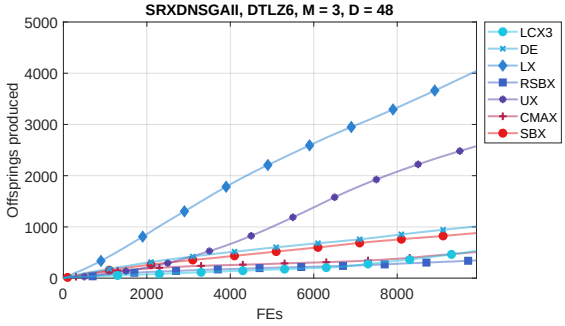
(a) Selection probabilities of crossover operators by NCRXS over the FEs.



(b) Distribution of the generations to the crossover operators by SRXD over the FEs.



(c) Cumulative number of individuals produced by each crossover operator in NCRXS.



(d) Cumulative number of individuals produced by each crossover operator inn SRXD.

Figure 4.8: Selection and distributions behaviour of NCRXS and SRXD on DTLZ7 with the number of the decision variables is multiplied by 4.

crossover operators to the best. The chart underlines the insights of the second graph. All IGD values of the Hyper-Heuristic operators are extremely low compared to the IGD values of the single crossover operators. Especially both NCR variants, R2XS and SRXD reached a value near zero, which is an extreme improvement to the single crossover operator values. Nevertheless, the IQR are partially enormous, which indicate that the quality is strongly varying in the other runs besides the median run. The IQR of NCRXS and SRXD, which are considered as the best HHS and HHD is comparably small.

In Figure 4.8 the selection behaviour of NCRXS, left column, and SRXD, right column, are visualized. In the upper row, the change of selection probability respectively the generation distribution of each single crossover operator is illustrated. The y-axis is here the percentage of the operator, the x-axis are the FEs. We choose a stacked bar chart, that always adds up to 100%, to emphasize the ratios between the crossover operators. In the lower row, we use line graphs to show, how many new individuals the operators produced cumulatively over FEs. Accordingly, the y-axis is the number of offspring and x-axis the FEs.

The NCRXS starts with a very high probability to pick CMAX operator. It is quickly replaced by the LCX3 and UX operator. The probability of selecting the CMAX operator is vanishing completely after 5.000 FEs. The SBX and LX operator gradually rise after 2.00 FEs. The RSBX and the LCX3 operator have a selection probability near zero for the whole process. In the end, the DE operator has a selection probability of about 10%, whereas the LX, the UX and the SBX have each

a probability of about 30%. The production graph in Figure 4.8 (c) shows that the most offspring were produced by the UX and the LX. The SBX is selected more often after about 4.000 FEs. The NCRXS rarely considers the other operators.

The SRXD behave a lot like the NCRXS. The first crossover combinations consist of the SBX, the CMAX, the LX and the DE operators. The portion of the population for the CMAX operator decreases until it is minimized around the 4.000th FE. At this point, the portion for the UX operator is increasing as well as the portion for the LX operator. Again, the distribution reaches a point, from where on it does not change the values much any more. After 6.000 FEs, the influence of the less important crossover operator rises again, so that for the last generations a more diverse distribution is used. The production graph in Figure 4.8 (d) shows, that the LX and the UX operator are used for most individuals.

Both Hyper-Heuristics refuse LCX, CMAX and RSBX, which correlates with the bad performance of these operators in the prior analysis. Surprisingly, although those operators were identified as not well performing, the UX operator, which had even worse results, is one of the most frequently used operators in both Hyper-Heuristics. However, they recognized, that the LX and the DE operator perform well on this problem. It is remarkable, that the Hyper-Heuristics combined the second worst performing operator, the UX, with the best performing operator, the LX, and performed massively better than the LX operator solely.

The results on the benchmark problems belonging to this category is varying. The Hyper-Heuristics outperform the best single crossover operators on DTLZ6 and WFG7. On WFG1, WFG2 and WFG6, only some Hyper-Heuristics perform better than the best single crossover operator. On DTLZ1 and WFG8, the best single crossover operator is superior to all Hyper-Heuristics. Nonetheless, the NCRXS delivers mostly better results than most of the single crossover operators on most problems.

Conclusion

In this section, we analyse different exemplary scenarios in detail to show the workwise of the Hyper-Heuristics. With this, we reach our fourth subgoal, to reason the selection behaviour and to identify advantages and disadvantages of Hyper-Heuristics.

In the first category, we examine the case, where all crossover operators achieve a similar quality. With these comparisons, we find out, that it is not relevant to have the best crossover operator in the pool, but to have a diverse pool. The combination of different operators, makes sure that the Hyper-Heuristics perform not only on the same level as the best crossover operator, but on a higher level. If a crossover operator performs well, depends not only on its properties, but also on the circumstance, like the distribution of the population and the phase of the solving process. We moreover conclude that the quality of the crossover operator in the single usage, is not decisive for the quality in the Hyper-Heuristic usage. For example, was the over bad performing LCX3 operator, often the preferred crossover operator of the Hyper-Heuristics, probably because it compensates the weaknesses of other operators very well. To find an optimal problem-solving process, a combination of crossover operator would most likely be superior to single crossover operator usages. To find

good combinations is manually really difficult because the available information are not unambiguous. Letting a Hyper-Heuristic learn online, which operators work best in the current setting, is an excellent alternative, as we show in this section.

In the second category, benchmarks ware gathered, where one or two operators are superior to the remaining. Unfortunately, this example showed, that the implemented Hyper-Heuristics are not flawless. In this analysis, we find out that the exploratory behaviour of the HHSs is not filed out, as it might exclude well performing operators, before they had a chance to perform well. For HHDs we find out, that it is a disadvantage that these Hyper-Heuristics are forced to use bad operators in ever generation. Both problems arise from the handling of the Exploration vs. Exploitation Dilemma.

In the third category, we analyse the case, that one or two operators perform worse than the remaining operators. It is remarkable that all Hyper-Heuristics can identify bad performing operators and reduce their influence to a minimum. To make that possible, the bad performing operator has to be clear, which works on rotated problems but not on, for example, multimodal problems. We note that our information about multimodal problems and operators, that work well on those, is too limited. More researches are necessary to find operators, that might also cover this case and to find methods for the Hyper-Heuristics to avoid local optimal solutions.

In the last category, we notice that the Hyper-Heuristics have difficulties in finding a good operator combination, when the operators are not clearly separable. If they sometimes work well, and sometimes not, the learning process would be disturbed. Fortunately, even a Hyper-Heuristic that selects the operators randomly outperforms most single crossover usages on most problems. Refinements on the available crossover operators, would be a good improvement to this problem.

4.2.4 Refinements and Final Comparison

In this section, the fifth subgoal, to readjust the parameters to show possible refinements, and the sixth subgoal, to give the last overview with the best Hyper-Heuristic of this work, are reached. In the prior section, we examine the behaviour and the features of the Hyper-Heuristics. We state advantages and disadvantages, which can be avoided by tuning the configurations of the Hyper-Heuristics. The biggest problem we recognize is the handling of the exploration and exploitation phases. On some problems, good operators are avoided too soon and can only hardly come back. On other problems, bad operators are kept for too long, which lowers the quality of the whole algorithm.

In the first part of this section, we analyse different refinements on the parameters of NCRXS and SRXD. In the second part, we finalize the evaluation by comparing the best Hyper-Heuristic variants to the single crossover operators.

Refinements

For this experimental analysis, we adjust different parameters on SRXD and NCRXS, to change the handling of the exploration and exploitation phases. We refer to the parameters, that were proposed in the Algorithms 3.1 and 3.2. In the following, we list the new algorithms and the changes we applied:

- LINEAR: Changes the reward-to-score function from cubic to linear
- LOGISTIC: Changes the reward-to-score function from a cubic function to the logistic function
- XOP3: Only use CMAX, RSBX and UX
- XOP16: Use CMAX, DE, LX and LCX with different parameters and fill the pool with 9 additional operators
- for distribution, LOS: "Last Operator Standing", sets the minimum portion to 0 so that we now allow the HHD to let down bad operators
- for selection, MINP0: We allow a minimal probability of 0. Beforehand, the minimum was 0.5%
- for selection, MINP5: We set the minimal probability to 5%
- for selection, EP5: increases the minimum exploration per operator from three to five generations
- for selection, EP7: increases the minimum exploration per operator from three to seven generations
- for selection, RR5: Changes the update rate of the reward function from every round to every fifth round.
- for selection, RR10: Changes the update rate of the reward function from every round to every tenth round.

The swapping of the reward-to-score functions, change the influence of the reward, so that the exploration phase could be emphasized. The cubic function leads to steep increase in scores, which accelerates the exploitation.

We reduce the selection pool to three operators: The UX operator, as the one that performs best and offers also good solutions for multimodal problems. The CMAX operator, as the operators that performs best on rotated problems, and a random operator, which is here the RSBX. For the growing selection pool, multiple variations of CMAX, DE, LX and LCX are added. With this, we show the impact of the preselection of crossover operators.

The "LOS" variant of the distribution is a possible solution to the missing exploitation in SRXD. When the minimal portion each operator receives is set to zero, bad performing operators would drop out of the selection pool. The equivalent for the HHSs is the change of the minimal probability, ***MINP***. This is per default set to 1%. Increasing this value to 5% could ease the comeback of operators, that perform bad in the beginning and well in the end.

Increasing ***EP***, the number of generations for the exploration phase per operator, and the ***RR***, the reward update rate, could lead to longer exploration phases for the selection and prevents to early drop out of good operators.

These algorithms are again used to solve the 20 benchmark problems with a D multiplied by four. This is executed with PlatEMO so that we present the results again in a PlatEMO table.

Table 4.7 summarizes the data collected, while the variants of SRXD solved the 20 benchmark problems with a D multiplied by four. It is noticeable, that the variant, using only three crossover operators, performed multiple times significantly better than the original. This was to expect, as too many operators can cover up the good ones. This is confirmed by the results of the XOP16 variant. This option performed worse than the original SRXD in 13 cases. The LOS variant, is not very promising as the results are mostly equal to and sometimes worse than the results of the original SRXD. Setting the minimum portion to zero can cause again the problem, that the operators are not well enough explored. Changing the reward-to-scoring function is interesting, as it directly influences the worth of a good reward. The original scoring function is cubic and therefore fulfils the goal that the operators benefit a lot from achieving the best reward. The logistic function uses a different approach by giving a nearly equal score to all operators that perform above the average. It supports the idea, of finding the best combination, more than exploiting the single best operator. The LOGISTIC variant was more often better than worse, thus examining different functions could be beneficial.

Table 4.7: Inverted Generational Distance of Tuned NCRXS+NSGA-II on DTLZ, RM and WFG with a number of decision variables multiplied by 4

Problem	M	D	LINEAR	LOGISTIC	LOS	XOP16	XOP3	SRXD
DTLZ1	3	28	5.0911e+1 (1.57e+1) −	5.2595e+1 (2.58e+1) −	4.1329e+1 (1.24e+1) ≈	5.1953e+1 (3.48e+1) −	3.0934e+1 (9.61e+0) +	3.9246e+1 (1.63e+1)
DTLZ2	3	48	9.6257e-2 (1.08e-2) +	9.7999e-2 (1.17e-2) +	1.0077e-1 (2.93e-2) +	1.3106e-1 (3.07e-2) −	8.9654e-2 (1.18e-2) +	1.1532e-1 (2.88e-2)
DTLZ3	3	48	5.4446e+2 (1.04e+2) −	5.5419e+2 (1.53e+2) −	3.8792e+2 (1.96e+2) ≈	2.4487e+2 (1.29e+2) +	2.8905e+2 (7.74e+1) +	4.4071e+2 (1.30e+2)
DTLZ4	3	48	1.0674e-1 (1.82e-2) ≈	9.1711e-2 (1.34e-2) +	1.0625e-1 (3.04e-2) ≈	1.3502e-1 (2.62e-2) −	1.0594e-1 (4.60e-1) ≈	1.1410e-1 (3.69e-2)
DTLZ5	3	48	2.5712e-2 (1.08e-2) +	2.5946e-2 (9.33e-3) +	2.4934e-2 (1.51e-2) ≈	4.7943e-2 (2.13e-2) −	2.0021e-2 (1.49e-2) +	3.2488e-2 (1.31e-2)
DTLZ6	3	48	5.7830e-3 (5.33e-4) ≈	5.7140e-3 (6.26e-4) ≈	1.1946e-2 (1.00e+0) −	9.9651e-1 (1.88e+0) −	5.5545e+0 (1.20e+1) −	5.9681e-3 (8.12e-1)
DTLZ7	3	88	1.2364e+0 (4.83e-1) −	9.1147e-1 (2.62e-1) ≈	1.3645e+0 (1.57e+0) −	3.7628e+0 (1.04e+0) −	3.2953e+0 (3.63e+0) ≈	9.0483e-1 (2.08e-1)
RM1	2	120	1.7815e-1 (6.39e-3) +	1.7717e-1 (8.48e-3) +	1.8377e-1 (5.71e-3) ≈	1.7985e-1 (7.51e-3) +	1.7657e-1 (1.83e-2) +	1.8299e-1 (5.90e-3)
RM2	2	120	2.8908e-1 (7.01e-3) ≈	2.9027e-1 (1.12e-2) ≈	2.8833e-1 (1.03e-2) ≈	2.8098e-1 (8.77e-3) +	2.8650e-1 (1.11e-2) ≈	2.9097e-1 (1.14e-2)
RM3	2	40	2.2855e+0 (3.39e-1) −	2.3483e+0 (2.03e-1) −	2.3560e+0 (3.51e-1) −	2.4629e+0 (3.36e-1) −	2.2884e+0 (3.75e-1) −	2.1076e+0 (2.04e-1)
RM4	3	48	4.7207e-1 (6.50e-2) +	4.6371e-1 (1.20e-1) ≈	4.2858e-1 (1.41e-1) +	5.4987e-1 (8.51e-2) ≈	4.4299e-1 (3.19e-1) +	5.1534e-1 (1.25e-1)
WFG1	3	48	1.2852e+0 (8.63e-2) −	1.2488e+0 (9.31e-2) −	1.3541e+0 (1.56e-1) −	1.5114e+0 (9.47e-2) −	1.2418e+0 (1.65e-1) ≈	1.2133e+0 (7.62e-2)
WFG2	3	48	2.7978e-1 (2.40e-2) ≈	2.8059e-1 (2.52e-2) ≈	2.7729e-1 (3.56e-2) ≈	3.0944e-1 (1.98e-2) −	2.5408e-1 (7.02e-2) ≈	2.7290e-1 (2.63e-2)
WFG3	3	48	3.0134e-1 (4.17e-2) ≈	2.9081e-1 (3.76e-2) ≈	3.0119e-1 (5.21e-2) ≈	3.3059e-1 (4.35e-2) −	2.4365e-1 (9.43e-2) +	3.0443e-1 (4.89e-2)
WFG4	3	48	3.1940e-1 (1.20e-2) ≈	3.1359e-1 (1.81e-2) ≈	3.1608e-1 (2.09e-2) ≈	3.3199e-1 (1.86e-2) −	2.9009e-1 (4.26e-2) +	3.1999e-1 (1.39e-2)
WFG5	3	48	3.0040e-1 (1.20e-2) −	2.9556e-1 (1.10e-2) ≈	3.0209e-1 (2.47e-2) −	3.0368e-1 (1.44e-2) −	3.9786e-1 (1.20e-1) −	2.8813e-1 (2.31e-2)
WFG6	3	48	3.5215e-1 (3.21e-2) ≈	3.4270e-1 (4.11e-2) +	3.6851e-1 (5.21e-2) ≈	3.5155e-1 (5.77e-2) +	3.5765e-1 (4.54e-2) ≈	3.6689e-1 (3.18e-2)
WFG7	3	48	3.4468e-1 (2.13e-2) +	3.4216e-1 (3.15e-2) +	3.4913e-1 (3.14e-2) +	3.8126e-1 (2.76e-2) ≈	3.5089e-1 (2.47e-2) +	3.7327e-1 (3.30e-2)
WFG8	3	48	4.4081e-1 (1.39e-2) +	4.3814e-1 (2.83e-2) ≈	4.4288e-1 (2.73e-2) ≈	4.6158e-1 (1.78e-2) −	4.3108e-1 (5.07e-2) +	4.4796e-1 (2.35e-2)
WFG9	3	48	3.1596e-1 (2.25e-2) ≈	3.1617e-1 (2.62e-2) ≈	3.2926e-1 (3.00e-2) −	3.1906e-1 (1.84e-2) ≈	3.1653e-1 (2.15e-2) ≈	3.1753e-1 (2.92e-2)
+/−/≈			6/6/8	6/4/10	3/6/11	4/13/3	10/3/7	

Table 4.8 summarizes the first part of the data collected, while the variants of NCRXS solved the 20 benchmark problems with a $\boldsymbol{D}$ multiplied by four. The first out of two tables include the changes analogue to the changes of the SRXD in Table 4.7.

In this table, nearly all variants on all problems perform equally or worse than the original NCRXS. Only the changes in the selection pool lead to an interesting result: The variant with 16 operators performed better than the original NCRXS on rotated problems, but worse on most others. It is needed to mention, that the added operators are mostly variants of the rotational invariant operators. Thus, the range of rotational invariant operators was increased and the chance of selecting a not rotational invariant operator is decreased. A similar phenomenon happens with a reduced selection pool. Here, is also an improvement on the rotated problem but also on other problems. Again, the pool consists of RSBX, CMAX and UX, thus two rotational invariant operators with one not rotational invariant operator. The chances of selecting a better crossover operator for rotated problem is therefore also in this case increased.

Table 4.8: Inverted Generational Distance of Tuned NCRXS+NSGA-II on DTLZ, RM and WFG with a number of decision variables multiplied by 4. Part 1

Problem	M	D	LINEAR	LOGISTIC	MINP0	MINP5	XOP16	XOP3	NCRXS
DTLZ1	3	28	3.4699e+1 (1.47e+1) ≈	5.3110e+1 (3.02e+1) −	2.8689e+1 (1.12e+1) +	4.2400e+1 (1.81e+1) −	1.8563e+1 (1.79e+1) +	2.1156e+1 (8.80e+0) +	3.7672e+1 (1.29e+1)
DTLZ2	3	48	7.9209e-2 (4.53e-3) ≈	8.0612e-2 (7.22e-3) ≈	7.8860e-2 (4.97e-3) ≈	8.1656e-2 (7.73e-3) ≈	8.0436e-2 (4.11e-3) ≈	7.3397e-2 (3.04e-3) +	7.9526e-2 (5.47e-3)
DTLZ3	3	48	4.6450e+2 (1.24e+2) ≈	4.9369e+2 (1.48e+2) −	4.3711e+2 (9.83e+1) ≈	4.5555e+2 (1.43e+2) ≈	9.9844e+1 (9.49e+1) +	2.4691e+2 (5.63e+1) +	4.4765e+2 (1.40e+2)
DTLZ4	3	48	9.0819e-2 (1.27e-2) ≈	8.9649e-2 (1.62e-2) ≈	8.8712e-2 (1.23e-2) ≈	9.2664e-2 (1.84e-2) −	9.9540e-2 (3.32e-2) −	7.5968e-2 (1.15e-2) +	8.4983e-2 (1.22e-2)
DTLZ5	3	48	1.9540e-2 (6.33e-3) ≈	1.9045e-2 (5.10e-3) ≈	2.0406e-2 (7.52e-3) ≈	1.9654e-2 (4.78e-3) ≈	1.7209e-2 (4.73e-3) ≈	1.2353e-2 (2.40e-3) +	1.9011e-2 (7.19e-3)
DTLZ6	3	48	6.2403e-3 (1.20e-3) ≈	6.0197e-3 (2.52e-3) ≈	6.4984e-3 (1.47e-3) ≈	6.5848e-3 (2.05e-3) ≈	6.1266e-3 (9.78e-1) ≈	3.5542e+0 (2.00e+0) −	6.2667e-3 (1.26e-3)
DTLZ7	3	88	1.4963e+0 (4.72e-1) ≈	1.5477e+0 (5.01e-1) ≈	1.3680e+0 (4.38e-1) ≈	1.4913e+0 (4.40e-1) ≈	2.5895e+0 (7.34e-1) −	1.2196e+0 (2.21e-1) +	1.4459e+0 (4.00e-1)
RM1	2	120	1.7906e-1 (1.06e-2) ≈	1.7471e-1 (1.13e-2) +	1.8102e-1 (7.74e-3) ≈	1.8145e-1 (1.08e-2) ≈	1.7063e-1 (5.30e-3) +	1.8345e-1 (7.23e-3) ≈	1.8027e-1 (9.06e-3)
RM2	2	120	2.8818e-1 (1.01e-2) ≈	2.9188e-1 (1.61e-2) ≈	2.9068e-1 (1.47e-2) ≈	2.9385e-1 (8.17e-3) ≈	2.6716e-1 (6.58e-3) +	2.9613e-1 (1.06e-2) ≈	2.9241e-1 (8.49e-3)
RM3	2	40	2.4653e+0 (3.55e-1) ≈	2.4166e+0 (2.45e-1) ≈	2.4017e+0 (3.57e-1) ≈	2.4188e+0 (2.76e-1) ≈	1.9989e+0 (1.69e-1) +	2.3156e+0 (2.58e-1) +	2.4102e+0 (3.75e-1)
RM4	3	48	4.7425e-1 (8.84e-2) ≈	4.3893e-1 (8.33e-2) ≈	4.7366e-1 (1.05e-1) ≈	4.2222e-1 (1.18e-1) ≈	4.9223e-1 (1.16e-1) −	4.1262e-1 (1.29e-1) ≈	4.3664e-1 (7.89e-2)
WFG1	3	48	1.2723e+0 (7.19e-2) −	1.2669e+0 (6.30e-2) −	1.1881e+0 (6.04e-2) ≈	1.2314e+0 (6.56e-2) −	1.3404e+0 (7.27e-2) −	1.3010e+0 (2.02e-1) −	1.1623e+0 (8.29e-2)
WFG2	3	48	2.6412e-1 (1.85e-2) ≈	2.7098e-1 (1.71e-2) ≈	2.7016e-1 (2.28e-2) ≈	2.6753e-1 (8.95e-3) ≈	2.8870e-1 (1.29e-2) −	2.4350e-1 (1.63e-2) +	2.6356e-1 (1.62e-2)
WFG3	3	48	2.7355e-1 (3.73e-2) ≈	2.7832e-1 (4.23e-2) ≈	2.7075e-1 (3.43e-2) ≈	2.7738e-1 (4.39e-2) ≈	3.0595e-1 (3.64e-2) −	2.4658e-1 (2.96e-2) +	2.7838e-1 (2.45e-2)
WFG4	3	48	2.9501e-1 (1.30e-2) −	3.0563e-1 (1.83e-2) −	2.8529e-1 (1.23e-2) ≈	3.0067e-1 (1.16e-2) −	3.0214e-1 (1.17e-2) −	2.8232e-1 (1.51e-2) ≈	2.8513e-1 (1.29e-2)
WFG5	3	48	2.9002e-1 (1.42e-2) −	2.9072e-1 (1.24e-2) −	2.8227e-1 (9.91e-3) ≈	2.9479e-1 (1.32e-2) −	2.8405e-1 (1.66e-2) ≈	2.9635e-1 (2.06e-2) −	2.7827e-1 (9.68e-3)
WFG6	3	48	3.1561e-1 (1.87e-2) −	3.2818e-1 (2.67e-2) −	3.0711e-1 (1.62e-2) ≈	3.2517e-1 (1.55e-2) −	3.2725e-1 (2.57e-2) −	3.2376e-1 (1.98e-2) −	3.0952e-1 (1.37e-2)
WFG7	3	48	3.3539e-1 (2.85e-2) ≈	3.3667e-1 (3.09e-2) ≈	3.3185e-1 (2.57e-2) ≈	3.2852e-1 (2.92e-2) ≈	3.6305e-1 (3.11e-2) −	3.2174e-1 (1.88e-2) +	3.4115e-1 (4.07e-2)
WFG8	3	48	4.1095e-1 (1.55e-2) −	4.1719e-1 (1.77e-2) −	4.0621e-1 (1.78e-2) −	4.0695e-1 (1.40e-2) −	4.3172e-1 (1.80e-2) −	3.7546e-1 (1.57e-2) +	3.9996e-1 (1.64e-2)
WFG9	3	48	3.0808e-1 (1.90e-2) ≈	3.1239e-1 (2.05e-2) ≈	3.0813e-1 (1.53e-2) ≈	3.0755e-1 (2.18e-2) ≈	3.0492e-1 (1.95e-2) ≈	3.0972e-1 (1.55e-2) ≈	3.0821e-1 (2.03e-2)
+/−/≈			0/5/15	1/7/12	1/1/18	0/7/13	5/10/5	11/4/5	

Table 4.9 shows that adjusting the exploration and exploitation by changing ***EP*** and ***RR*** did mostly not change the results. In the prior section, we analyse example trends of the selection and notice, that there are nearly no dramatical changes and the trend stagnates soon or changes very slowly. Thus, updating the reward only after 10 or 5 generation, has not much impact on the behaviour. Enlarging the exploration phase, did seemingly not lead to a more accurate scoring of the crossover operators. There is probably a better way to explore the operator's qualities, without wasting too many FEs on that. For example, the Hyper-Heuristic could start with a distribution, for the exploration and end with a selection for the exploitation.

Table 4.9: Inverted Generational Distance of Tuned NCRXS+NSGA-II on DTLZ, RM and WFG with a number of decision variables multiplied by 4. Part 2.

Problem	M	D	EP5	EP7	RR10	RR5	NCRXS
DTLZ1	3	28	3.7582e+1 (1.60e+1) ≈	4.2477e+1 (1.01e+1) −	3.4427e+1 (1.78e+1) ≈	3.0243e+1 (1.24e+1) ≈	3.7672e+1 (1.29e+1)
DTLZ2	3	48	7.7530e-2 (4.85e-3) +	7.7543e-2 (5.96e-3) ≈	7.9304e-2 (4.53e-3) ≈	7.8034e-2 (9.22e-3) ≈	7.9526e-2 (5.47e-3)
DTLZ3	3	48	4.6910e+2 (1.08e+2) ≈	5.2992e+2 (1.78e+2) −	4.3276e+2 (8.03e+1) ≈	4.5041e+2 (1.69e+2) ≈	4.4765e+2 (1.40e+2)
DTLZ4	3	48	8.6117e-2 (1.01e-2) ≈	9.0203e-2 (1.22e-2) ≈	8.5903e-2 (1.13e-2) ≈	8.6878e-2 (1.49e-2) ≈	8.4983e-2 (1.22e-2)
DTLZ5	3	48	2.0446e-2 (6.10e-3) ≈	1.8981e-2 (5.32e-3) ≈	2.1967e-2 (8.49e-3) −	2.0728e-2 (5.55e-3) ≈	1.9011e-2 (7.19e-3)
DTLZ6	3	48	6.3600e-3 (2.08e-3) ≈	7.3967e-3 (9.51e-1) −	5.9661e-3 (6.44e-4) ≈	6.1091e-3 (1.14e-3) ≈	6.2667e-3 (1.26e-3)
DTLZ7	3	88	1.5880e+0 (2.45e-1) ≈	1.6992e+0 (4.45e-1) −	1.5173e+0 (5.25e-1) ≈	1.5599e+0 (4.76e-1) ≈	1.4459e+0 (4.00e-1)
RM1	2	120	1.7944e-1 (8.55e-3) ≈	1.7775e-1 (5.75e-3) ≈	1.8257e-1 (8.33e-3) ≈	1.8078e-1 (8.77e-3) ≈	1.8027e-1 (9.06e-3)
RM2	2	120	2.9529e-1 (1.31e-2) ≈	2.9032e-1 (1.03e-2) ≈	2.9566e-1 (1.47e-2) ≈	2.9434e-1 (1.02e-2) ≈	2.9241e-1 (8.49e-3)
RM3	2	40	2.5036e+0 (3.44e-1) ≈	2.4148e+0 (3.87e-1) ≈	2.4181e+0 (2.86e-1) ≈	2.4665e+0 (4.24e-1) ≈	2.4102e+0 (3.75e-1)
RM4	3	48	4.4379e-1 (1.19e-1) ≈	4.2125e-1 (1.02e-1) ≈	4.5862e-1 (1.32e-1) ≈	4.8229e-1 (9.24e-2) ≈	4.3664e-1 (7.89e-2)
WFG1	3	48	1.2142e+0 (6.23e-2) −	1.2567e+0 (8.35e-2) −	1.1832e+0 (8.58e-2) ≈	1.1845e+0 (9.64e-2) ≈	1.1623e+0 (8.29e-2)
WFG2	3	48	2.7158e-1 (2.16e-2) ≈	2.7657e-1 (2.34e-2) −	2.6436e-1 (1.47e-2) ≈	2.6429e-1 (1.27e-2) ≈	2.6356e-1 (1.62e-2)
WFG3	3	48	2.6894e-1 (2.67e-2) ≈	2.7877e-1 (4.03e-2) ≈	2.7246e-1 (3.43e-2) ≈	2.6577e-1 (4.29e-2) ≈	2.7838e-1 (2.45e-2)
WFG4	3	48	2.8748e-1 (1.49e-2) ≈	2.8808e-1 (1.40e-2) ≈	2.8484e-1 (8.92e-3) ≈	2.8455e-1 (1.37e-2) ≈	2.8513e-1 (1.29e-2)
WFG5	3	48	2.8135e-1 (1.26e-2) ≈	2.7953e-1 (1.84e-2) ≈	2.7870e-1 (1.28e-2) ≈	2.8419e-1 (1.73e-2) ≈	2.7827e-1 (9.68e-3)
WFG6	3	48	3.1273e-1 (2.21e-2) ≈	3.1288e-1 (1.46e-2) ≈	3.0848e-1 (1.81e-2) ≈	3.0927e-1 (2.08e-2) ≈	3.0952e-1 (1.37e-2)
WFG7	3	48	3.2204e-1 (1.84e-2) ≈	3.3042e-1 (2.32e-2) ≈	3.3644e-1 (3.05e-2) ≈	3.3488e-1 (3.37e-2) ≈	3.4115e-1 (4.07e-2)
WFG8	3	48	4.0917e-1 (1.64e-2) −	4.1384e-1 (1.73e-2) −	4.0296e-1 (1.48e-2) ≈	3.9438e-1 (1.90e-2) ≈	3.9996e-1 (1.64e-2)
WFG9	3	48	3.0528e-1 (1.82e-2) ≈	3.0439e-1 (1.14e-2) ≈	3.0801e-1 (1.61e-2) ≈	3.0640e-1 (1.59e-2) ≈	3.0821e-1 (2.03e-2)
+/−/≈			1/2/17	0/7/13	0/1/19	0/0/20	

We consider the XOP3_NCRXS as the best overall Hyper-Heuristic. XOP3_SRXD is the best HHD, as they surpass their original Hyper-Heuristics more frequently than the other variation and they more often improve the performance than impair it. Thus, the last comparison is done with these variants.

Final Comparison

In this section, the last analysis is processed, and the last subgoal is reached. With this section, we give a last overview, about the best performing Hyper-Heuristic compared to all single crossover operators.

For this comparison, the data from the first experiment and most relevant data from the last experiment are combined in one PlatEMO table. We extend this table to show the comparison between all single crossover operators and each of XOP3_SRXD and XOP3_NCRXS. To accomplish this, each cell of the table has two markers now, the left symbolizes the results of the comparison with XOP3_SRXD and the right analogue for XOP3_NCRXS. There is also a fourth marker (o), which symbolizes that there is no comparison with the corresponding algorithm because the column belongs to that algorithm. Furthermore, a second summary row was added. The upper one summarizes the results of the comparison with XOP3_SRXD and the lower one analogue for XOP3_NCRXS. The results can be found in Table 4.10.

Table 4.10 shows the superiority of the Hyper-Heuristics to the single crossover operators. Both variants surpass the CMAX, the LXC3, the LX, the RSBX and the SBX on nearly all problems. Only the DE and the UX operator can keep up on a few problems, but are mostly also outperformed by the Hyper-Heuristics. In the comparison between the two Hyper-Heuristics, the XOP3_NCRXS performs better, except for three problems including rotated problems and WFG9, the problem were only the SBX operator performs good. This could be again an indicator, that the NCRXS does not explore enough and not frequent enough. As we know from our analyzations, a combination of UX and CMAX solves rotated problems better, than only UX or only CMAX. Thus, it is here more important, that the Hyper-Heuristic utilizes both alternately. Thus, the idea of combining HHDs and HHSs could be beneficial. Maybe, another Hyper-Heuristic would be useful, to select the best selection heuristic for each generation.

Table 4.10: IGD of NCRXS and SRXD with only three operators compared to the single crossover operators on DTLZ, RM and WFG with a number of decision variables multiplied by 4.

Problem	*M*	*D*	CMAX	DE	LCX3	LX	RSBX	UX	SBX	XOP3+SRXD	XOP3+NCRXS
DTLZ1	3	28	1.3051e+2 (1.54e+2) −/−	1.3657e+1 (3.56e+1) +/≈	4.5758e+2 (5.23e+1) −/−	8.7842e+1 (3.46e+1) −/−	4.0550e+2 (5.89e+1) −/−	2.8021e+1 (6.82e+0) ≈/−	4.1689e+1 (1.11e+1) −/−	3.0934e+1 (9.61e+0) o/−	2.1156e+1 (8.80e+0) +/o
DTLZ2	3	48	2.5952e-1 (4.08e-2) −/−	3.5053e-1 (4.91e-2) −/−	6.5261e-1 (2.23e-1) −/−	5.1957e-1 (7.57e-2) −/−	1.7191e-1 (3.90e-2) −/−	8.4655e-2 (6.87e-3) +/−	1.0340e-1 (1.02e-2) −/−	8.9654e-2 (1.18e-2) o/−	7.3397e-2 (3.04e-3) +/o
DTLZ3	3	48	2.1849e+2 (2.24e+2) +/≈	1.2427e+2 (3.16e+2) +/+	2.8606e+3 (2.35e+2) −/−	5.3459e+2 (1.66e+2) −/−	2.6124e+3 (2.82e+2) −/−	2.6012e+2 (4.50e+1) +/−	3.8932e+2 (9.09e+1) −/−	2.8905e+2 (7.74e+1) o/−	2.4691e+2 (5.63e+1) +/o
DTLZ4	3	48	9.7893e-1 (4.31e-2) −/−	7.0977e-1 (1.63e-1) −/−	1.0381e+0 (4.45e-2) −/−	1.0583e+0 (1.96e-1) −/−	7.7333e-1 (1.31e-1) −/−	9.1171e-2 (4.66e-1) ≈/−	1.2286e-1 (4.34e-1) −/−	1.0594e-1 (4.60e-1) o/−	7.5968e-2 (1.15e-2) +/o
DTLZ5	3	48	7.4963e-2 (3.53e-2) −/−	1.8644e-1 (4.41e-2) −/−	1.6515e-1 (2.72e-2) −/−	2.3802e-1 (9.42e-2) −/−	5.9278e-2 (2.33e-2) −/−	2.9694e-2 (9.06e-3) −/−	3.8987e-2 (1.16e-2) −/−	2.0021e-2 (1.49e-2) o/−	1.2353e-2 (2.40e-3) +/o
DTLZ6	3	48	2.0908e+1 (2.47e+0) −/−	1.6154e+1 (2.54e+0) −/−	4.1300e+1 (4.17e-1) −/−	1.5146e+1 (1.62e+0) −/−	2.5873e+1 (4.28e+0) −/−	2.7700e+1 (9.95e-1) −/−	1.7075e+1 (2.07e+0) −/−	5.5545e+0 (1.20e+1) o/−	3.5542e+0 (2.00e+0) +/o
DTLZ7	3	88	7.9289e+0 (8.57e-1) −/−	7.3604e+0 (8.31e-1) −/−	1.0457e+1 (8.11e-1) −/−	8.4172e+0 (1.04e+0) −/−	7.8837e+0 (6.76e-1) −/−	6.3786e-1 (6.39e-2) +/+	8.4442e-1 (1.85e-1) ≈/+	3.2953e+0 (3.63e+0) o/≈	1.2196e+0 (2.21e-1) ≈/o
RM1	2	120	1.8591e-1 (1.08e-2) −/−	1.9627e-1 (1.12e-2) −/−	2.2002e-1 (3.04e-2) −/−	1.9694e-1 (9.63e-3) −/−	1.8536e-1 (9.50e-3) −/≈	3.1063e-1 (5.89e-2) −/−	3.2392e-1 (6.06e-2) −/−	1.7657e-1 (1.83e-2) o/+	1.8345e-1 (7.23e-3) −/o
RM2	2	120	2.9658e-1 (1.65e-2) ≈/≈	3.0784e-1 (1.10e-2) −/−	3.3557e-1 (5.60e-2) −/−	3.1735e-1 (1.73e-2) −/−	2.9888e-1 (1.39e-2) −/≈	5.0495e-1 (2.20e-2) −/−	5.1121e-1 (1.62e-2) −/−	2.8650e-1 (1.11e-2) o/+	2.9613e-1 (1.06e-2) −/o
RM3	2	40	3.2367e+0 (2.35e-1) −/−	3.8133e+0 (3.38e-1) −/−	2.8294e+0 (2.80e-1) −/−	3.0036e+0 (3.68e-1) −/−	2.3261e+0 (3.52e-1) ≈/≈	2.1805e+0 (3.55e-1) ≈/+	2.6871e+0 (3.79e-1) −/−	2.2884e+0 (3.75e-1) o/≈	2.3156e+0 (2.58e-1) ≈/o
RM4	3	48	6.2730e-1 (1.21e-1) −/−	3.2677e-1 (1.59e-1) +/+	5.9346e-1 (1.91e-1) −/−	8.1436e-1 (1.72e-1) −/−	8.5435e-1 (1.91e-1) −/−	5.4186e-1 (6.55e-2) −/−	5.4905e-1 (7.04e-2) −/−	4.4299e-1 (3.19e-1) o/≈	4.1262e-1 (1.29e-1) ≈/o
WFG1	3	48	1.9930e+0 (3.73e-2) −/−	1.6952e+0 (2.95e-2) −/−	2.1803e+0 (1.04e-1) −/−	1.5957e+0 (4.56e-2) −/−	1.6317e+0 (7.44e-2) −/−	1.5766e+0 (7.86e-2) −/−	1.2488e+0 (7.84e-2) ≈/+	1.2418e+0 (1.65e-1) o/+	1.3010e+0 (2.02e-1) −/o
WFG2	3	48	3.9027e-1 (3.06e-2) −/−	4.2026e-1 (4.11e-2) −/−	6.7758e-1 (1.72e-1) −/−	4.5944e-1 (4.19e-2) −/−	4.2226e-1 (3.80e-2) −/−	2.4771e-1 (1.92e-2) ≈/−	2.9229e-1 (3.70e-2) −/−	2.5408e-1 (7.02e-2) o/−	2.4350e-1 (1.63e-2) +/o
WFG3	3	48	4.0666e-1 (3.88e-2) −/−	4.9583e-1 (3.71e-2) −/−	4.4644e-1 (3.89e-2) −/−	4.2654e-1 (5.13e-2) −/−	3.9635e-1 (6.35e-2) −/−	2.5306e-1 (3.79e-2) ≈/≈	3.2445e-1 (2.96e-2) −/−	2.4365e-1 (9.43e-2) o/≈	2.4658e-1 (2.96e-2) ≈/o
WFG4	3	48	3.4066e-1 (1.29e-2) −/−	3.9480e-1 (1.85e-2) −/−	5.5746e-1 (8.43e-2) −/−	3.8044e-1 (2.31e-2) −/−	3.5379e-1 (1.65e-2) −/−	2.6542e-1 (1.47e-2) +/+	3.1997e-1 (1.29e-2) −/−	2.9009e-1 (4.26e-2) o/−	2.8232e-1 (1.51e-2) +/o
WFG5	3	48	4.0457e-1 (1.10e-1) =/−	3.0355e-1 (1.90e-2) +/≈	9.7587e-1 (5.43e-2) −/−	3.1485e-1 (2.22e-2) +/−	5.3270e-1 (4.56e-2) −/−	3.0396e-1 (1.24e-2) +/≈	3.4070e-1 (2.12e-2) ≈/−	3.9786e-1 (1.20e-1) o/−	2.9635e-1 (2.06e-2) +/o
WFG6	3	48	4.9574e-1 (3.36e-2) −/−	3.2251e-1 (4.17e-2) +/≈	7.9809e-1 (5.57e-2) −/−	5.1765e-1 (1.47e-1) −/−	5.4631e-1 (2.82e-2) −/−	3.2196e-1 (2.16e-2) +/≈	3.7961e-1 (2.32e-2) −/−	3.5765e-1 (4.54e-2) o/−	3.2376e-1 (1.98e-2) +/o
WFG7	3	48	4.2011e-1 (2.67e-2) −/−	4.9870e-1 (2.38e-2) −/−	6.3657e-1 (4.81e-2) −/−	4.8521e-1 (3.10e-2) −/−	4.4220e-1 (2.95e-2) −/−	4.1445e-1 (9.14e-2) −/−	4.3422e-1 (8.50e-2) −/−	3.5089e-1 (2.47e-2) o/−	3.2174e-1 (1.88e-2) +/o
WFG8	3	48	4.9337e-1 (3.06e-2) −/−	5.6502e-1 (3.10e-2) −/−	8.1349e-1 (9.89e-2) −/−	5.7695e-1 (3.85e-2) −/−	5.1354e-1 (1.87e-2) −/−	3.3915e-1 (1.88e-2) +/+	3.8080e-1 (1.28e-2) +/≈	4.3108e-1 (5.07e-2) o/−	3.7546e-1 (1.57e-2) +/o
WFG9	3	48	3.5803e-1 (2.62e-2) −/−	3.4441e-1 (3.63e-2) −/−	4.3033e-1 (6.82e-2) −/−	3.7866e-1 (4.83e-2) −/−	3.7190e-1 (2.50e-2) −/−	3.8706e-1 (4.16e-2) −/−	4.0673e-1 (3.55e-2) −/−	3.1653e-1 (2.15e-2) o/≈	3.0972e-1 (1.65e-2) ≈/o
+/−/≈ to SRXD			1/18/1	5/15/0	0/20/0	1/19/0	0/19/1	7/8/5	1/16/3		12/3/5
+/−/≈ to NCRXS			0/18/2	2/15/3	0/20/0	0/20/0	0/17/3	4/13/3	2/17/1	3/12/5	

4.3 Summary of Results and Overall Discussion

In this chapter, we achieve all remaining subgoals and thus can answer our overall question, whether Hyper-Heuristics are suitable as selectors of crossover operators.

First, we achieve the third subgoal by analysing the quality of single crossover operators. We find out, that the UX and the SBX are superior on most problems, that we selected for this work. To our test suite, we added rotated problems to examine the performance on those. Not only did the RSBX outperform the SBX, which confirms the work of Pan et Al. [14], the other rotational invariant operators, CMAX and DE, perform even better in most cases. Thus, the interplay between rotated problems and crossover operators is clear, whereas the multimodality of problems creates greater challenges. We know that for multimodal problems, the spread of the offspring, needs to be high so that the algorithm does not stick in a local optimum. The UX operator, for example, fulfils this condition. Self-Adaptive operators, on the other hand, get into troubles when they do not increase the variance in each generation and converge too fast. All in all, we do not have enough information on multimodality and through our experiments we did not find a clear rule, which operator can solve those problems best. More research on multimodality would be necessary.

Secondly, we achieve the fourth subgoal, by analysing the quality and the behaviour of Hyper-Heuristics. We conclude from the quality analysis, that the Hyper-Heuristics are superior to most single crossover operators. This effect can be emphasized by increasing the complexity of the problem. We use a scaling of the number of decision variables. From the quality comparison, we selected the NCRXS as the best HHS and the SRXD as the best HHD. We noted that Hyper-Heuristics can run into difficulties, if, for instance, the pool of crossover operators does only offer one good operator, or it offers only equally well performing operators. It is remarkable, that we do not need the best operators for a problem in the selection pool, but diverse operators, so that they can compensate each other's weaknesses. Moreover, the quality of the crossover operator in the pool does not only depend on the quality, it can achieve solely, but also on other circumstance, like the distribution of the population or the state of the algorithm. This makes it even harder, to manually select a well performing crossover operator, thus the advantage of the use of Hyper-Heuristics is underlined.

We find further disadvantages of our implementation that include the balancing of the exploitation and the exploration. Whereas the HHDs are always exploring all operators quality, the HHSs are only exploiting the best performing operators. The selectors have a small exploration phase, where they use each operator a few times, before selecting the best with Roulette Wheel Selection. Nonetheless, problems arise in our experiments, where the best operator performed bad in the beginning and was nearly excluded from the process. On the other hand, the distributors always use all operators, even if they worsen the quality of the solutions. Adjustments to these features could lead to a great improvement.

Those adjustments are tested in the final part of our evaluation. We tune different parameters that change the exploration and exploitation behaviour, and compare those new versions to the original algorithms. The results show, that the most parameter tunings did not deliver sufficient results. Only one adjustment led to a

clear improvement: The shrinking of the operator pool. At the beginning of this work, we select seven crossover operators. Not all of these operators contribute to a quality increase, so we selected three operators: The UX for multimodal problems and because it is generally good. The CMAX because it performs impressively on rotated problems, and a randomly selected third operator, which is here the RSBX. As this new variant is considered as the best Hyper-Heuristics in this work, we chose them for the final comparison between the best Hyper-Heuristics and the single crossover operators. This last comparison underlines the superiority of the Hyper-Heuristics. They outperform nearly all single crossover operators and most problems. It is noticeable, that both variants, the NCRXS and SRXS, seem to still have some preferences in problems, where one of them performs better than the other. Thus, there is still optimization potential. For example, by combining both variants and use another higher-level heuristic to choose between the distribution for exploration and the selection for exploitation.

5. Conclusion

At the beginning of this work we state our overall goal, to show, that Hyper-Heuristic are suitable as selectors of crossover operators in MOEAs. We divided this goal into 6 subgoals:

1. Research on existing crossover operators and Hyper-Heuristics on MOEAs to identify their components
2. Design new variants of those components and combine them to propose different versions of Hyper-Heuristic online selectors for crossover operators
3. Perform an experimental evaluation with the selected crossover operators and identify their interplay with different problem properties
4. Perform an experimental evaluation with the proposed Hyper-Heuristics and identify the influence of the components on the result's quality
5. Readjust parameters of selected Hyper-Heuristics and analyse the result's quality to show, that further refinements are possible
6. Choose the best performing Hyper-Heuristics and compare them to NSGA-II with single crossover operators

Throughout this work, we accomplish all goals. The results are summarized in the next section, and afterwards an outlook on further works is given.

5.1 Summary

This work starts with clarification of basic information, that are necessary for this work. We researched prior works on crossover operators and Hyper-Heuristics to give an overview on their properties and their features. Doing this, we reach our first subgoal. We present five different crossover operators and their properties and review four different Hyper-Heuristic implementations to identify their main components: Selection Heuristic, Selection Pool, Reward function.

In the second chapter, we use this information to design our own Hyper-Heuristics. We present two different selection heuristics, selection and distribution, and four different reward functions, R2, Survival, NDS, CD and uniform reward. We combine these components to eight different Hyper-Heuristics Online Learner. Moreover, the selection pool for all variants is introduced. We modified existing algorithms and operators to add those to the selected crossover operators. In the end, we have seven different crossover operators: SBX, UX, RSBX, CMAX, DE, LCX3 and LX. Thus, we accomplish the second subgoal.

In the third chapter, we execute the experimental evaluation to reach the remaining goals. We first compared the crossover operators, recognizing, that the UX is superior to the most, but other operators surpass it on rotated problems. In contrast to the clear relationship between rotated problems and rotational invariant operators, the interplay between operators and multimodal problems is too unambiguous. Not only single crossover operators encounter difficulties on those problems, but also the Hyper-Heuristics in the second part of the evaluation. Nevertheless, the Hyper-Heuristics perform better than most single crossover operators on most problems. In the third part of the analysis, we make in depth analyses of the behaviour of Hyper-Heuristics. We mainly find out, that not only the selection of the best operator matters but also the best combination of operators. We conclude that a diverse selection pool, even with operators that did not perform well solely, can outperform the best single crossover operators. Thus, to optimize the crossover operators selection, not only one but a combination of operators should be found. This is even harder to do manually, thus the advantage of Hyper-Heuristics are underlined. Despite that, our implementation is not flawless, as we find problems with the balance of exploration and exploitation phases. Whereas the HHSs exploit the best crossover operators most of the time, the HHDs explores the quality of the crossover operators for the whole process. In the final part of the Evaluation, we adjust parameters that are responsible for this balance and try to find an improved version.

As a result of these tests, we receive the best versions of Hyper-Heuristics, which are namely the NCRXS, as the best HHS, and the SRXD, as the best HHS, both of them with a reduced selection pool, so that only UX, CMAX and RSBX are part of it. The reduction of the selection pool leads to an easier and faster exploration. It should be kept in mind, that the coverage of problem properties and a diverse set of operators, is still the most important insights on the selection pool.

To answer the main question of this work, whether Hyper-Heuristics are suitable as crossover selectors for MOEAs, we compare our final Hyper-Heuristics again with all single crossover operators. With an overwhelmingly clear result, we can say, yes, Hyper-Heuristics are absolutely suitable as crossover operator selectors. They improve the performance in nearly all cases and offer a simpler way of selecting the proper algorithm for a problem, as they are appropriate for a wider number of problems than single crossover operators.

5.2 Future Work

As an extension of this work, further researches on multimodal problems would be a good start. Additionally, more problem properties, for example, the separability

can be examined and clarified, how crossover operators could work around these properties. The main problem with multimodal problems is that the algorithm could be stuck in a local optimum. Instead of researching for crossover operators, that could solve this problem, maybe directly adding a feature to the Hyper-Heuristic would be helpful. For example, could the information about the improvements against the prior generations be used, to notice, when the algorithms converge to an optimum. In the case, that the improvement rate is low, the variance of the population could be increased by using a crossover operator that would do this.

The other problem with our Hyper-Heuristics is the balance between exploitation and exploration phases of the crossover operators. As stated before, a solution could be to combine a HHS, as the exploiter, with a HHD, as the explorer. In addition, it could be beneficial to not limit these phases, but to let another heuristic decide on the current phase. As we noticed in the in-depth analysis on Hyper-Heuristic behaviour, the quality of crossover operators depend on many circumstances, including the current distribution of the population or the phase of the algorithm, thus reevaluations of the qualities every few generations could improve this balancing problem.

A. Appendix

A.1 Experiment Results

A.1.1 Variants of NSGA-II in Comparison

Table A.1: Hyper Volume of NSGA-II with different crossover operators on DTLZ, RM and WFG with their default configurations.

Problem	M	D	CMAX	DE	LCX3	LX	RSBX	UX	SBX
DTLZ1	3	7	0.0000e+0 (0.00e+0) –	0.0000e+0 (0.00e+0) –	0.0000e+0 (0.00e+0) –	0.0000e+0 (0.00e+0) –	0.0000e+0 (0.00e+0) –	5.5323e-1 (5.89e-1) ≈	6.9244e-1 (7.52e-1)
DTLZ2	3	12	5.0935e-1 (1.14e-2) –	4.9602e-1 (1.13e-2) –	2.8466e-1 (5.67e-2) –	4.6336e-1 (3.08e-2) –	5.2270e-1 (6.61e-3) –	5.3418e-1 (5.96e-3) +	5.3076e-1 (3.91e-3)
DTLZ3	3	12	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ4	3	12	3.8983e-1 (1.88e-1) –	4.8936e-1 (1.27e-2) –	8.7919e-2 (3.76e-3) –	4.1118e-1 (6.15e-2) –	4.8251e-1 (1.09e-2) –	5.3430e-1 (1.93e-1) ≈	5.3058e-1 (1.18e-2)
DTLZ5	3	12	1.9834e-1 (3.83e-4) –	1.9647e-1 (5.65e-4) –	1.8597e-1 (7.36e-3) –	1.9817e-1 (3.39e-4) –	1.9844e-1 (4.19e-4) ≈	1.9876e-1 (2.33e-4) +	1.9851e-1 (3.35e-4)
DTLZ6	3	12	0.0000e+0 (0.00e+0) –	8.2645e-3 (1.96e-1) –	0.0000e+0 (0.00e+0) –	0.0000e+0 (7.03e-2) –	0.0000e+0 (0.00e+0) –	0.0000e+0 (0.00e+0) –	1.9948e-1 (2.97e-4)
DTLZ7	3	22	7.8722e-4 (2.83e-3) –	1.0440e-4 (7.16e-3) –	0.0000e+0 (0.00e+0) –	0.0000e+0 (1.66e-3) –	9.4934e-4 (3.51e-3) –	2.5772e-1 (6.01e-3) +	2.4558e-1 (1.13e-2)
RM1	2	30	7.1643e-1 (6.93e-3) +	7.1650e-1 (7.65e-4) +	5.2936e-1 (1.27e-2) –	6.7855e-1 (5.67e-2) +	6.8337e-1 (2.63e-2) +	5.7951e-1 (3.03e-2) +	5.5426e-1 (3.53e-2)
RM2	2	30	4.4168e-1 (9.82e-4) +	4.4088e-1 (1.09e-3) +	1.7543e-1 (1.54e-2) +	4.3100e-1 (8.91e-2) +	3.1823e-1 (6.80e-2) +	1.3238e-1 (3.98e-2) +	1.0089e-1 (2.73e-2)
RM3	2	10	1.0627e-1 (2.58e-2) +	1.9012e-1 (2.20e-1) +	0.0000e+0 (0.00e+0) –	1.7967e-1 (6.33e-2) +	6.1134e-2 (7.65e-2) +	0.0000e+0 (3.33e-3) ≈	0.0000e+0 (0.00e+0)
RM4	3	12	5.3327e-1 (3.91e-3) +	5.2346e-1 (4.74e-3) +	1.1336e-1 (3.86e-2) –	5.1735e-1 (8.55e-3) +	5.0496e-1 (8.87e-3) +	4.2024e-1 (3.27e-2) –	4.3272e-1 (2.58e-2)
WFG1	3	12	7.2021e-2 (4.00e-2) –	2.0079e-1 (2.30e-2) –	3.0126e-2 (5.10e-2) –	2.9365e-1 (3.89e-2) –	2.4752e-1 (2.85e-2) –	3.7830e-1 (8.38e-2) –	7.0611e-1 (8.35e-2)
WFG2	3	12	8.7320e-1 (1.30e-2) –	8.6391e-1 (1.31e-2) –	7.3637e-1 (4.74e-2) –	8.3028e-1 (1.93e-2) –	8.7329e-1 (1.40e-2) –	9.1732e-1 (5.69e-3) +	9.0761e-1 (6.41e-3)
WFG3	3	12	3.3033e-1 (1.07e-2) –	3.2558e-1 (1.45e-2) –	2.8497e-1 (2.13e-2) –	3.4208e-1 (2.69e-2) –	3.6911e-1 (1.59e-2) –	3.8559e-1 (7.43e-3) +	3.7811e-1 (4.36e-3)
WFG4	3	12	4.6690e-1 (1.01e-2) –	4.5074e-1 (7.39e-3) –	3.9355e-1 (1.95e-2) –	4.4664e-1 (1.24e-2) –	4.6124e-1 (9.33e-3) –	5.2737e-1 (5.26e-3) +	5.0646e-1 (7.18e-3)
WFG5	3	12	4.3637e-1 (3.06e-2) –	4.7148e-1 (1.49e-2) –	1.7585e-1 (1.65e-2) –	4.6220e-1 (1.45e-2) –	4.1595e-1 (2.97e-2) –	4.8816e-1 (5.94e-3) +	4.8252e-1 (8.98e-3)
WFG6	3	12	4.5968e-1 (2.13e-2) ≈	4.2219e-1 (2.14e-2) –	2.8300e-1 (3.86e-2) –	3.8582e-1 (2.38e-2) –	4.5111e-1 (1.69e-2) ≈	4.6505e-1 (2.69e-2) +	4.5147e-1 (1.76e-2)
WFG7	3	12	4.6647e-1 (1.70e-2) –	4.4689e-1 (1.39e-2) –	3.2658e-1 (1.92e-2) –	4.1722e-1 (2.84e-2) –	4.6757e-1 (1.71e-2) –	5.1496e-1 (6.93e-3) +	5.1303e-1 (8.70e-3)
WFG8	3	12	3.6499e-1 (1.34e-2) –	3.6310e-1 (1.23e-2) –	2.4520e-1 (3.17e-2) –	3.3419e-1 (2.73e-2) –	3.5916e-1 (1.53e-2) –	4.3509e-1 (1.09e-2) +	4.2736e-1 (6.64e-3)
WFG9	3	12	4.8001e-1 (1.14e-2) ≈	3.8640e-1 (6.58e-2) –	4.1446e-1 (3.42e-2) –	4.4932e-1 (2.34e-2) –	4.8146e-1 (1.15e-2) ≈	4.8581e-1 (8.81e-3) ≈	4.8592e-1 (1.25e-2)
+/−/≈			4/13/3	4/15/1	1/18/1	4/15/1	4/12/4	12/3/5	

Table A.2: Hyper Volume of NSGA-II with different Hyper-Heuristic crossover operators on DTLZ, RM and WFG with a number of decision variables multiplied by 4.

Problem	M	D	CMAX	DE	LCX3	LX	RSBX	UX	SBX
DTLZ1	3	28	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (2.49e-1) −	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ2	3	48	2.2751e-1 (3.78e-2) −	1.4124e-1 (4.11e-2) −	1.9131e-2 (5.39e-2) −	5.4452e-2 (2.38e-2) −	3.2255e-1 (4.64e-2) −	4.8652e-1 (2.15e-2) +	4.3879e-1 (2.48e-2)
DTLZ3	3	48	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) −	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ4	3	48	1.9633e-2 (1.33e-2) −	0.0000e+0 (2.15e-2) −	0.0000e+0 (0.00e+0) −	0.0000e+0 (0.00e+0) −	1.2997e-2 (2.92e-2) −	4.7841e-1 (1.70e-1) +	4.0042e-1 (9.71e-2)
DTLZ5	3	48	1.2220e-1 (2.44e-2) −	5.3526e-2 (1.65e-2) −	8.0797e-2 (2.14e-2) −	5.0582e-2 (3.31e-2) −	1.4868e-1 (1.89e-2) −	1.7796e-1 (6.01e-3) +	1.6658e-1 (1.08e-2)
DTLZ6	3	48	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ7	3	88	0.0000e+0 (0.00e+0) −	0.0000e+0 (0.00e+0) −	0.0000e+0 (0.00e+0) −	0.0000e+0 (0.00e+0) −	0.0000e+0 (0.00e+0) −	6.1756e-2 (1.05e-2) +	3.3299e-2 (2.24e-2)
RM1	2	120	5.2395e-1 (6.04e-3) +	5.1241e-1 (1.07e-2) +	4.9330e-1 (1.69e-2) +	5.0801e-1 (9.11e-3) +	5.2135e-1 (9.02e-3) +	4.6872e-1 (1.16e-2) +	4.5845e-1 (1.63e-2)
RM2	2	120	1.7385e-1 (1.03e-2) +	1.6096e-1 (7.30e-3) +	1.5318e-1 (4.52e-2) +	1.5624e-1 (1.69e-2) +	1.7386e-1 (1.21e-2) +	6.1698e-2 (7.63e-3) +	5.1375e-2 (7.79e-3)
RM3	2	40	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
RM4	3	48	5.0447e-2 (5.72e-2) +	2.1318e-1 (1.37e-1) +	6.7406e-2 (7.80e-2) +	6.3786e-3 (1.97e-2) −	0.0000e+0 (1.97e-3) −	2.4855e-2 (3.05e-2) ≈	1.9093e-2 (2.64e-2)
WFG1	3	48	5.4083e-2 (2.00e-2) −	1.7806e-1 (1.65e-2) −	0.0000e+0 (0.00e+0) −	2.7111e-1 (1.80e-2) −	2.1870e-1 (2.52e-2) −	1.9146e-1 (3.85e-2) −	3.9386e-1 (2.83e-2)
WFG2	3	48	7.5886e-1 (1.71e-2) −	7.3862e-1 (1.37e-2) −	6.4959e-1 (4.75e-2) −	7.2805e-1 (1.64e-2) −	7.4904e-1 (1.90e-2) −	8.5798e-1 (1.61e-2) +	8.3203e-1 (2.06e-2)
WFG3	3	48	2.4306e-1 (1.55e-2) −	2.0323e-1 (2.12e-2) −	2.0766e-1 (1.50e-2) −	2.3110e-1 (9.41e-3) −	2.5069e-1 (1.65e-2) −	3.1482e-1 (1.45e-2) +	2.8507e-1 (1.42e-2)
WFG4	3	48	4.2864e-1 (9.62e-3) −	4.0003e-1 (9.04e-3) −	3.4575e-1 (2.18e-2) −	4.0304e-1 (1.63e-2) −	4.1805e-1 (1.39e-2) −	4.9938e-1 (1.19e-2) +	4.5494e-1 (4.63e-3)
WFG5	3	48	3.8197e-1 (6.49e-2) −	4.5440e-1 (1.90e-2) +	1.3836e-1 (1.25e-2) −	4.4672e-1 (1.30e-2) +	3.0428e-1 (2.05e-2) −	4.5338e-1 (6.87e-3) +	4.3058e-1 (1.10e-2)
WFG6	3	48	3.3068e-1 (1.76e-2) −	4.8165e-1 (2.97e-2) +	2.2141e-1 (1.93e-2) −	4.3249e-1 (4.05e-2) +	3.1299e-1 (1.55e-2) −	4.4779e-1 (1.25e-2) +	4.0970e-1 (8.36e-3)
WFG7	3	48	3.7560e-1 (1.53e-2) −	3.3389e-1 (1.36e-2) −	2.9271e-1 (1.98e-2) −	3.3900e-1 (1.90e-2) −	3.6618e-1 (1.20e-2) −	3.9431e-1 (5.70e-2) ≈	3.8745e-1 (4.39e-2)
WFG8	3	48	3.3160e-1 (1.51e-2) −	3.0189e-1 (1.44e-2) −	2.3908e-1 (2.06e-2) −	2.9805e-1 (2.09e-2) −	3.2450e-1 (8.76e-3) −	4.2726e-1 (1.06e-2) +	4.0538e-1 (7.84e-3)
WFG9	3	48	4.1548e-1 (1.99e-2) +	4.3169e-1 (2.90e-2) +	3.8279e-1 (4.16e-2) ≈	4.1649e-1 (2.90e-2) +	4.0778e-1 (1.04e-2) +	3.9101e-1 (2.58e-2) ≈	3.7793e-1 (2.24e-2)
+/−/≈			4/12/4	6/12/2	3/12/5	5/11/4	3/13/4	12/1/7	

Table A.3: Hyper Volume of NSGA-II with different Hyper-Heuristic crossover operators on DTLZ, RM and WFG with a number of decision variables multiplied by 6.

Problem	M	D	CMAX	DE	LCX3	LX	RSBX	UX	SBX
DTLZ1	3	42	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) –	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ2	3	72	1.2647e-1 (4.41e-2) –	5.3370e-2 (3.02e-2) –	2.1627e-2 (5.35e-2) –	4.1308e-3 (1.27e-2) –	1.6832e-1 (3.68e-2) –	3.8916e-1 (4.67e-2) +	2.9982e-1 (4.59e-2)
DTLZ3	3	72	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) –	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ4	3	72	0.0000e+0 (0.00e+0) –	0.0000e+0 (0.00e+0) –	0.0000e+0 (0.00e+0) –	0.0000e+0 (0.00e+0) –	0.0000e+0 (0.00e+0) –	3.0556e-1 (1.11e-1) +	2.5506e-1 (3.46e-2)
DTLZ5	3	72	6.2793e-2 (2.49e-2) –	1.3793e-2 (1.41e-2) –	4.6286e-2 (1.62e-2) –	4.0271e-3 (8.18e-3) –	8.4457e-2 (1.77e-2) ≈	1.2707e-1 (2.42e-2) +	8.5883e-2 (1.65e-2)
DTLZ6	3	72	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ7	3	132	0.0000e+0 (0.00e+0) –	0.0000e+0 (0.00e+0) –	0.0000e+0 (0.00e+0) –	0.0000e+0 (0.00e+0) –	0.0000e+0 (0.00e+0) –	1.9750e-2 (8.01e-3) +	6.6569e-3 (4.83e-3)
RM1	2	180	5.1650e-1 (4.84e-3) +	5.0172e-1 (1.05e-2) +	4.7890e-1 (3.44e-2) +	4.9901e-1 (5.82e-3) +	5.1380e-1 (6.10e-3) +	4.4742e-1 (1.74e-2) +	4.2741e-1 (1.19e-2)
RM2	2	180	1.7413e-1 (1.01e-2) +	1.5448e-1 (1.31e-2) +	1.2025e-1 (5.97e-2) +	1.5627e-1 (1.07e-2) +	1.7237e-1 (9.07e-3) +	4.1161e-2 (9.75e-3) +	2.2896e-2 (1.74e-2)
RM3	2	60	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
RM4	3	72	2.4881e-2 (4.27e-2) +	3.1296e-2 (7.55e-2) +	4.3350e-2 (6.33e-2) +	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
WFG1	3	72	4.2851e-2 (7.34e-3) –	1.8696e-1 (1.82e-2) –	0.0000e+0 (0.00e+0) –	2.6680e-1 (1.03e-2) –	2.1021e-1 (2.37e-2) –	1.6408e-1 (3.14e-2) –	3.4701e-1 (6.58e-3)
WFG2	3	72	7.4544e-1 (7.98e-3) –	7.2702e-1 (1.25e-2) –	6.4255e-1 (4.96e-2) –	7.1312e-1 (1.31e-2) –	7.3492e-1 (1.46e-2) –	8.3158e-1 (1.17e-2) +	7.9837e-1 (1.55e-2)
WFG3	3	72	2.3218e-1 (9.07e-3) –	1.9865e-1 (1.71e-2) –	1.9152e-1 (1.43e-2) –	2.0713e-1 (1.45e-2) –	2.3865e-1 (1.26e-2) –	2.8712e-1 (2.40e-2) +	2.5189e-1 (1.35e-2)
WFG4	3	72	4.2334e-1 (6.73e-3) –	3.9161e-1 (1.17e-2) –	3.4587e-1 (1.85e-2) –	3.9473e-1 (1.87e-2) –	4.1537e-1 (1.33e-2) –	4.8722e-1 (1.23e-2) +	4.3666e-1 (8.48e-3)
WFG5	3	72	3.7381e-1 (4.19e-2) –	4.6009e-1 (2.81e-2) +	1.3435e-1 (1.33e-2) –	4.4342e-1 (2.71e-2) +	2.9624e-1 (3.97e-2) –	4.3172e-1 (7.42e-3) +	3.9674e-1 (6.74e-3)
WFG6	3	72	3.1805e-1 (1.35e-2) –	4.8431e-1 (2.32e-2) +	2.1429e-1 (2.16e-2) –	4.4720e-1 (6.28e-2) +	3.0132e-1 (1.18e-2) –	4.2184e-1 (1.43e-2) +	3.7442e-1 (1.07e-2)
WFG7	3	72	3.5928e-1 (7.58e-3) +	3.2545e-1 (8.69e-3) –	2.7890e-1 (1.84e-2) –	3.3320e-1 (1.16e-2) ≈	3.5204e-1 (9.90e-3) +	3.3498e-1 (1.67e-2) ≈	3.3327e-1 (1.28e-2)
WFG8	3	72	3.3266e-1 (9.04e-3) –	2.9639e-1 (8.85e-3) –	2.4735e-1 (2.59e-2) –	2.9684e-1 (1.40e-2) –	3.2322e-1 (1.02e-2) –	4.1512e-1 (6.37e-3) +	3.8621e-1 (1.01e-2)
WFG9	3	72	4.0422e-1 (1.36e-2) +	4.0778e-1 (2.90e-2) +	3.7624e-1 (3.18e-2) +	4.1580e-1 (2.45e-2) +	3.9348e-1 (2.43e-2) +	3.5443e-1 (2.73e-2) ≈	3.4377e-1 (2.27e-2)
+/−/≈			5/11/4	6/12/2	4/12/4	5/9/6	4/10/6	12/1/7	

Table A.4: Hyper Volume of NSGA-II with different Hyper-Heuristic crossover operators on DTLZ, RM and WFG with their default configurations.

Problem	*M*	*D*	NCRXD	NCRXS	R2XD	R2XS	SRXD	SRXS	URXD	URXS	SBX
DTLZ1	3	7	0.0000e+0 (1.83e-1) –	7.9528e-1 (9.18e-2) +	5.4720e-1 (6.59e-1) ≈	1.6213e-1 (6.86e-1) ≈	7.8403e-1 (6.12e-1) +	0.0000e+0 (5.77e-3) –	4.2698e-2 (1.32e-1) –	1.0624e-1 (3.72e-1) –	6.9244e-1 (7.52e-1)
DTLZ2	3	12	5.2692e-1 (6.92e-3) –	5.3301e-1 (7.33e-3) +	5.2831e-1 (7.76e-3) –	5.2735e-1 (9.13e-3) –	5.2594e-1 (6.39e-3) –	5.2785e-1 (8.15e-3) –	5.2991e-1 (6.22e-3) ≈	5.2881e-1 (5.89e-3) ≈	5.3076e-1 (3.91e-3)
DTLZ3	3	12	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ4	3	12	5.3221e-1 (6.16e-3) ≈	5.3564e-1 (3.96e-3) +	5.3120e-1 (9.18e-3) ≈	5.3184e-1 (7.34e-3) ≈	5.3213e-1 (5.58e-3) ≈	5.2736e-1 (8.13e-3) ≈	5.3139e-1 (4.15e-3) ≈	5.3064e-1 (6.02e-3) ≈	5.3058e-1 (1.18e-2)
DTLZ5	3	12	1.9888e-1 (3.98e-4) +	1.9929e-1 (3.25e-4) +	1.9899e-1 (5.48e-4) +	1.9876e-1 (5.04e-4) +	1.9902e-1 (4.07e-4) +	1.9856e-1 (5.59e-4) ≈	1.9889e-1 (3.85e-4) +	1.9883e-1 (3.76e-4) +	1.9851e-1 (3.35e-4)
DTLZ6	3	12	1.9954e-1 (3.74e-4) ≈	1.9954e-1 (5.47e-4) ≈	1.9951e-1 (2.47e-4) ≈	1.9957e-1 (4.14e-4) ≈	1.9957e-1 (2.56e-4) +	1.9927e-1 (8.17e-4) ≈	1.9956e-1 (2.80e-4) +	1.9954e-1 (3.39e-4) ≈	1.9948e-1 (2.97e-4)
DTLZ7	3	22	2.1824e-1 (2.02e-2) –	2.2686e-1 (3.24e-2) –	2.5298e-1 (1.16e-2) +	2.5023e-1 (1.77e-2) ≈	2.5038e-1 (1.08e-2) ≈	1.3715e-1 (8.37e-2) –	2.1546e-1 (1.72e-2) –	1.9622e-1 (1.99e-2) –	2.4558e-1 (1.13e-2)
RM1	2	30	7.1649e-1 (3.49e-3) +	7.1710e-1 (1.04e-2) +	6.9942e-1 (4.68e-2) +	7.1107e-1 (4.05e-2) +	6.7546e-1 (4.17e-2) +	7.1667e-1 (4.99e-3) +	7.1795e-1 (2.43e-3) +	7.1700e-1 (5.26e-3) +	5.5426e-1 (3.53e-2)
RM2	2	30	4.3907e-1 (1.04e-2) +	4.4193e-1 (1.24e-2) +	4.3177e-1 (3.85e-2) +	4.4064e-1 (3.09e-2) +	4.2275e-1 (7.49e-2) +	4.4041e-1 (5.59e-3) +	4.4256e-1 (1.09e-3) +	4.4101e-1 (4.05e-3) +	1.0089e-1 (2.73e-2)
RM3	2	10	2.0435e-1 (3.36e-2) +	2.1690e-1 (4.37e-2) +	1.9596e-1 (9.84e-2) +	1.8716e-1 (5.46e-2) +	2.2713e-1 (2.54e-2) +	1.7617e-1 (4.01e-2) +	1.9935e-1 (3.10e-2) +	1.9513e-1 (4.14e-2) +	0.0000e+0 (0.00e+0)
RM4	3	12	5.2403e-1 (5.30e-3) +	5.2519e-1 (6.92e-3) +	5.2597e-1 (7.34e-3) +	5.2719e-1 (8.28e-3) +	5.2647e-1 (6.77e-3) +	5.2462e-1 (8.35e-3) +	5.2681e-1 (4.75e-3) +	5.2719e-1 (6.07e-3) +	4.3272e-1 (2.58e-2)
WFG1	3	12	5.1981e-1 (8.70e-2) –	7.3500e-1 (8.78e-2) ≈	6.6618e-1 (1.08e-1) –	6.2224e-1 (9.21e-2) –	7.1972e-1 (7.50e-2) ≈	4.2044e-1 (9.82e-2) –	5.2029e-1 (1.05e-1) –	4.9330e-1 (6.85e-2) –	7.0611e-1 (8.35e-2)
WFG2	3	12	9.0548e-1 (5.75e-3) ≈	9.1101e-1 (4.60e-3) +	9.0841e-1 (7.81e-3) ≈	9.0482e-1 (8.26e-3) ≈	9.0794e-1 (4.21e-3) ≈	9.0687e-1 (5.55e-3) ≈	9.0487e-1 (3.21e-3) ≈	9.0546e-1 (7.33e-3) ≈	9.0761e-1 (6.41e-3)
WFG3	3	12	3.7179e-1 (1.06e-2) –	3.8085e-1 (8.19e-3) ≈	3.7835e-1 (1.59e-2) ≈	3.7168e-1 (1.13e-2) –	3.7526e-1 (1.74e-2) –	3.7347e-1 (1.16e-2) –	3.7339e-1 (9.58e-3) –	3.7218e-1 (1.22e-2) –	3.7811e-1 (4.36e-3)
WFG4	3	12	4.8347e-1 (8.12e-3) –	5.1856e-1 (1.06e-2) +	4.9183e-1 (1.50e-2) –	4.8745e-1 (8.68e-3) –	4.9026e-1 (1.32e-2) –	4.8180e-1 (9.57e-3) –	4.8563e-1 (5.96e-3) –	4.8390e-1 (1.14e-2) –	5.0646e-1 (7.18e-3)
WFG5	3	12	4.5953e-1 (7.47e-3) –	4.8791e-1 (4.88e-3) +	4.7747e-1 (8.06e-3) ≈	4.7275e-1 (1.19e-2) –	4.8163e-1 (1.04e-2) ≈	4.6216e-1 (1.62e-2) –	4.7216e-1 (7.65e-3) –	4.7082e-1 (1.11e-2) –	4.8252e-1 (8.98e-3)
WFG6	3	12	4.7672e-1 (1.10e-2) +	4.8931e-1 (1.30e-2) +	4.7253e-1 (1.77e-2) +	4.7569e-1 (1.60e-2) +	4.7336e-1 (1.24e-2) +	4.7765e-1 (1.52e-2) +	4.7574e-1 (9.59e-3) +	4.8005e-1 (1.65e-2) +	4.5147e-1 (1.76e-2)
WFG7	3	12	5.0530e-1 (7.80e-3) –	5.0712e-1 (8.31e-3) –	5.0265e-1 (1.68e-2) –	5.0271e-1 (1.19e-2) –	5.0251e-1 (1.54e-2) –	5.0138e-1 (8.16e-3) –	5.0224e-1 (7.73e-3) –	5.0256e-1 (7.59e-3) –	5.1303e-1 (8.70e-3)
WFG8	3	12	4.0519e-1 (1.14e-2) –	4.2442e-1 (8.44e-3) –	4.0388e-1 (2.41e-2) –	4.0148e-1 (1.60e-2) –	3.7896e-1 (1.75e-2) –	4.0457e-1 (1.42e-2) –	4.0130e-1 (1.43e-2) –	4.0444e-1 (1.21e-2) –	4.2736e-1 (6.64e-3)
WFG9	3	12	4.8688e-1 (6.50e-3) ≈	4.8991e-1 (9.74e-3) +	4.8774e-1 (8.60e-3) ≈	4.8479e-1 (8.65e-3) ≈	4.8554e-1 (1.36e-2) ≈	4.8823e-1 (9.09e-3) ≈	4.8982e-1 (7.24e-3) ≈	4.8838e-1 (8.96e-3) ≈	4.8592e-1 (1.25e-2)
+/−/≈			6/9/5	13/3/4	7/5/8	6/7/7	8/5/7	5/9/6	7/8/5	6/8/6	

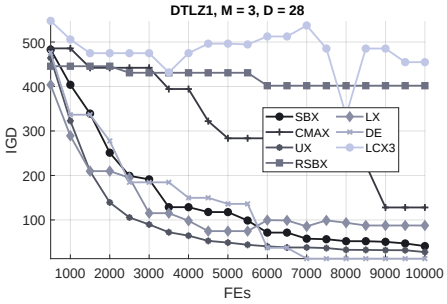
Table A.5: Hyper Volume of NSGA-II with different Hyper-Heuristic crossover operators on DTLZ, RM and WFG with a number of decision variables multiplied by 4.

Problem	M	D	NCRXD	NCRXS	R2XD	R2XS	SRXD	SRXS	URXD	URXS	SBX
DTLZ1	3	28	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ2	3	48	4.6920e-1 (2.43e-2) +	4.8962e-1 (1.47e-2) +	4.4845e-1 (4.15e-2) ≈	4.3097e-1 (6.04e-2) ≈	4.0184e-1 (5.41e-2) −	4.7205e-1 (2.22e-2) +	4.5393e-1 (2.31e-2) +	4.6809e-1 (1.66e-2) +	4.3879e-1 (2.48e-2)
DTLZ3	3	48	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ4	3	48	4.1830e-1 (5.22e-2) +	4.7833e-1 (3.07e-2) +	4.5302e-1 (3.13e-2) +	4.6841e-1 (5.29e-2) +	4.0829e-1 (6.69e-2) ≈	4.4157e-1 (4.62e-2) +	4.2766e-1 (4.39e-2) +	4.5029e-1 (4.98e-2) +	4.0042e-1 (9.71e-2)
DTLZ5	3	48	1.8255e-1 (5.45e-3) +	1.8302e-1 (5.96e-3) +	1.7533e-1 (7.36e-3) +	1.7279e-1 (1.98e-2) +	1.7087e-1 (1.49e-2) ≈	1.8299e-1 (8.17e-3) +	1.8011e-1 (7.89e-3) +	1.8409e-1 (4.16e-3) +	1.6658e-1 (1.08e-2)
DTLZ6	3	48	1.9522e-1 (1.99e-1) +	1.9928e-1 (7.06e-4) +	0.0000e+0 (1.91e-1) −	0.0000e+0 (1.99e-1) −	1.9916e-1 (2.00e-1) +	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (1.22e-1) −	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ7	3	88	2.5112e-3 (4.16e-3) −	3.6198e-3 (4.47e-3) −	3.6026e-2 (4.39e-2) ≈	1.1051e-2 (2.75e-2) −	2.5905e-2 (2.08e-2) ≈	0.0000e+0 (0.00e+0) −	4.3356e-4 (1.63e-3) −	0.0000e+0 (0.00e+0) −	3.3299e-2 (2.24e-2)
RM1	2	120	5.4369e-1 (6.22e-3) +	5.3185e-1 (7.75e-3) +	5.2468e-1 (7.33e-3) +	5.3104e-1 (9.04e-3) +	5.2529e-1 (5.90e-3) +	5.4089e-1 (9.80e-3) +	5.3650e-1 (7.40e-3) +	5.3775e-1 (7.08e-3) +	4.5845e-1 (1.63e-2)
RM2	2	120	1.8503e-1 (8.94e-3) +	1.7513e-1 (9.32e-3) +	1.8041e-1 (1.31e-2) +	1.7733e-1 (1.23e-2) +	1.7829e-1 (9.28e-3) +	1.7824e-1 (7.25e-3) +	1.8012e-1 (1.01e-2) +	1.7910e-1 (8.01e-3) +	5.1375e-2 (7.79e-3)
RM3	2	40	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
RM4	3	48	1.6045e-1 (5.79e-2) +	1.3420e-1 (4.54e-2) +	5.8128e-2 (5.46e-2) +	8.3768e-2 (6.76e-2) +	1.0081e-1 (7.58e-2) +	2.0107e-1 (1.33e-1) +	1.5854e-1 (5.91e-2) +	1.7086e-1 (8.00e-2) +	1.9093e-2 (2.64e-2)
WFG1	3	48	2.9412e-1 (2.39e-2) −	4.1605e-1 (2.76e-2) +	3.6754e-1 (3.50e-2) −	3.4322e-1 (2.91e-2) −	4.0934e-1 (2.49e-2) +	2.8888e-1 (5.30e-2) −	3.2021e-1 (2.69e-2) −	3.2477e-1 (3.84e-2) −	3.9386e-1 (2.83e-2)
WFG2	3	48	8.3046e-1 (1.54e-2) ≈	8.4766e-1 (6.88e-3) +	8.3689e-1 (2.57e-2) ≈	8.3250e-1 (1.97e-2) ≈	8.3911e-1 (1.67e-2) ≈	8.3574e-1 (1.60e-2) ≈	8.3438e-1 (9.90e-3) ≈	8.3149e-1 (1.64e-2) ≈	8.3203e-1 (2.06e-2)
WFG3	3	48	2.7910e-1 (1.20e-2) −	2.9505e-1 (1.10e-2) +	2.7062e-1 (1.61e-2) −	2.8174e-1 (1.69e-2) ≈	2.7687e-1 (1.25e-2) −	2.8606e-1 (1.48e-2) ≈	2.8477e-1 (1.38e-2) ≈	2.7966e-1 (1.56e-2) ≈	2.8507e-1 (1.42e-2)
WFG4	3	48	4.5202e-1 (8.82e-3) ≈	4.8656e-1 (1.33e-2) +	4.5677e-1 (8.95e-3) ≈	4.5191e-1 (5.09e-3) −	4.4936e-1 (1.19e-2) −	4.4975e-1 (1.44e-2) −	4.5465e-1 (8.21e-3) ≈	4.5154e-1 (8.83e-3) ≈	4.5494e-1 (4.63e-3)
WFG5	3	48	4.1362e-1 (1.73e-2) −	4.8315e-1 (8.68e-3) +	4.6570e-1 (1.30e-2) +	4.4672e-1 (3.37e-2) +	4.7456e-1 (1.09e-2) +	4.1458e-1 (3.50e-2) −	4.4892e-1 (1.82e-2) +	4.3741e-1 (2.03e-2) +	4.3058e-1 (1.10e-2)
WFG6	3	48	4.2606e-1 (1.94e-2) +	4.6203e-1 (9.63e-3) +	4.3709e-1 (4.32e-2) +	4.2579e-1 (3.45e-2) +	4.1417e-1 (2.08e-2) ≈	4.1523e-1 (2.82e-2) ≈	4.3084e-1 (1.38e-2) +	4.2296e-1 (1.28e-2) +	4.0970e-1 (8.36e-3)
WFG7	3	48	4.3200e-1 (1.58e-2) +	4.3777e-1 (2.41e-2) +	4.3959e-1 (1.49e-2) +	4.3256e-1 (1.55e-2) +	4.0904e-1 (2.36e-2) +	4.3659e-1 (1.63e-2) +	4.3119e-1 (1.51e-2) +	4.3823e-1 (2.62e-2) +	3.8745e-1 (4.39e-2)
WFG8	3	48	3.7064e-1 (1.01e-2) −	3.8769e-1 (1.13e-2) −	3.6853e-1 (1.40e-2) −	3.6790e-1 (1.28e-2) −	3.5587e-1 (1.11e-2) −	3.6499e-1 (1.30e-2) −	3.6923e-1 (8.99e-3) −	3.6761e-1 (7.07e-3) −	4.0538e-1 (7.84e-3)
WFG9	3	48	4.4542e-1 (1.43e-2) +	4.5037e-1 (1.03e-2) +	4.4966e-1 (1.88e-2) +	4.4080e-1 (1.58e-2) +	4.4285e-1 (1.41e-2) +	4.4578e-1 (1.67e-2) +	4.4849e-1 (1.04e-2) +	4.4382e-1 (1.50e-2) +	3.7793e-1 (2.24e-2)
+/−/≈			10/5/5	15/2/3	9/4/7	9/5/6	8/4/8	8/5/7	10/4/6	10/3/7	

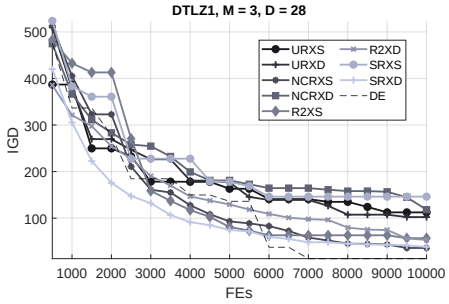
Table A.6: Hyper Volume of NSGA-II with different Hyper-Heuristic crossover operators on DTLZ, RM and WFG with a number of decision variables multiplied by 6.

Problem	*M*	*D*	NCRXD	NCRXS	R2XD	R2XS	SRXD	SRXS	URXD	URXS	SBX
DTLZ1	3	42	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ2	3	72	3.8050e-1 (4.79e-2) +	4.1592e-1 (4.41e-2) +	3.0304e-1 (6.45e-2) ≈	3.0655e-1 (1.10e-1) ≈	2.9423e-1 (7.64e-2) ≈	3.7898e-1 (4.84e-2) +	3.5641e-1 (5.56e-2) +	3.8609e-1 (3.31e-2) +	2.8560e-1 (4.11e-2)
DTLZ3	3	72	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ4	3	72	2.7132e-1 (5.49e-2) +	3.6582e-1 (6.21e-2) +	3.3695e-1 (5.41e-2) +	3.8463e-1 (7.21e-2) +	2.9834e-1 (5.66e-2) +	3.0995e-1 (8.59e-2) +	2.8329e-1 (5.57e-2) +	3.2407e-1 (6.78e-2) +	2.4855e-1 (4.34e-2)
DTLZ5	3	72	1.3037e-1 (2.26e-2) +	1.4484e-1 (2.74e-2) +	1.2032e-1 (4.51e-2) +	1.1902e-1 (3.57e-2) +	1.1286e-1 (3.61e-2) +	1.4678e-1 (3.18e-2) +	1.3020e-1 (3.06e-2) +	1.4101e-1 (1.66e-2) +	7.3106e-2 (1.71e-2)
DTLZ6	3	72	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (1.99e-1) −	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) −	0.0000e+0 (1.99e-1) −	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ7	3	132	0.0000e+0 (0.00e+0) −	0.0000e+0 (2.30e-4) −	4.4998e-3 (7.69e-3) −	6.6838e-5 (2.38e-3) −	2.5708e-3 (3.78e-3) −	0.0000e+0 (0.00e+0) −	0.0000e+0 (0.00e+0) −	0.0000e+0 (0.00e+0) −	8.0500e-3 (4.65e-3)
RM1	2	180	5.3026e-1 (4.15e-3) +	5.2005e-1 (5.30e-3) +	5.1783e-1 (4.75e-3) +	5.2016e-1 (7.52e-3) +	5.1642e-1 (4.73e-3) +	5.2784e-1 (6.26e-3) +	5.2650e-1 (2.61e-3) +	5.2468e-1 (4.58e-3) +	4.2741e-1 (1.19e-2)
RM2	2	180	1.8137e-1 (9.07e-3) +	1.7192e-1 (9.51e-3) +	1.7883e-1 (8.76e-3) +	1.7288e-1 (7.53e-3) +	1.8051e-1 (6.57e-3) +	1.7087e-1 (9.51e-3) +	1.7432e-1 (7.07e-3) +	1.7500e-1 (6.34e-3) +	2.2896e-2 (1.74e-2)
RM3	2	60	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
RM4	3	72	4.4349e-2 (3.72e-2) +	3.7443e-2 (3.34e-2) +	8.5929e-3 (3.02e-2) +	2.5604e-2 (4.29e-2) +	1.8558e-2 (2.46e-2) +	5.2180e-2 (4.88e-2) +	4.7828e-2 (5.96e-2) +	4.7804e-2 (5.48e-2) +	0.0000e+0 (0.00e+0)
WFG1	3	72	2.7657e-1 (3.04e-2) −	3.7228e-1 (2.23e-2) +	3.3448e-1 (2.72e-2) −	3.1227e-1 (3.05e-2) −	3.5436e-1 (2.47e-2) ≈	2.6934e-1 (2.46e-2) −	2.9882e-1 (1.92e-2) −	3.0054e-1 (2.76e-2) −	3.5006e-1 (1.41e-2)
WFG2	3	72	8.1184e-1 (1.54e-2) +	8.2965e-1 (1.43e-2) +	8.1575e-1 (2.45e-2) +	8.0537e-1 (3.24e-2) ≈	8.1687e-1 (1.30e-2) +	8.1188e-1 (1.14e-2) +	8.1076e-1 (1.42e-2) +	8.1299e-1 (1.22e-2) +	7.9655e-1 (1.52e-2)
WFG3	3	72	2.5637e-1 (1.37e-2) +	2.7958e-1 (1.40e-2) +	2.5609e-1 (2.42e-2) ≈	2.5615e-1 (1.52e-2) +	2.5801e-1 (1.53e-2) +	2.6510e-1 (1.10e-2) +	2.6064e-1 (1.34e-2) +	2.6302e-1 (9.98e-3) +	2.4899e-1 (1.36e-2)
WFG4	3	72	4.4421e-1 (9.81e-3) +	4.7121e-1 (1.39e-2) +	4.4547e-1 (8.12e-3) +	4.4210e-1 (8.31e-3) +	4.4180e-1 (7.65e-3) +	4.4037e-1 (1.08e-2) +	4.4718e-1 (7.08e-3) +	4.4580e-1 (6.24e-3) +	4.3377e-1 (9.44e-3)
WFG5	3	72	4.0629e-1 (1.69e-2) +	4.7559e-1 (6.62e-3) +	4.5955e-1 (2.16e-2) +	4.2691e-1 (2.38e-2) +	4.6844e-1 (1.70e-2) +	3.8837e-1 (6.00e-2) ≈	4.4148e-1 (1.76e-2) +	4.1797e-1 (2.66e-2) +	3.9496e-1 (1.49e-2)
WFG6	3	72	4.0836e-1 (1.36e-2) +	4.4357e-1 (1.05e-2) +	4.1976e-1 (3.09e-2) +	4.0787e-1 (2.33e-2) +	3.9401e-1 (1.99e-2) +	3.8948e-1 (2.39e-2) +	4.0851e-1 (1.58e-2) +	4.0810e-1 (1.84e-2) +	3.7131e-1 (1.27e-2)
WFG7	3	72	4.1092e-1 (1.69e-2) +	4.0884e-1 (3.69e-2) +	3.9982e-1 (2.01e-2) +	4.1190e-1 (2.20e-2) +	3.9042e-1 (1.82e-2) +	4.0611e-1 (1.88e-2) +	4.0905e-1 (9.22e-3) +	4.1002e-1 (1.63e-2) +	3.3687e-1 (1.55e-2)
WFG8	3	72	3.6211e-1 (1.39e-2) −	3.8131e-1 (8.63e-3) −	3.6308e-1 (1.54e-2) −	3.5879e-1 (1.55e-2) −	3.5414e-1 (1.09e-2) −	3.6257e-1 (1.12e-2) −	3.5941e-1 (6.30e-3) −	3.5785e-1 (8.76e-3) −	3.8716e-1 (7.16e-3)
WFG9	3	72	4.4274e-1 (1.01e-2) +	4.5164e-1 (1.37e-2) +	4.3872e-1 (1.37e-2) +	4.3861e-1 (1.27e-2) +	4.3689e-1 (2.21e-2) +	4.3185e-1 (1.76e-2) +	4.3972e-1 (9.04e-3) +	4.4329e-1 (1.09e-2) +	3.3908e-1 (3.20e-2)
+/−/≈			13/3/4	14/3/3	11/3/6	11/4/5	12/3/5	12/3/5	13/3/4	13/3/4	

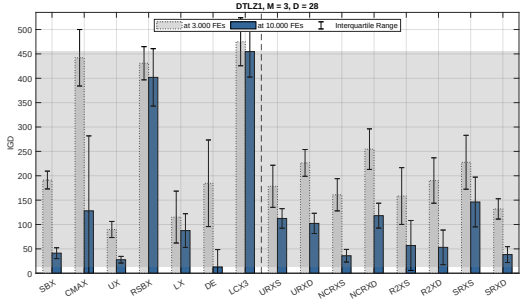
A.1.2 IGD Comparison Plots



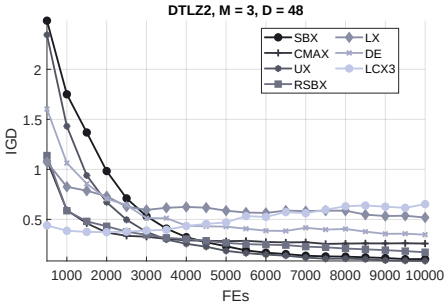
(a) IGD of NSGA-II with single crossover operators.



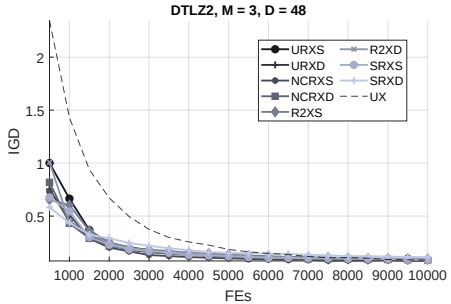
(b) IGD of NSGA-II with hyper-heuristic crossover operators compared to DE+NSGA-II.



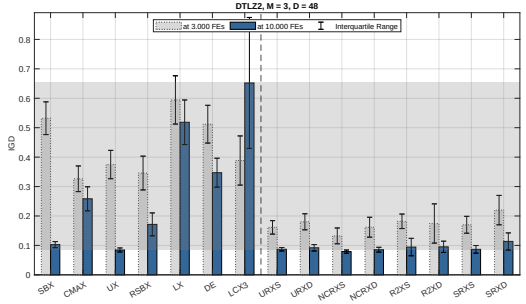
(c) IGD after 3000 FEs and after 10.000 FEs.



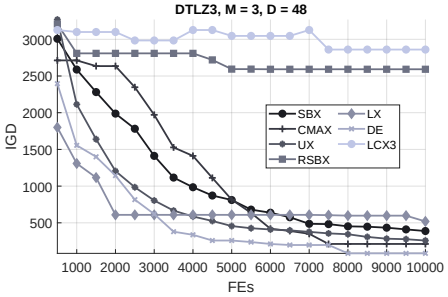
(d) IGD of NSGA-II with single crossover operators.



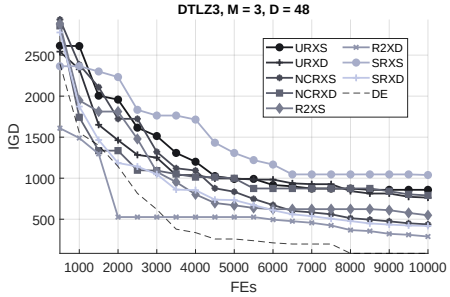
(e) IGD of NSGA-II with hyper-heuristic crossover operators compared to UX+NSGA-II.



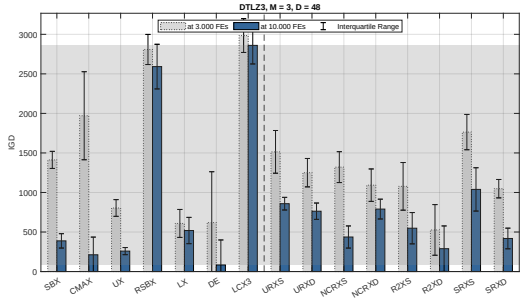
(f) IGD after 3000 FEs and after 10.000 FEs.



(g) IGD of NSGA-II with single crossover operators.

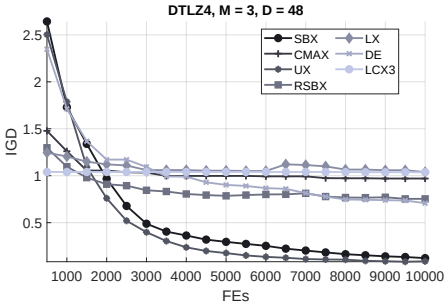


(h) IGD of NSGA-II with hyper-heuristic crossover operators compared to DE+NSGA-II.

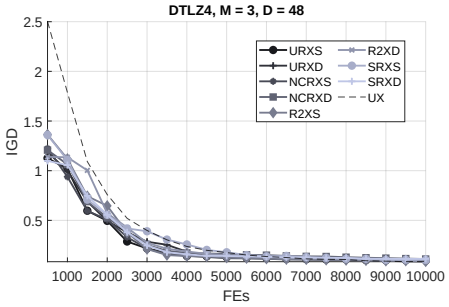


(i) IGD after 3000 FEs and after 10.000 FEs.

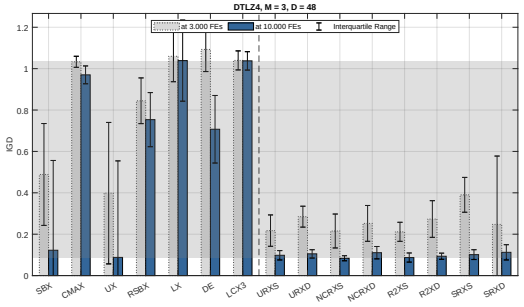
Figure A.1: IGD measured on DTLZ1-3 Benchmark problems. The number of the decision variables was multiplied by 4 to increase difficulty.



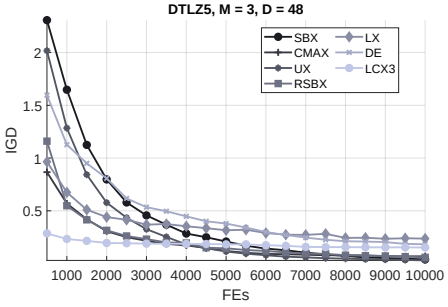
(a) IGD of NSGA-II with single crossover operators.



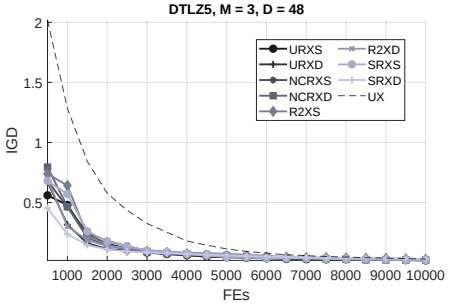
(b) IGD of NSGA-II with hyper-heuristic crossover operators compared to UX+NSGA-II



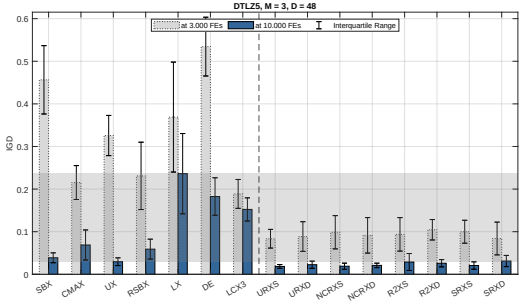
(c) IGD after 3000 FEs and after 10.000 FEs.



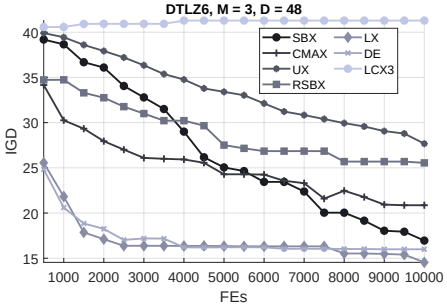
(d) IGD of NSGA-II with single crossover operators.



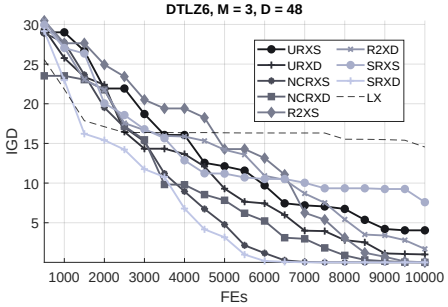
(e) IGD of NSGA-II with hyper-heuristic crossover operators compared to UX+NSGA-II



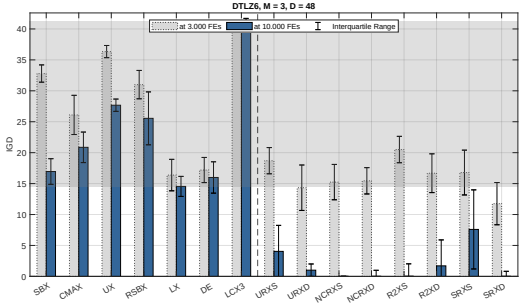
(f) IGD after 3000 FEs and after 10.000 FEs.



(g) IGD of NSGA-II with single crossover operators.

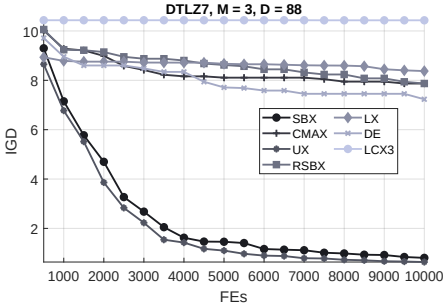


(h) IGD of NSGA-II with hyper-heuristic crossover operators compared to UX+NSGA-II

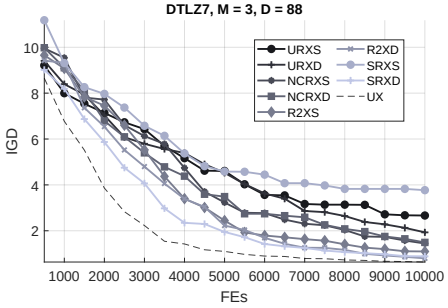


(i) IGD after 3000 FEs and after 10.000 FEs.

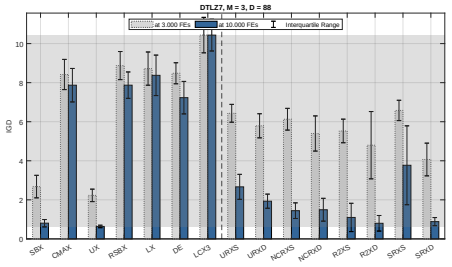
Figure A.2: IGD measured on DTLZ5-6 Benchmark problems. The number of the decision variables was multiplied by 4 to increase difficulty.



(a) IGD of NSGA-II with single crossover operators.

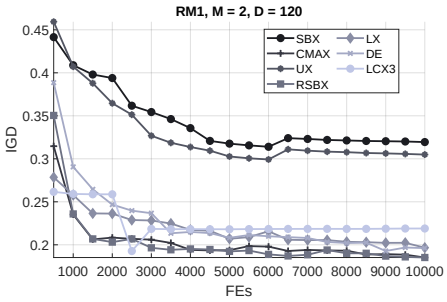


(b) IGD of NSGA-II with hyper-heuristic crossover operators compared to UX+NSGA-II.

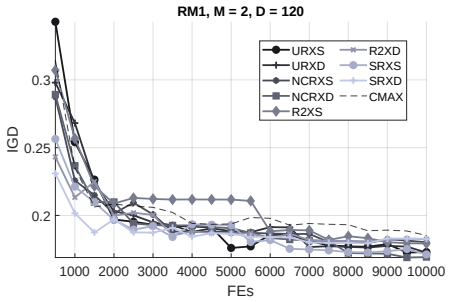


(c) IGD after 3000 FEs and after 10.000 FEs.

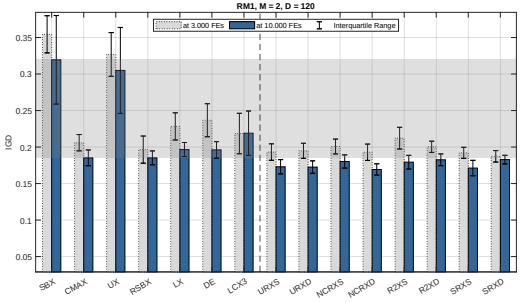
Figure A.3: IGD measured on DTLZ7 Benchmark problems. The number of the decision variables was multiplied by 4 to increase difficulty.



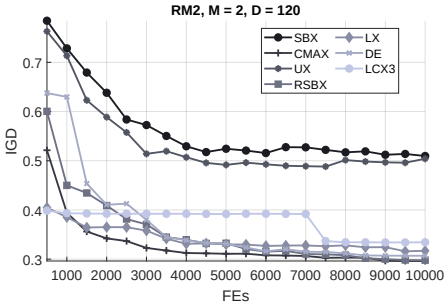
(a) IGD of NSGA-II with single crossover operators.



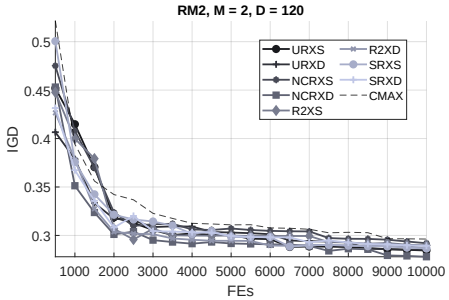
(b) IGD of NSGA-II with hyper-heuristic crossover operators compared to CMAX+NSGA-II.



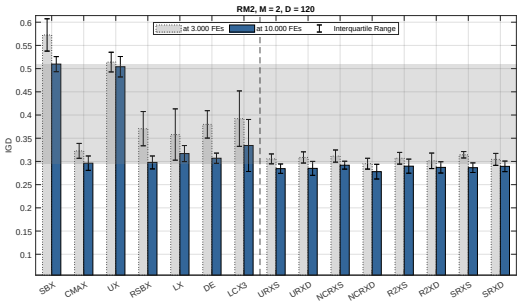
(c) IGD after 3000 FEs and after 10.000 FEs.



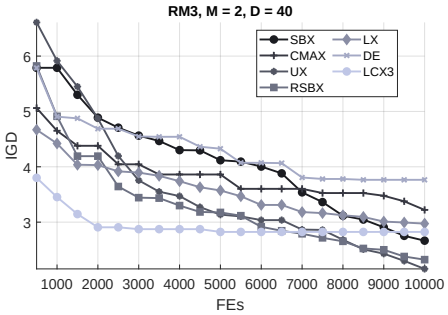
(d) IGD of NSGA-II with single crossover operators.



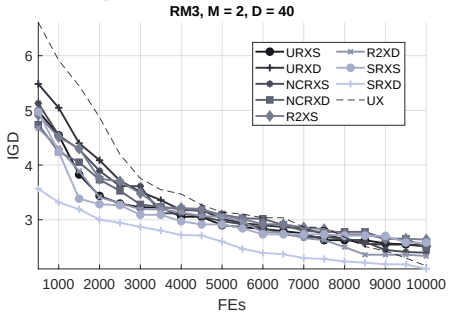
(e) IGD of NSGA-II with hyper-heuristic crossover operators compared to CMAX+NSGA-II.



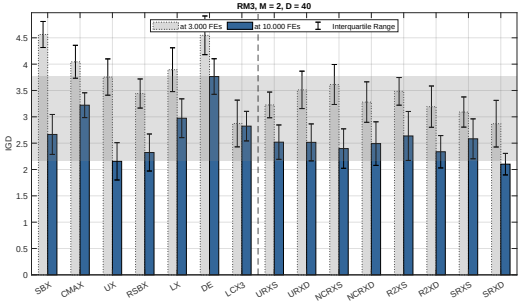
(f) IGD after 3000 FEs and after 10.000 FEs.



(g) IGD of NSGA-II with single crossover operators.

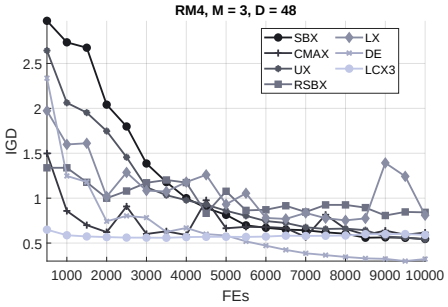


(h) IGD of NSGA-II with hyper-heuristic crossover operators compared to UX+NSGA-II.

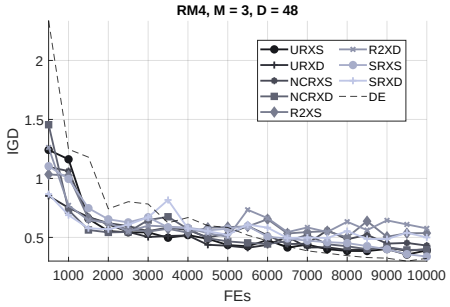


(i) IGD after 3000 FEs and after 10.000 FEs.

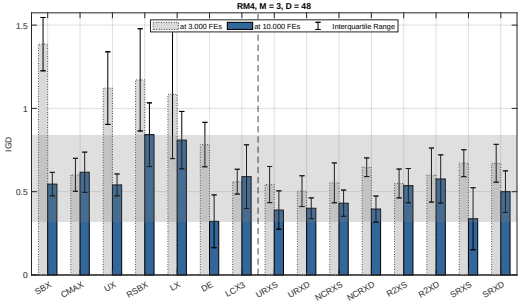
Figure A.4: IGD measured on RM1-3 Benchmark problems. The number of the decision variables was multiplied by 4 to increase difficulty.



(a) IGD of NSGA-II with single crossover operators.

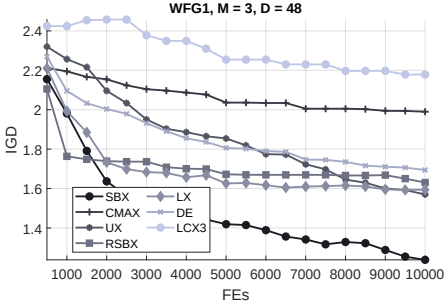


(b) IGD of NSGA-II with hyper-heuristic crossover operators compared to DE+NSGA-II.

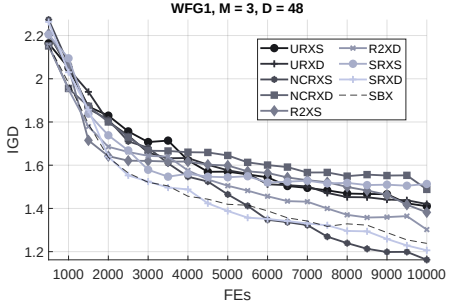


(c) IGD after 3000 FEs and after 10.000 FEs.

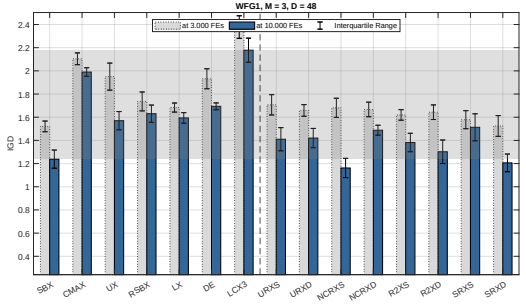
Figure A.5: IGD measured on RM4 Benchmark problems. The number of the decision variables was multiplied by 4 to increase difficulty.



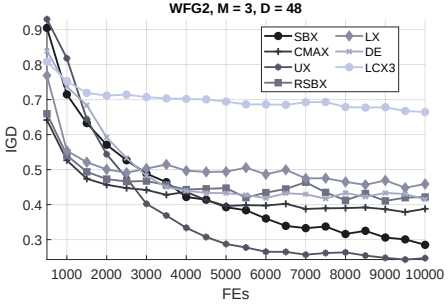
(a) IGD of NSGA-II with single crossover operators.



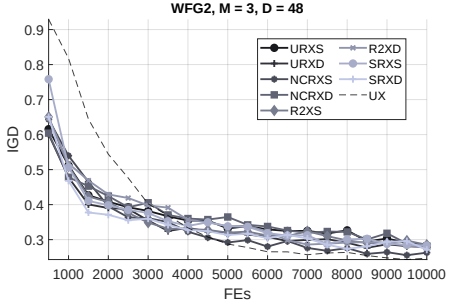
(b) IGD of NSGA-II with hyper-heuristic crossover operators compared to SBX+NSGA-II.



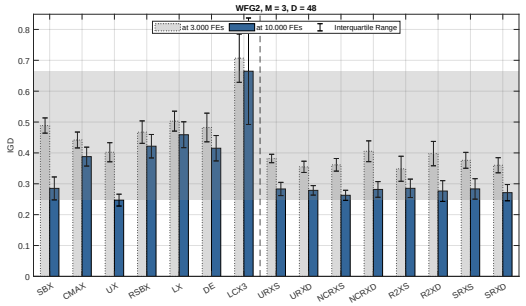
(c) IGD after 3000 FEs and after 10.000 FEs.



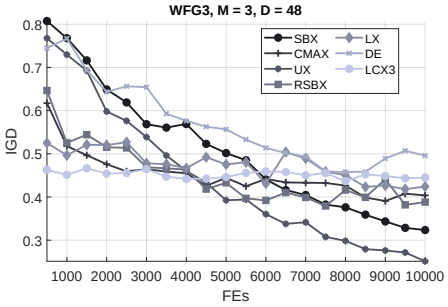
(d) IGD of NSGA-II with single crossover operators.



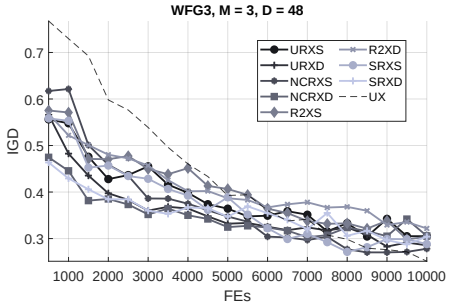
(e) IGD of NSGA-II with hyper-heuristic crossover operators compared to UX+NSGA-II.



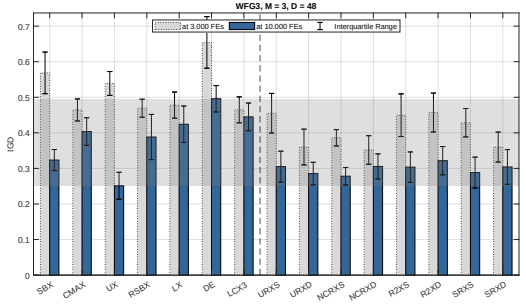
(f) IGD after 3000 FEs and after 10.000 FEs.



(g) IGD of NSGA-II with single crossover operators.

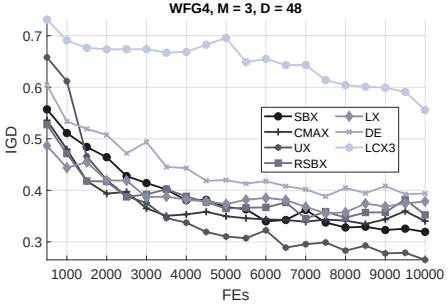


(h) IGD of NSGA-II with hyper-heuristic crossover operators compared to UX+NSGA-II.

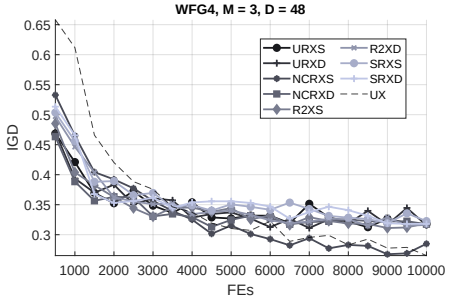


(i) IGD after 3000 FEs and after 10.000 FEs.

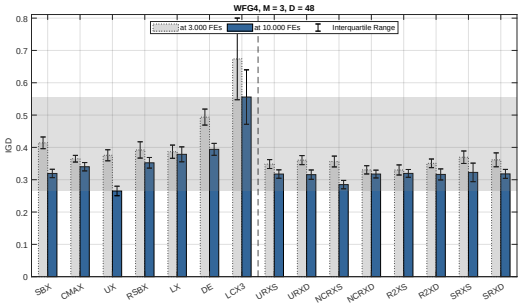
Figure A.6: IGD measured on WFG1-3 Benchmark problems. The number of the decision variables was multiplied by 4 to increase difficulty.



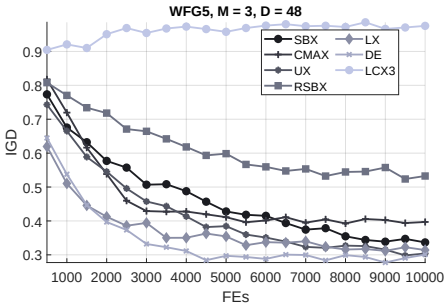
(a) IGD of NSGA-II with single crossover operators.



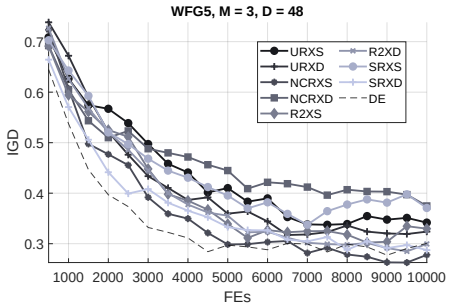
(b) IGD of NSGA-II with hyper-heuristic crossover operators compared to UX+NSGA-II.



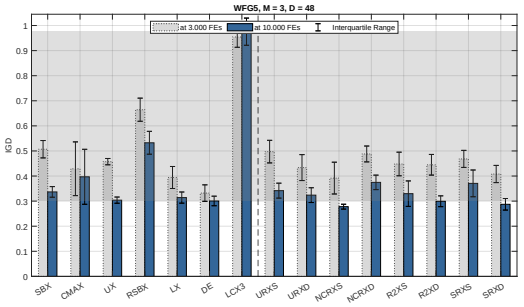
(c) IGD after 3000 FEs and after 10.000 FEs.



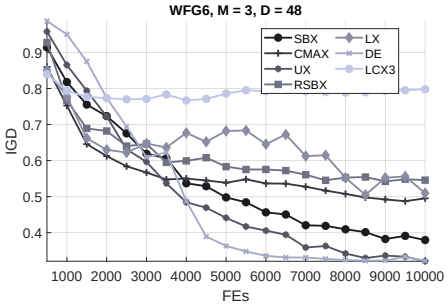
(d) IGD of NSGA-II with single crossover operators.



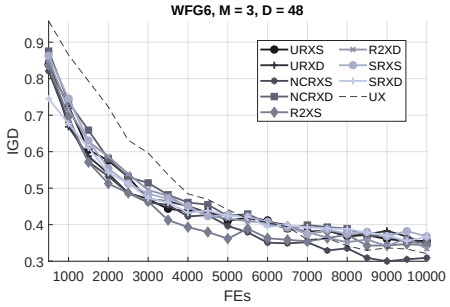
(e) IGD of NSGA-II with hyper-heuristic crossover operators compared to DE+NSGA-II.



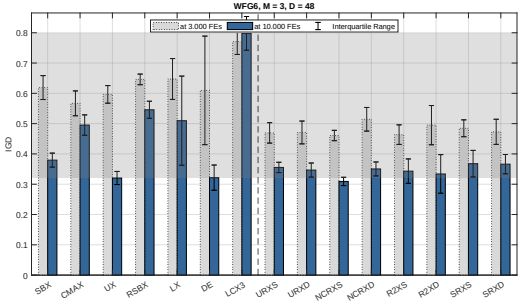
(f) IGD after 3000 FEs and after 10.000 FEs.



(g) IGD of NSGA-II with single crossover operators.

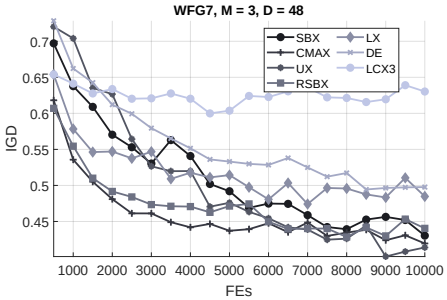


(h) IGD of NSGA-II with hyper-heuristic crossover operators compared to UX+NSGA-II.

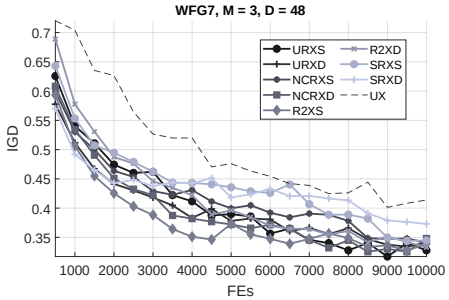


(i) IGD after 3000 FEs and after 10.000 FEs.

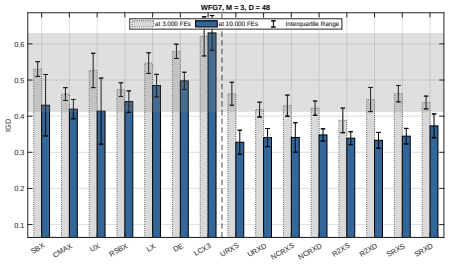
Figure A.7: IGD measured on WFG4-6 Benchmark problems. The number of the decision variables was multiplied by 4 to increase difficulty.



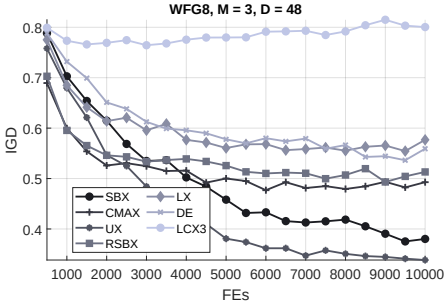
(a) IGD of NSGA-II with single crossover operators.



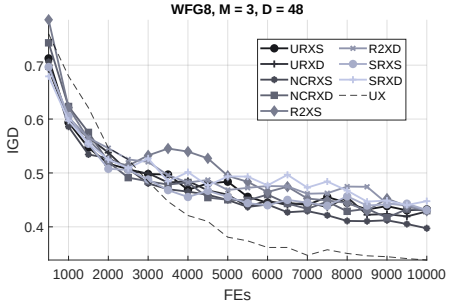
(b) IGD of NSGA-II with hyper-heuristic crossover operators compared to UX+NSGA-II



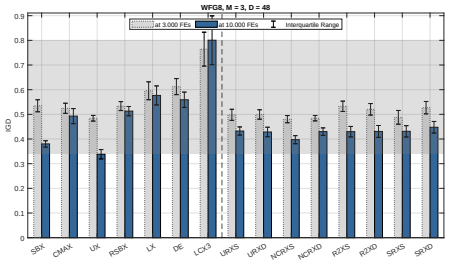
(c) IGD after 3000 FEs and after 10.000 FEs.



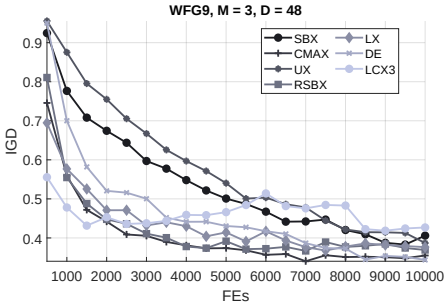
(d) IGD of NSGA-II with single crossover operators.



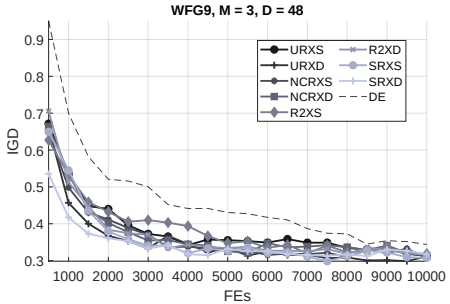
(e) IGD of NSGA-II with hyper-heuristic crossover operators compared to UX+NSGA-II



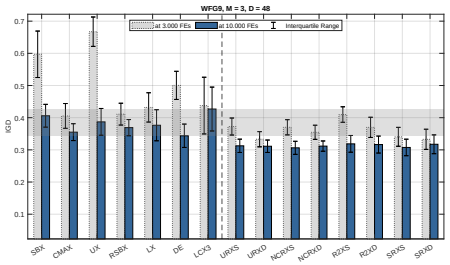
(f) IGD after 3000 FEs and after 10.000 FEs.



(g) IGD of NSGA-II with single crossover operators.



(h) IGD of NSGA-II with hyper-heuristic crossover operators compared to DE+NSGA-II



(i) IGD after 3000 FEs and after 10.000 FEs.

Figure A.8: IGD measured on WFG7-9 Benchmark. The number of the decision variables was multiplied by 4 to increase difficulty.

A.1.3 Selection probability or Distribution over FEs

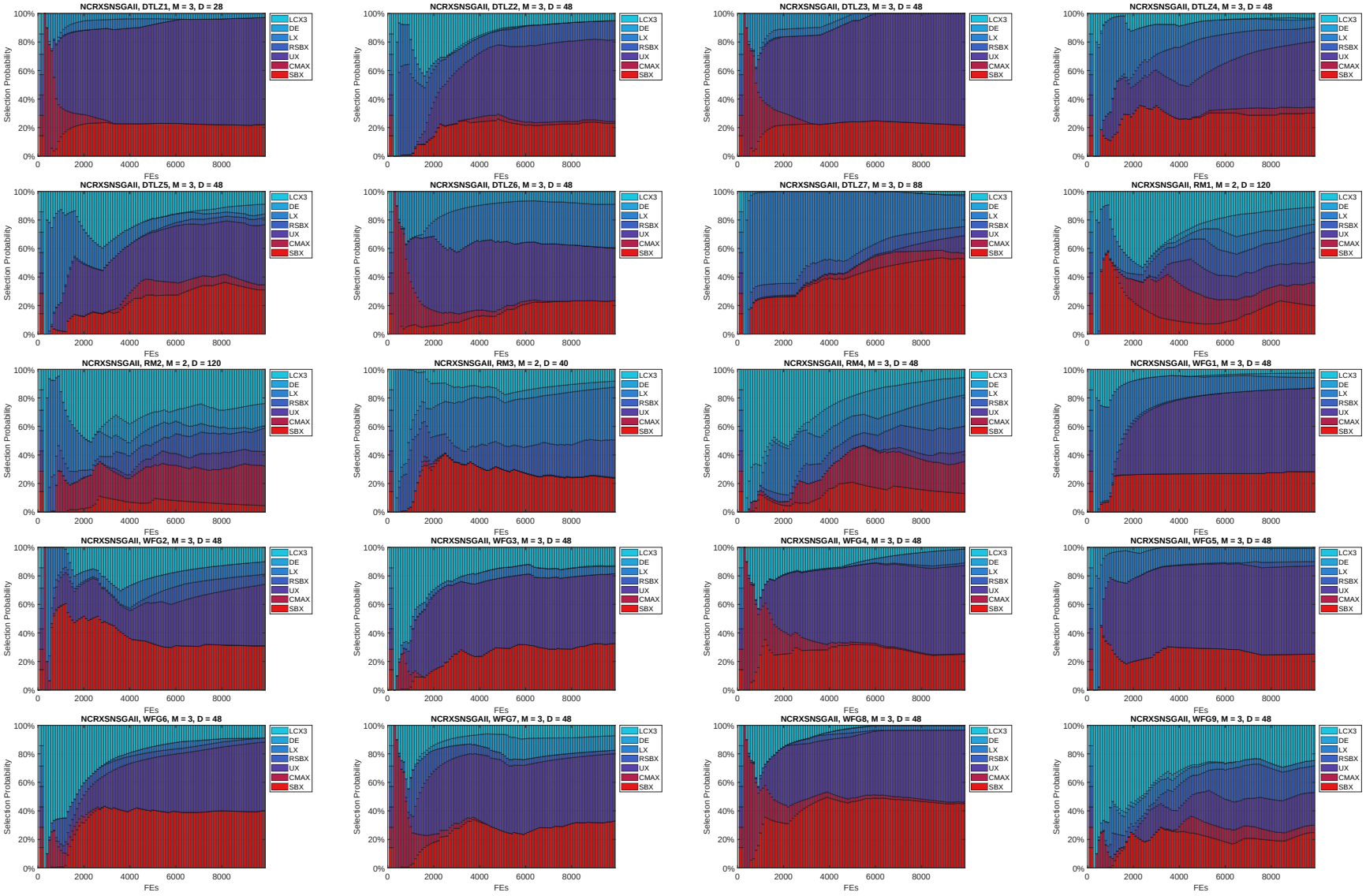


Figure A.9: Trend of selection probability over FEs of NCRXSNSGA-II on DTLZ1-7, RM1-4 and WFG1-9 with D multiplied by 4.

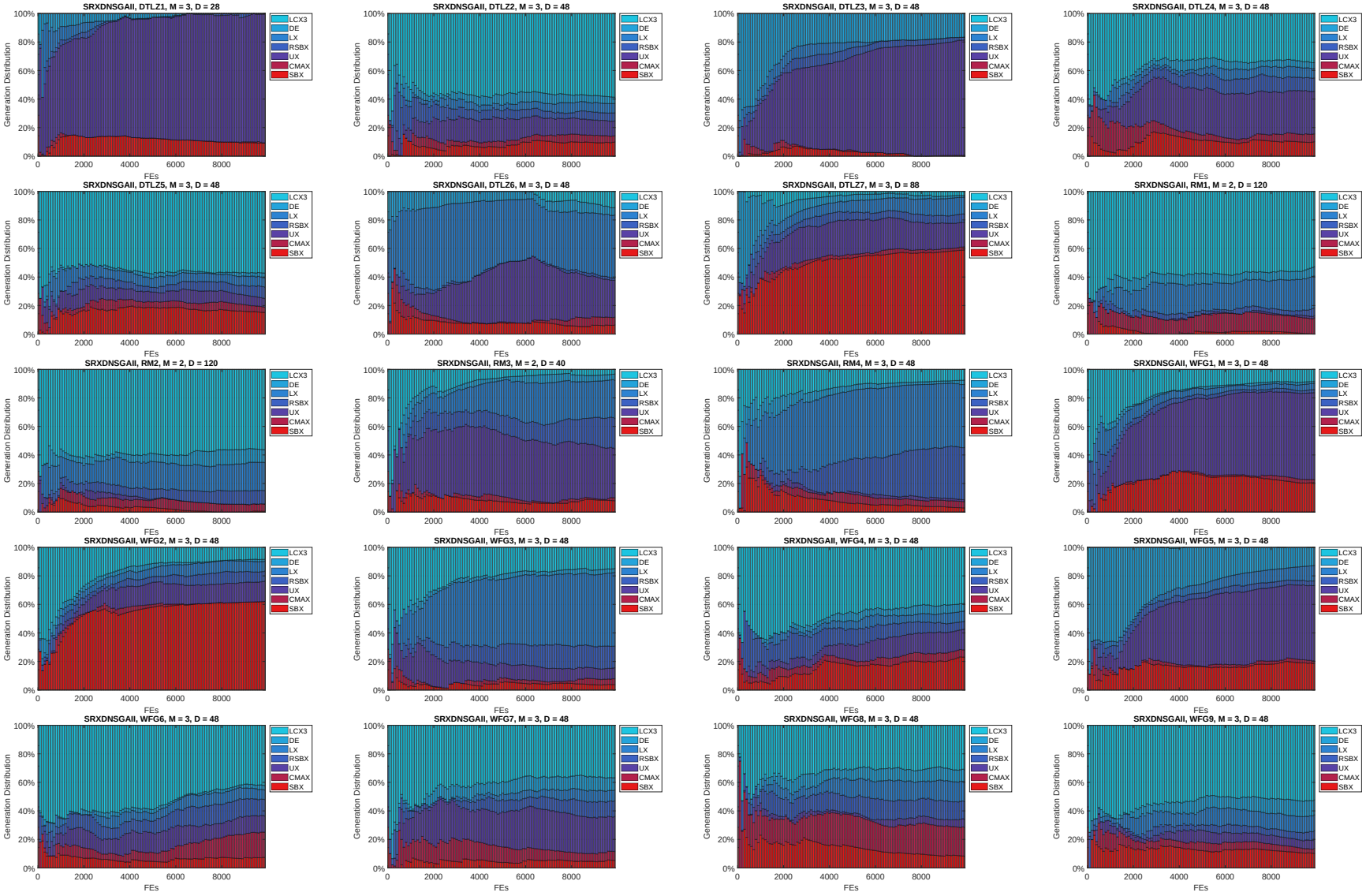


Figure A.10: Trend of Distribution over FEs of SRXDNSGA-II on DTLZ1-7, RM1-4 and WFG1-9 with D multiplied by 4.

A.1.4 Cumulative Number of Offspring

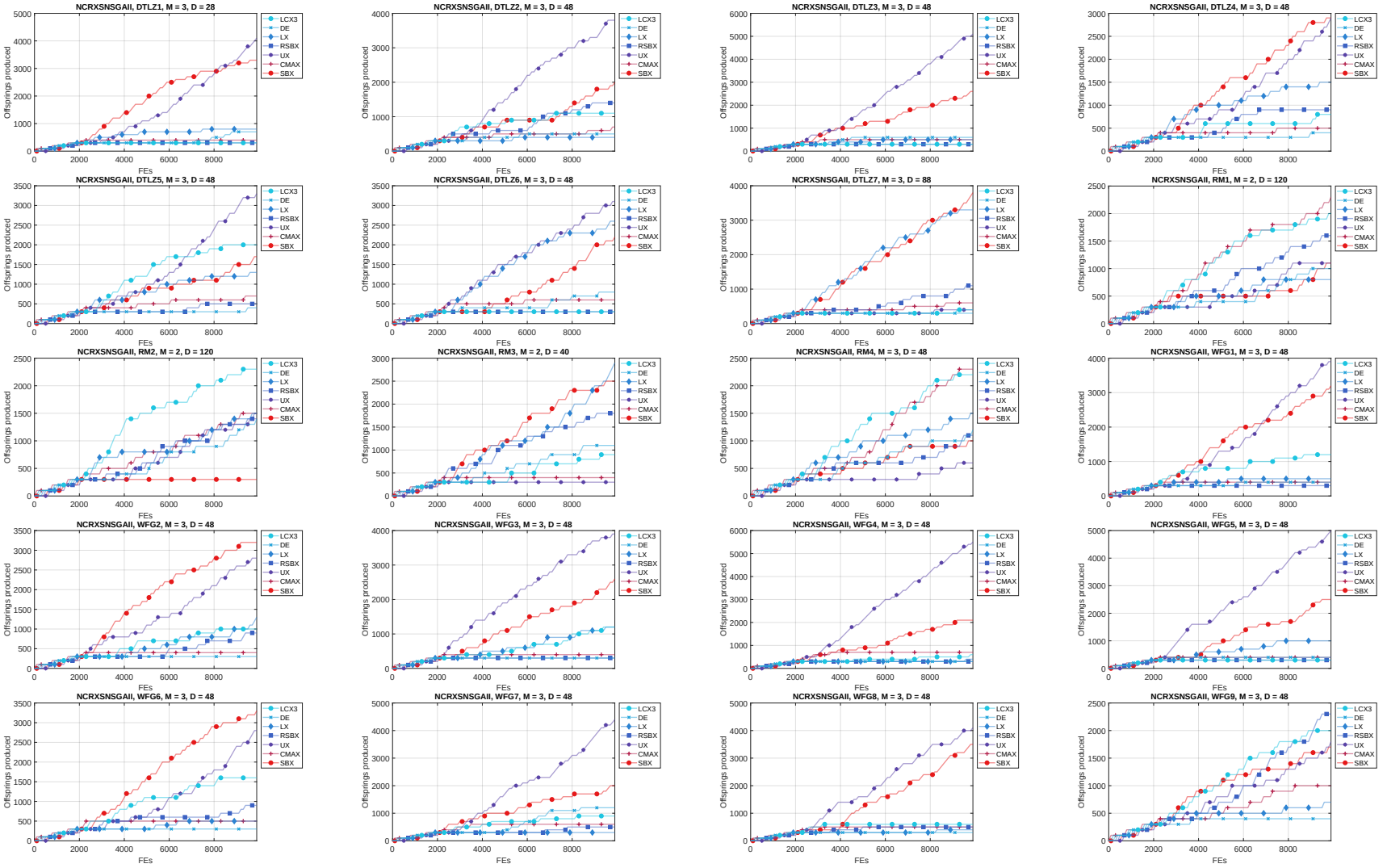


Figure A.11: Cumulative number of offspring generated by the different crossover operators in NCRXSNSGA-II on DTLZ1-7, RM1-4 and WFG1-9 with D multiplied by 4.

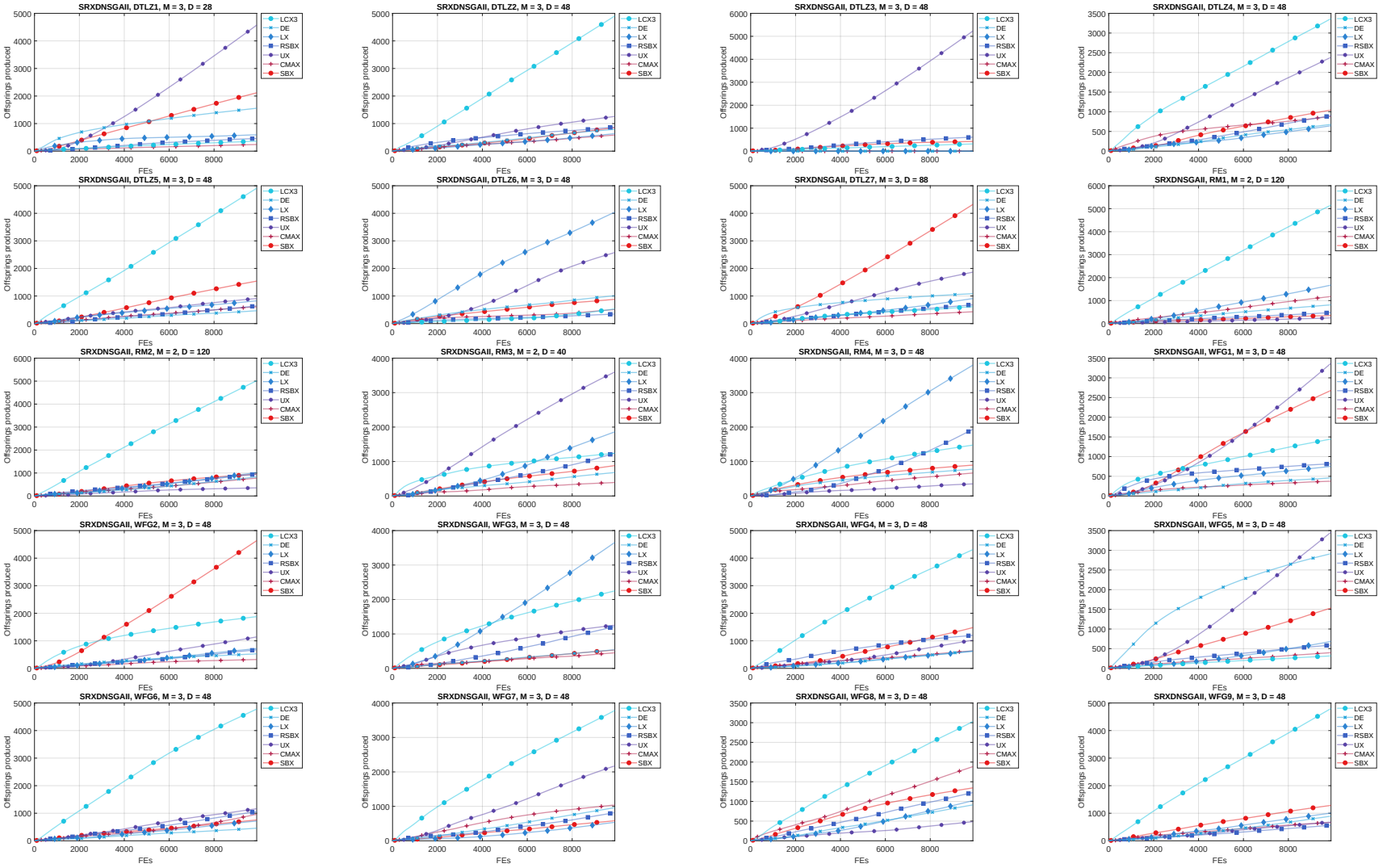


Figure A.12: Cumulative number of offspring generated by the different crossover operators in SRXDNSGA-II on DTLZ1-7, RM1-4 and WFG1-9 with D multiplied by 4.

A.1.5 Hyper Volume of Tuned-NCRXS and Tuned-SRXD

Table A.7: HV of Tuned NCRXS+NSGA-II on DTLZ, RM and WFG with a number of decision variables multiplied by 4

Problem	*M*	*D*	LINEAR	LOGISTIC	LOS	XOP16	XOP3	SRXD
DTLZ1	3	28	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ2	3	48	4.4646e-1 (2.22e-2) +	4.4698e-1 (2.51e-2) +	4.3395e-1 (6.34e-2) +	3.7084e-1 (5.81e-2) −	4.6531e-1 (3.64e-2) +	4.0184e-1 (5.41e-2)
DTLZ3	3	48	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ4	3	48	4.2955e-1 (4.97e-2) ≈	4.5863e-1 (3.09e-2) +	4.2764e-1 (6.07e-2) ≈	3.7576e-1 (4.66e-2) −	4.2940e-1 (1.50e-1) ≈	4.0829e-1 (6.69e-2)
DTLZ5	3	48	1.7729e-1 (1.17e-2) +	1.7528e-1 (9.90e-3) +	1.7730e-1 (1.44e-2) ≈	1.5461e-1 (1.86e-2) −	1.8392e-1 (1.25e-2) +	1.7087e-1 (1.49e-2)
DTLZ6	3	48	1.9947e-1 (3.42e-4) ≈	1.9940e-1 (4.40e-4) ≈	1.9523e-1 (1.99e-1) ≈	0.0000e+0 (1.97e-1) −	0.0000e+0 (0.00e+0) −	1.9916e-1 (2.00e-1)
DTLZ7	3	88	6.3967e-3 (1.09e-2) −	2.4989e-2 (2.45e-2) ≈	4.9831e-3 (2.73e-2) −	0.0000e+0 (0.00e+0) −	0.0000e+0 (6.02e-2) ≈	2.5905e-2 (2.08e-2)
RM1	2	120	5.3052e-1 (6.00e-3) +	5.2868e-1 (5.28e-3) +	5.2440e-1 (7.88e-3) ≈	5.2709e-1 (5.28e-3) +	5.3904e-1 (1.45e-2) +	5.2529e-1 (5.90e-3)
RM2	2	120	1.8039e-1 (5.91e-3) ≈	1.7818e-1 (1.06e-2) ≈	1.8178e-1 (8.23e-3) ≈	1.8783e-1 (7.23e-3) +	1.7982e-1 (9.42e-3) ≈	1.7829e-1 (9.28e-3)
RM3	2	40	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
RM4	3	48	1.3108e-1 (4.83e-2) +	1.3028e-1 (7.47e-2) +	1.5222e-1 (9.56e-2) +	8.2424e-2 (5.09e-2) ≈	1.3514e-1 (2.30e-1) +	1.0081e-1 (7.58e-2)
WFG1	3	48	3.7672e-1 (2.39e-2) −	3.9369e-1 (3.14e-2) −	3.4851e-1 (5.70e-2) −	3.0436e-1 (2.75e-2) −	3.8315e-1 (5.46e-2) −	4.0934e-1 (2.49e-2)
WFG2	3	48	8.3776e-1 (1.28e-2) ≈	8.3514e-1 (1.39e-2) ≈	8.3759e-1 (2.96e-2) ≈	8.0827e-1 (1.03e-2) −	8.5362e-1 (5.34e-2) ≈	8.3911e-1 (1.67e-2)
WFG3	3	48	2.8619e-1 (1.76e-2) +	2.8270e-1 (1.45e-2) +	2.8022e-1 (2.90e-2) ≈	2.6600e-1 (1.77e-2) −	3.1722e-1 (5.10e-2) +	2.7687e-1 (1.25e-2)
WFG4	3	48	4.5319e-1 (8.44e-3) +	4.5667e-1 (7.38e-3) +	4.5368e-1 (1.57e-2) +	4.3741e-1 (1.26e-2) −	4.8463e-1 (3.83e-2) +	4.4936e-1 (1.19e-2)
WFG5	3	48	4.6683e-1 (7.72e-3) −	4.6927e-1 (8.99e-3) −	4.6608e-1 (2.19e-2) −	4.5861e-1 (1.22e-2) −	3.8616e-1 (9.72e-2) −	4.7456e-1 (1.09e-2)
WFG6	3	48	4.2354e-1 (1.84e-2) ≈	4.2984e-1 (2.30e-2) +	4.1972e-1 (4.01e-2) ≈	4.7501e-1 (1.69e-2) +	4.2422e-1 (3.87e-2) ≈	4.1417e-1 (2.08e-2)
WFG7	3	48	4.2980e-1 (1.84e-2) +	4.3281e-1 (2.39e-2) +	4.2516e-1 (2.44e-2) +	4.0226e-1 (1.82e-2) ≈	4.2515e-1 (2.21e-2) +	4.0904e-1 (2.36e-2)
WFG8	3	48	3.6224e-1 (6.74e-3) +	3.6115e-1 (1.48e-2) +	3.6087e-1 (2.02e-2) ≈	3.4783e-1 (1.18e-2) −	3.6540e-1 (3.50e-2) +	3.5587e-1 (1.11e-2)
WFG9	3	48	4.4402e-1 (1.76e-2) ≈	4.4398e-1 (2.02e-2) ≈	4.3697e-1 (1.87e-2) −	4.4254e-1 (1.17e-2) ≈	4.4121e-1 (1.34e-2) ≈	4.4285e-1 (1.41e-2)
+/−/≈			8/3/9	10/2/8	4/4/12	3/11/6	8/3/9	

Table A.8: HV of Tuned NCRXS+NSGA-II on DTLZ, RM and WFG with a number of decision variables multiplied by 4. Part 1.

Problem	M	D	LINEAR	LOGISTIC	MINR0	MINR5	XOP16	XOP3	NCRXS
DTLZ1	3	28	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ2	3	48	4.9036e-1 (1.07e-2) ≈	4.8902e-1 (1.06e-2) ≈	4.9062e-1 (1.29e-2) ≈	4.8685e-1 (1.53e-2) ≈	4.8370e-1 (1.27e-2) ≈	5.0482e-1 (8.93e-3) +	4.8962e-1 (1.47e-2)
DTLZ3	3	48	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ4	3	48	4.6007e-1 (3.35e-2) ≈	4.6570e-1 (3.15e-2) −	4.7244e-1 (2.44e-2) ≈	4.6433e-1 (4.15e-2) −	4.4301e-1 (4.12e-2) −	5.0337e-1 (2.23e-2) +	4.7833e-1 (3.07e-2)
DTLZ5	3	48	1.8282e-1 (5.73e-3) ≈	1.8410e-1 (4.63e-3) ≈	1.8239e-1 (6.58e-3) ≈	1.8284e-1 (4.76e-3) ≈	1.8475e-1 (4.50e-3) ≈	1.9103e-1 (1.66e-3) +	1.8302e-1 (5.96e-3)
DTLZ6	3	48	1.9927e-1 (6.42e-4) ≈	1.9937e-1 (2.98e-3) ≈	1.9908e-1 (7.73e-4) ≈	1.9906e-1 (1.04e-3) ≈	1.9937e-1 (2.00e-1) ≈	0.0000e+0 (0.00e+0) −	1.9928e-1 (7.06e-4)
DTLZ7	3	88	3.0214e-3 (4.54e-3) ≈	2.8570e-3 (3.13e-3) −	4.6552e-3 (5.64e-3) ≈	3.5055e-3 (3.76e-3) ≈	0.0000e+0 (0.00e+0) −	7.7907e-3 (7.48e-3) +	3.6198e-3 (4.47e-3)
RM1	2	120	5.3207e-1 (7.62e-3) ≈	5.3338e-1 (6.84e-3) ≈	5.3161e-1 (5.26e-3) ≈	5.3278e-1 (6.00e-3) ≈	5.3564e-1 (3.33e-3) +	5.3523e-1 (6.34e-3) +	5.3185e-1 (7.75e-3)
RM2	2	120	1.7654e-1 (8.60e-3) ≈	1.7641e-1 (1.15e-2) ≈	1.7666e-1 (1.18e-2) ≈	1.7408e-1 (8.03e-3) ≈	1.9938e-1 (5.48e-3) +	1.7242e-1 (6.30e-3) ≈	1.7513e-1 (9.32e-3)
RM3	2	40	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
RM4	3	48	1.2206e-1 (5.49e-2) ≈	1.3994e-1 (5.31e-2) ≈	1.2938e-1 (8.66e-2) ≈	1.5306e-1 (8.55e-2) ≈	1.0599e-1 (4.84e-2) −	1.4920e-1 (8.46e-2) ≈	1.3420e-1 (4.54e-2)
WFG1	3	48	3.7840e-1 (2.90e-2) −	3.8360e-1 (2.91e-2) −	4.0909e-1 (2.02e-2) ≈	3.9249e-1 (2.27e-2) −	3.5407e-1 (2.68e-2) −	3.5897e-1 (8.20e-2) −	4.1605e-1 (2.76e-2)
WFG2	3	48	8.4791e-1 (1.67e-2) ≈	8.4318e-1 (7.87e-3) −	8.4879e-1 (1.31e-2) ≈	8.4645e-1 (1.11e-2) ≈	8.2477e-1 (9.24e-3) −	8.7362e-1 (6.70e-3) +	8.4766e-1 (6.88e-3)
WFG3	3	48	2.9533e-1 (1.58e-2) ≈	2.9214e-1 (1.17e-2) ≈	3.0037e-1 (1.58e-2) ≈	2.9754e-1 (1.81e-2) ≈	2.8138e-1 (1.79e-2) −	3.1445e-1 (9.45e-3) +	2.9505e-1 (1.10e-2)
WFG4	3	48	4.7482e-1 (1.00e-2) −	4.6633e-1 (1.37e-2) −	4.8313e-1 (6.97e-3) ≈	4.7047e-1 (8.61e-3) −	4.6598e-1 (8.59e-3) −	4.9238e-1 (1.08e-2) +	4.8656e-1 (1.33e-2)
WFG5	3	48	4.7530e-1 (8.25e-3) −	4.7263e-1 (1.13e-2) −	4.8106e-1 (8.63e-3) ≈	4.6907e-1 (1.06e-2) −	4.7999e-1 (5.93e-3) −	4.6557e-1 (1.08e-2) −	4.8315e-1 (8.68e-3)
WFG6	3	48	4.5438e-1 (1.28e-2) −	4.4597e-1 (1.80e-2) −	4.6293e-1 (1.05e-2) ≈	4.5046e-1 (1.20e-2) −	4.5038e-1 (1.32e-2) −	4.4844e-1 (1.49e-2) −	4.6203e-1 (9.63e-3)
WFG7	3	48	4.4311e-1 (2.35e-2) ≈	4.4037e-1 (1.94e-2) ≈	4.4571e-1 (1.67e-2) ≈	4.4652e-1 (1.98e-2) ≈	4.2101e-1 (2.05e-2) −	4.5470e-1 (1.06e-2) +	4.3777e-1 (2.41e-2)
WFG8	3	48	3.8205e-1 (1.01e-2) −	3.7957e-1 (1.33e-2) −	3.8694e-1 (8.76e-3) ≈	3.8357e-1 (1.00e-2) −	3.6685e-1 (8.71e-3) −	4.0261e-1 (9.49e-3) +	3.8769e-1 (1.13e-2)
WFG9	3	48	4.5096e-1 (1.16e-2) ≈	4.4822e-1 (1.19e-2) ≈	4.5405e-1 (9.90e-3) ≈	4.5224e-1 (1.53e-2) ≈	4.5644e-1 (1.38e-2) ≈	4.5049e-1 (1.06e-2) ≈	4.5037e-1 (1.03e-2)
+/−/≈			0/5/15	0/8/12	0/0/20	0/6/14	2/11/7	10/4/6	

Table A.9: HV of Tuned NCRXS+NSGA-II on DTLZ, RM and WFG with a number of decision variables multiplied by 4. Part 2.

Problem	M	D	EP5	EP7	RR10	RR5	NCRXS
DTLZ1	3	28	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ2	3	48	4.9603e-1 (1.71e-2) +	4.9354e-1 (1.28e-2) ≈	4.8932e-1 (9.16e-3) ≈	4.8843e-1 (1.74e-2) ≈	4.8962e-1 (1.47e-2)
DTLZ3	3	48	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
DTLZ4	3	48	4.7597e-1 (2.42e-2) ≈	4.6583e-1 (2.44e-2) ≈	4.7143e-1 (2.04e-2) ≈	4.7336e-1 (3.40e-2) ≈	4.7833e-1 (3.07e-2)
DTLZ5	3	48	1.8184e-1 (6.12e-3) ≈	1.8306e-1 (5.24e-3) ≈	1.8040e-1 (8.11e-3) −	1.8161e-1 (6.26e-3) ≈	1.8302e-1 (5.96e-3)
DTLZ6	3	48	1.9920e-1 (9.60e-4) ≈	1.9855e-1 (1.99e-1) −	1.9934e-1 (5.34e-4) ≈	1.9931e-1 (7.14e-4) ≈	1.9928e-1 (7.06e-4)
DTLZ7	3	88	2.8594e-3 (2.03e-3) ≈	1.8613e-3 (3.74e-3) −	2.7250e-3 (4.69e-3) ≈	3.0853e-3 (5.21e-3) ≈	3.6198e-3 (4.47e-3)
RM1	2	120	5.3345e-1 (5.40e-3) ≈	5.3370e-1 (4.62e-3) ≈	5.2968e-1 (6.05e-3) ≈	5.3086e-1 (5.69e-3) ≈	5.3185e-1 (7.75e-3)
RM2	2	120	1.7270e-1 (1.09e-2) ≈	1.7736e-1 (8.91e-3) ≈	1.7257e-1 (1.24e-2) ≈	1.7331e-1 (8.44e-3) ≈	1.7513e-1 (9.32e-3)
RM3	2	40	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0) ≈	0.0000e+0 (0.00e+0)
RM4	3	48	1.3442e-1 (5.64e-2) ≈	1.5565e-1 (7.64e-2) +	1.1624e-1 (8.83e-2) ≈	1.2582e-1 (7.44e-2) ≈	1.3420e-1 (4.54e-2)
WFG1	3	48	3.9902e-1 (2.25e-2) −	3.8910e-1 (3.95e-2) −	4.1785e-1 (3.20e-2) ≈	4.1512e-1 (3.15e-2) ≈	4.1605e-1 (2.76e-2)
WFG2	3	48	8.4723e-1 (9.22e-3) ≈	8.4373e-1 (1.10e-2) −	8.4751e-1 (1.50e-2) ≈	8.4949e-1 (1.30e-2) ≈	8.4766e-1 (6.88e-3)
WFG3	3	48	2.9698e-1 (1.02e-2) ≈	2.9612e-1 (1.48e-2) ≈	3.0032e-1 (1.61e-2) ≈	3.0044e-1 (1.46e-2) ≈	2.9505e-1 (1.10e-2)
WFG4	3	48	4.8163e-1 (1.00e-2) −	4.7894e-1 (1.27e-2) −	4.8231e-1 (1.15e-2) ≈	4.8521e-1 (8.40e-3) ≈	4.8656e-1 (1.33e-2)
WFG5	3	48	4.7771e-1 (8.77e-3) −	4.7717e-1 (8.57e-3) −	4.7969e-1 (8.14e-3) ≈	4.7839e-1 (7.05e-3) −	4.8315e-1 (8.68e-3)
WFG6	3	48	4.5787e-1 (1.50e-2) ≈	4.5462e-1 (1.63e-2) −	4.6130e-1 (1.36e-2) ≈	4.6239e-1 (1.08e-2) ≈	4.6203e-1 (9.63e-3)
WFG7	3	48	4.5280e-1 (1.24e-2) +	4.4345e-1 (1.44e-2) ≈	4.3687e-1 (2.64e-2) ≈	4.4321e-1 (2.20e-2) ≈	4.3777e-1 (2.41e-2)
WFG8	3	48	3.8356e-1 (9.06e-3) −	3.8112e-1 (1.03e-2) −	3.9111e-1 (9.13e-3) ≈	3.9032e-1 (1.25e-2) ≈	3.8769e-1 (1.13e-2)
WFG9	3	48	4.5338e-1 (1.25e-2) ≈	4.5675e-1 (1.44e-2) ≈	4.5271e-1 (1.37e-2) ≈	4.5185e-1 (1.30e-2) ≈	4.5037e-1 (1.03e-2)
+/−/≈			2/4/14	1/8/11	0/1/19	0/1/19	

A.1.6 Hyper Volume of NCRXS Compared to NSGA-II with Single Crossover operators

Table A.10: Hyper Volume of NCRXD with only three operators compared to the single crossover operators on DTLZ, RM and WFG with a number of decision variables multiplied by 4.

Problem	*M*	*D*	CMAX	DE	LCX3	LX	RSBX	UX	SBX	XOP3_SRXD	XOP3_NCRXS
DTLZ1	3	28	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (2.49e-1) −/−	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0)o/≈	0.0000e+0 (0.00e+0) ≈/o
DTLZ2	3	48	2.2751e-1 (3.78e-2) −/−	1.4124e-1 (4.11e-2) −/−	1.9131e-2 (5.39e-2) −/−	5.4452e-2 (2.38e-2) −/−	3.2255e-1 (4.64e-2) −/−	4.8652e-1 (2.15e-2) +/−	4.3879e-1 (2.48e-2) −/−	4.6531e-1 (3.64e-2) o/−	5.0482e-1 (8.93e-3)+/o
DTLZ3	3	48	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) −/−	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) o/≈	0.0000e+0 (0.00e+0) ≈/o
DTLZ4	3	48	1.9633e-2 (1.33e-2) −/−	0.0000e+0 (2.15e-2) −/−	0.0000e+0 (0.00e+0) −/−	0.0000e+0 (0.00e+0) −/−	1.2997e-2 (2.92e-2) −/−	4.7841e-1 (1.70e-1) ≈/−	4.0042e-1 (9.71e-2) −/−	4.2940e-1 (1.50e-1) o/−	5.0337e-1 (2.23e-2) +/o
DTLZ5	3	48	1.2220e-1 (2.44e-2) −/−	5.3526e-2 (1.65e-2) −/−	8.0797e-2 (2.14e-2) −/−	5.0582e-2 (3.31e-2) −/−	1.4868e-1 (1.89e-2) −/−	1.7796e-1 (6.01e-3) −/−	1.6658e-1 (1.08e-2) −/−	1.8392e-1 (1.25e-2) o/−	1.9103e-1 (1.66e-3) +/o
DTLZ6	3	48	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) o/≈	0.0000e+0 (0.00e+0) ≈/≈
DTLZ7	3	88	0.0000e+0 (0.00e+0) −/−	0.0000e+0 (0.00e+0) −/−	0.0000e+0 (0.00e+0) −/−	0.0000e+0 (0.00e+0) −/−	0.0000e+0 (0.00e+0) −/−	6.1756e-2 (1.05e-2) +/+	3.3299e-2 (2.24e-2) ≈/+	0.0000e+0 (6.02e-2) o/≈	7.7907e-3 (7.48e-3)≈/o
RM1	2	120	5.2395e-1 (6.04e-3) −/−	5.1241e-1 (1.07e-2) −/−	4.9330e-1 (1.69e-2) −/−	5.0801e-1 (9.11e-3) −/−	5.2135e-1 (9.02e-3) −/−	4.6872e-1 (1.16e-2) −/−	4.5845e-1 (1.63e-2) −/−	5.3904e-1 (1.45e-2) o/≈	5.3523e-1 (6.34e-3) ≈/o
RM2	2	120	1.7385e-1 (1.03e-2) ≈/+	1.6096e-1 (7.30e-3) −/−	1.5318e-1 (4.52e-2) −/−	1.5624e-1 (1.69e-2) −/−	1.7386e-1 (1.21e-2) ≈/≈	6.1698e-2 (7.63e-3) −/−	5.1375e-2 (7.79e-3) −/−	1.7982e-1 (9.42e-3) o/+	1.7242e-1 (6.30e-3) −/o
RM3	2	40	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) ≈/≈	0.0000e+0 (0.00e+0) o/≈	0.0000e+0 (0.00e+0) ≈/o
RM4	3	48	5.0447e-2 (5.72e-2) −/−	2.1318e-1 (1.37e-1) +/+	6.7406e-2 (7.80e-2) −/−	6.3786e-3 (1.97e-2) −/−	0.0000e+0 (1.97e-3) −/−	2.4855e-2 (3.05e-2) −/−	1.9093e-2 (2.64e-2) −/−	1.3514e-1 (2.30e-1) o/≈	1.4920e-1 (8.46e-2) ≈/o
WFG1	3	48	5.4083e-2 (2.00e-2) −/−	1.7806e-1 (1.65e-2) −/−	0.0000e+0 (0.00e+0) −/−	2.7111e-1 (1.80e-2) −/−	2.1870e-1 (2.52e-2) −/−	1.9146e-1 (3.85e-2) −/−	3.9386e-1 (2.83e-2) ≈/+	3.8315e-1 (5.46e-2) o/+	3.5897e-1 (8.20e-2) −/o
WFG2	3	48	7.5886e-1 (1.71e-2) −/−	7.3862e-1 (1.37e-2) −/−	6.4959e-1 (4.75e-2) −/−	7.2805e-1 (1.64e-2) −/−	7.4904e-1 (1.90e-2) −/−	8.5798e-1 (1.61e-2) ≈/−	8.3203e-1 (2.06e-2) ≈/−	8.5362e-1 (5.34e-2) o/−	8.7362e-1 (6.70e-3) +/o
WFG3	3	48	2.4306e-1 (1.65e-2) −/−	2.0323e-1 (2.12e-2) −/−	2.0766e-1 (1.50e-2) −/−	2.3110e-1 (9.41e-3) −/−	2.5069e-1 (1.65e-2) −/−	3.1482e-1 (1.45e-2) ≈/≈	2.8507e-1 (1.42e-2) −/−	3.1722e-1 (5.10e-2) o/≈	3.1445e-1 (9.45e-3) ≈/o
WFG4	3	48	4.2864e-1 (9.62e-3) −/−	4.0003e-1 (9.04e-3) −/−	3.4575e-1 (2.18e-2) −/−	4.0304e-1 (1.63e-2) −/−	4.1805e-1 (1.39e-2) −/−	4.9938e-1 (1.19e-2) +/+	4.5494e-1 (4.63e-3) −/−	4.8463e-1 (3.83e-2) o/−	4.9238e-1 (1.08e-2) +/o
WFG5	3	48	3.8197e-1 (6.49e-2) ≈/−	4.5440e-1 (1.90e-2) +/−	1.3836e-1 (1.25e-2) −/−	4.4672e-1 (1.30e-2) +/−	3.0428e-1 (2.05e-2) −/−	4.5338e-1 (6.87e-3) +/−	4.3058e-1 (1.10e-2) ≈/−	3.8616e-1 (9.72e-2) o/−	4.6557e-1 (1.08e-2) +/o
WFG6	3	48	3.3068e-1 (1.76e-2) −/−	4.8165e-1 (2.97e-2) +/+	2.2141e-1 (1.93e-2) −/−	4.3249e-1 (4.05e-2) +/−	3.1299e-1 (1.65e-2) −/−	4.4779e-1 (1.25e-2) +/≈	4.0970e-1 (8.36e-3) ≈/−	4.2422e-1 (3.87e-2) o/−	4.4844e-1 (1.49e-2) +/o
WFG7	3	48	3.7560e-1 (1.53e-2) −/−	3.3389e-1 (1.36e-2) −/−	2.9271e-1 (1.98e-2) −/−	3.3900e-1 (1.90e-2) −/−	3.6618e-1 (1.20e-2) −/−	3.9431e-1 (5.70e-2) −/−	3.8745e-1 (4.39e-2) −/−	4.2515e-1 (2.21e-2) o/−	4.5470e-1 (1.06e-2) +/o
WFG8	3	48	3.3160e-1 (1.51e-2) −/−	3.0189e-1 (1.44e-2) −/−	2.3908e-1 (2.06e-2) −/−	2.9805e-1 (2.09e-2) −/−	3.2450e-1 (8.76e-3) −/−	4.2726e-1 (1.06e-2) +/+	4.0538e-1 (7.84e-3) +/≈	3.6540e-1 (3.50e-2) o/−	4.0261e-1 (9.49e-3) +/o
WFG9	3	48	4.1548e-1 (1.99e-2) −/−	4.3169e-1 (2.90e-2) −/−	3.8279e-1 (4.16e-2) −/−	4.1649e-1 (2.90e-2) −/−	4.0778e-1 (1.04e-2) −/−	3.9101e-1 (2.58e-2) −/−	3.7793e-1 (2.24e-2) −/−	4.4121e-1 (1.34e-2) o/−	4.5049e-1 (1.06e-2) +/o
+/−/≈ to SRXD			0/14/6	3/15/2	0/16/4	2/14/4	0/16/4	6/7/7	1/10/9		10/2/8
+/−/≈ to NCRXS			1/15/4	2/16/2	0/16/4	0/16/4	0/15/5	3/11/6	2/13/5	2/10/8	

Bibliography

[1] L. Sheng, Z. Ibrahim, S. Buyamin, A. Ahmad, M. Z. Mohd Tumari, M. F. Mat Jusof, and N. Aziz, "Multi-objective particle swarm optimization algorithms - a leader selection overview," *International Journal of Simulation Systems Science & Technology*, vol. 15, pp. 6–19, 08 2014.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[3] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 477–506, Oct. 2006.

[4] Qingfu Zhang, Aimin Zhou, and Yaochu Jin, "RM-MEDA: A Regularity Model-Based Multiobjective Estimation of Distribution Algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, pp. 41–63, Feb. 2008.

[5] K. Deb and H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, pp. 577–601, Aug. 2014.

[6] Qingfu Zhang and Hui Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, pp. 712–731, Dec. 2007.

[7] K. P. Dahal, K. C. Tan, P. I. Cowling, and J. Kacprzyk, eds., *Evolutionary Scheduling*, vol. 49 of *Studies in Computational Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[8] D. H. Wolpert, W. G. Macready, and H. Park, "No Free Lunch Theorems for Search," p. 32.

[9] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, Apr. 1997.

[10] N. Hansen and A. Ostermeier, "Completely Derandomized Self-Adaptation in Evolution Strategies," *Evolutionary Computation*, vol. 9, pp. 159–195, 06 2001.

[11] R. Kruse, C. Borgelt, C. Braune, S. Mostaghim, and M. Steinbrecher, "Fundamental Evolutionary Algorithms," in *Computational Intelligence*, pp. 245–297, London: Springer London, 2016. Series Title: Texts in Computer Science.

[12] R. Storn, "Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces," p. 16.

[13] A. Iorio and X. Li, "Rotationally Invariant Crossover Operators in Evolutionary Multi-objective Optimization," in *Simulated Evolution and Learning* (D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, T.-D. Wang, X. Li, S.-H. Chen, X. Wang, H. Abbass, H. Iba, G.-L. Chen, and X. Yao, eds.), vol. 4247, pp. 310–317, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. Series Title: Lecture Notes in Computer Science.

[14] L. Pan, W. Xu, L. Li, C. He, and R. Cheng, "Adaptive simulated binary crossover for rotated multi-objective optimization," *Swarm and Evolutionary Computation*, vol. 60, p. 100759, 2021.

[15] H.-G. Beyer and K. Deb, "On self-adaptive features in real-parameter evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 3, pp. 250–270, 2001.

[16] K. Deb, D. Joshi, and A. Anand, "Real-coded evolutionary algorithms with parent-centric recombination," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, vol. 1, pp. 61–66 vol.1, 2002.

[17] G. Syswerda, "Uniform crossover in genetic algorithms," 01 1989.

[18] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Syst.*, vol. 9, 1995.

[19] K. Deep and M. Thakur, "A new crossover operator for real coded genetic algorithms," *Applied Mathematics and Computation*, vol. 188, pp. 895–911, May 2007.

[20] S. Tsutsui, M. Yamamura, and T. Higuchi, "Multi-parent Recombination with Simplex Crossover in Real Coded Genetic Algorithms," p. 9.

[21] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," in *Foundations of Genetic Algorithms* (L. D. WHITLEY, ed.), vol. 2 of *Foundations of Genetic Algorithms*, pp. 187–202, Elsevier, 1993.

[22] L. Eshelman, K. Mathias, and J. Schaffer, *Convergence Controlled Variation*, vol. 4, pp. 203–. 01 1997.

[23] L. J. Eshelman, K. E. Mathias, and J. D. Schaffer, "Crossover operator biases: Exploiting the population distribution," in *ICGA*, 1997.

[24] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable Test Problems for Evolutionary Multiobjective Optimization," in *Evolutionary Multiobjective Optimization* (A. Abraham, L. Jain, and R. Goldberg, eds.), pp. 105–145, London: Springer-Verlag, 2005. Series Title: Advanced Information and Knowledge Processing.

[25] S. Huband, L. Barone, L. While, and P. Hingston, "A scalable multi-objective test problem toolkit," in *Evolutionary Multi-Criterion Optimization* (C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, eds.), (Berlin, Heidelberg), pp. 280–295, Springer Berlin Heidelberg, 2005.

[26] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation*, vol. 8, pp. 173–195, June 2000.

[27] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[28] D. Van Veldhuizen and G. Lamont, "On measuring multiobjective evolutionary algorithm performance," in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, vol. 1, pp. 204–211 vol.1, 2000.

[29] P. Czyżżak and A. Jaszkiewicz, "Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization," *Journal of Multi-Criteria Decision Analysis*, vol. 7, pp. 34–47, Jan. 1998.

[30] C. A. Coello Coello and M. Reyes Sierra, "A Study of the Parallelization of a Coevolutionary Multi-objective Evolutionary Algorithm," in *MICAI 2004: Advances in Artificial Intelligence* (G. Goos, J. Hartmanis, J. van Leeuwen, R. Monroy, G. Arroyo-Figueroa, L. E. Sucar, and H. Sossa, eds.), vol. 2972, pp. 688–697, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. Series Title: Lecture Notes in Computer Science.

[31] M. P. Hansen and A. Jaszkiewicz, "Evaluating the quality of approximations to the non-dominated set," p. 31.

[32] D. Brockhoff, T. Wagner, and H. Trautmann, "On the properties of the r2 indicator," in *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, (New York, NY, USA), p. 465–472, Association for Computing Machinery, 2012.

[33] E. Zitzler, J. Knowles, and L. Thiele, *Quality Assessment of Pareto Set Approximations*, pp. 373–404. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[34] P. Ross, "Hyper-Heuristics," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques* (E. K. Burke and G. Kendall, eds.), pp. 529–556, Boston, MA: Springer US, 2005.

[35] G. Fritsche and A. Pozo, "A hyper-heuristic collaborative multi-objective evolutionary algorithm," in *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 354–359, 2018.

[36] L. Hong, J. H. Drake, J. R. Woodward, and E. Özcan, "Automatically Designing More General Mutation Operators of Evolutionary Programming for Groups of Function Classes Using a Hyper-Heuristic," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, (Denver Colorado USA), pp. 725–732, ACM, July 2016.

[37] I. Ono, H. Kita, and S. Kobayashi, "A Robust Real-Coded Genetic Algorithm using Unimodal Normal Distribution Crossover Augmented by Uniform Crossover : Effects of Self-Adaptation of Crossover Probabilities," p. 8.

[38] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: A MATLAB platform for evolutionary multi-objective optimization," *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, 2017.

I herewith assure that I wrote the present thesis independently, that the thesis has not been partially or fully submitted as graded academic work and that I have used no other means than the ones indicated. I have indicated all parts of the work in which sources are used according to their wording or to their meaning.

Magdeburg, 12th February 2022