Analysis of the Distribution of Pareto Optimal Solutions on various Multi-Objective Evolutionary Algorithms

Guido Arnau Antonucci

BSc Computer Science
Submission Date: May 18, 2016
Supervisor: Dr. Peter J. Bentley

Department of Computer Science
University College London

This report is submitted as part requirement for the BSc Degree in Computer Science at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.
Abstract

Multi-Objective Evolutionary Algorithms (MOEA) find multiple equally optimal solutions that minimise two or more objective functions at the same time through the usage of evolutionary strategies. The performance of MOEA is evaluated according to the convergence to the optimal solution set of the problem and the assemblage of a diverse collection of solutions. While the convergence rate of MOEA is often compared, comparisons of the diversity of the found solutions set is often overlooked.

This project compares the quality of the distributions of solutions produced by various popular and novel MOEA. Additionally, biases that MOEA might have are also identified. Before performing the comparisons, a series of tests are conducted in order to verify the capacity of two standard diversity metrics to accurately measure the diversity of a solution set. The Spread metric is found to convey more information about the distribution of solutions than the Spacing metric and thus, produce a better estimation of the quality.

SPEA2, SMPSO, GDE3, AbYSS are found to produce uniform and spread distributions on the tested problems. On the other hand, widely used MOEA such as NSGA-II, $\epsilon$-MOEA, IBEA or CMA-ES are found to produce worse distributions than the previously stated algorithms.
## Contents

4.2 Difficulties Selecting MOEA ........................................ 42
4.3 Results ............................................................... 43
4.4 Analysis ............................................................... 44

5 Conclusions .............................................................. 64
  5.1 Future Work ......................................................... 67

Appendices ............................................................... 69

A Appendices ............................................................. 69
  A.1 User Manual ........................................................ 69
     A.1.1 MOEAFramework/ ............................................. 69
     A.1.2 Comparisons/ .................................................. 70
  A.2 MOEA Default Parametrisation .................................. 73
  A.3 Tuned Parameters .................................................. 73
  A.4 Code Listings ....................................................... 73
     A.4.1 Spread Metric ................................................ 73
     A.4.2 Parameter tuning for NSGA-II ............................. 80
     A.4.3 Comparison of MOEA ....................................... 82
  A.5 Project Plan ........................................................ 89
  A.6 Interim Report ...................................................... 92

Bibliography ............................................................ 95
Chapter 1

Introduction

Single-Objective Optimisation refers to finding an optimal solution to an objective function, while Multi-Objective Optimisation finds solutions to two or more objectives at the same time. The objectives are usually at conflict, meaning that optimising one of them might worsen another one. This results in many optimal solutions which present different trade-offs between the objectives.

Evolutionary Algorithms (EA) are ideal candidates to solve Multi-Objective Problems (MOP) as they work with multiple solutions at the same time throughout their execution (also referred to as population). EA adapted to solve Multi-Objective Problems (MOP) are referred to as Multi-Objective Evolutionary Algorithms (MOEA). How good a MOEA performs when solving a MOP will be judged by the following criteria:

- **Convergence.** In other words, how close the found set of solutions is to the true set of optimal solutions.

- MOP’s optimal solution sets are usually uncountable, but EAs can only maintain a finite sized population. The **Diversity** of the solution set will refer to as how well the solutions are evenly spread out in reference to the true optimal solution set.

Having solutions that are clumped in an area of the true optimal solution set hides other potential solutions and thus, does not give a good solution to the MOP. The capacity of a MOEA to generate evenly spread solutions close to the true optimal
solution set will dictate its performance.

Past research often which compares different MOEA does not focus solely on diversity [1,2] or is now outdated [3,4].

Thus, in this project we are going to focus on comparing the quality of the distribution of solutions produced by various MOEA. Additionally, we will classify the biases or irregularities that each MOEA might have.

In order to accomplish these tasks we need to:

1. Identify both popular and state-of-the-art MOEA. Research the available implementations for the algorithms.

2. Identify a set of MOPs with varied characteristics and difficulty to run the tests on.

3. Research the literature for diversity metrics and implement the most promising ones.

4. Tune the parameters of the selected MOEA against the chosen diversity metrics. Improvements in the spread of the distribution indicate that the metric correctly measures diversity. The most accurate one will be selected.

5. Test the MOEA on the selected MOPs and compare the diversity scores obtained. The process will be automated in order to generate histograms of the solutions found and box plots of the metrics gathered throughout the multiple runs of the MOEA.

Having this information would allow to take informed decisions when choosing a MOEA. One could even use a MOEA in particular in order to generate solutions in a determined range of the solution set. Moreover, the best performing diversity maintaining strategies will be identified.

1.1 Report Structure

Chapter 2 formally introduces the concepts and terminology used throughout the report in detail. A brief description of the tested MOEA is also given. Chapter
3 gives an overview of the process followed to achieve the stated objectives. An introduction and a comparison of the diversity metrics used throughout the literature is also performed. One of these metrics will be selected to use in upcoming tests. Additionally, a brief overview of the software framework used to support the development of the project and existing alternatives is performed. Chapter 4 describes the set-up in which the tests were performed alongside an analysis of the results gathered throughout the comparisons of the diversity of the selected MOEA. Finally, Chapter 5 summarises the most significant results obtained throughout the project and outlines possible future work to be done.
Chapter 2

Background

This chapter introduces the required concepts needed to define the objectives of the project as well as the terminology used throughout the report. The research performed to write it constitutes the first phase of the project in which the MOEA were selected in order to be tested in later stages.

2.1 Multi-Objective Optimisation

Single Objective Optimisation refers to finding one or more solutions that minimise or maximise a given objective function. Multi-Objective Optimisation thus refers to the problem of finding solutions which optimise two or more objective functions. The objective functions are usually in conflict: a solution that minimises one of the objectives might worsen another one. Defining a single solution as the optimal one is thus not possible. Instead, there will be multiple optimal solutions defined by the various trade-offs between the objective functions.

Many day to day problems posses multi-objective nature, where two objectives are at conflict. For example, when a manufacturer wishes to produce a good he would want to minimise its cost while also maximising its quality. A product with low price and low quality would be placed at one extreme of the set of possible solutions, while a product with high price and high quality would be found in the opposite one. Many "middle-ground" solutions can be found between them. Which solution is the most adequate is a subjective choice and will be decided by the
preferences of the manufacturer.

2.1.1 Multi-Objective Problem

A Multi-Objective Problem (MOP) is defined as the minimisation (or maximisation, since it can be converted to a minimisation problem by multiplying each objective function by -1 [5, p. 14]) of:

**Definition 2.1.1 (Multi-Objective problem [5]).**

\[
\begin{align*}
\text{minimise}/\text{maximise} & \quad f_m(x) \quad m = 1, 2, \ldots, M; \\
\text{subject to} & \quad g_j(x) \geq 0 \quad j = 1, 2, \ldots, J; \\
& \quad h_k(x) = 0 \quad k = 1, 2, \ldots, K; \\
& \quad x \in \Omega
\end{align*}
\]

A solution \( x \) is a vector of \( n \) decision variables \( x = (x_1, x_2, \ldots, x_n)^\top \in \Omega \). \( \Omega \) will be referred to as decision space. The vector \( f(x) = (f_1(x), f_2(x), \ldots, f_M(x))^\top \) captures the values of each of the \( M \) objective functions w.r.t the solution \( x \). The space containing this vector will be referred to as objective space.

2.1.1.1 An Example

The following multi-objective problem (MOP) can be found in [6], referenced throughout the report as Schaffer’s first function:

\[
\begin{align*}
\text{minimise} & \quad f_1(x) = x^2 \\
& \quad f_2(x) = (x - 2)^2
\end{align*}
\]

A plot of the two functions can be seen in Figure 2.1. In order to solve the problem a value for \( x \) which minimises both functions must be found. One might assume that \( x = 1 \) is the solution that minimises both objectives. However, \( x = 0 \) and \( x = 2 \), which minimise \( f_1 \) and \( f_2 \) respectively, are solutions which also minimise both functions, albeit not in a balanced manner.

The solution \( x = 3 \) is clearly not optimal since the value of both objectives can be reduced by choosing \( x = 2 \), for example.
As previously mentioned, a multi-objective problem usually has a set of solutions. Thus, \( x \in [0, 2] \) are the range of solutions which minimise the MOP presented and are highlighted in blue in Figure 2.1. Choosing which single solution is the best or more appropriate will be a task left to the Decision Maker (DM), who’s role is to choose the solution which better fits its needs depending on the importance he will give to each objective [7, p. 5].

![Figure 2.1: Plot of the multi-objective problem Schaffer with its respective Pareto Front highlighted in blue.](image)

### 2.2 Pareto Terminology

As shown in the previous example, a multi-objective problem will be solved by a set of solutions rather than by a single one. These solutions are optimal because the value of a component of the objective vector \( f(x) \) cannot be decreased without worsening the other one. These solutions are referred to as Pareto optimal. We should thus introduce the concept of Pareto dominance:

**Definition 2.2.1 (Pareto Dominance [7]).** A vector \( u = (u_1, u_2, ..., u_k) \) is said to dominate another vector \( v = (v_1, v_2, ..., v_k) \) if and only if \( \forall i \in \{1, ..., k\} \), \( u_i \leq v_i \land \exists i \in \{1, ..., k\} : u_i < v_i \) (denoted as \( u \preceq v \), an illustration shown in Figure 2.2).
Figure 2.2: Representation of a set of nondominated solutions (blue) in objective space with respect to two objectives $f_1$ and $f_2$. Source: [7].

The set of solutions which are *nondominated* by any other solution in the decision space of a problem are referred to as the Pareto Optimal Space of the problem:

**Definition 2.2.2 (Pareto Optimal Space [7]).** Given a MOP defined as $f(x) = (f_1(x), f_2(x), \ldots, f_M(x))$, its Pareto Optimal Set is:

$$\mathcal{P}^* := \{ x \in \Omega \mid \nexists x' \in \Omega : f(x') \preceq f(x) \}$$

The corresponding set of vectors of the *Pareto Optimal Set* in objective space will be referred as *Pareto Front*:

**Definition 2.2.3 (Pareto Front [7]).** Given a MOP defined as $f(x) = (f_1(x), \ldots, f_M(x))$, its Pareto Front is:

$$\mathcal{PF}^* := \{ f(x) \mid x \in \mathcal{P}^* \}$$

A MOP’s Pareto Front usually is an uncountable set (as previously seen in Schaffer’s example) and thus cannot be wholly represented by a computer with finite word-lengths. For this reason a multi-objective algorithm will return a discrete
approximation of \( \mathcal{P}^F \) which will be referred as \( PF_{\text{known}} \). Their respective Pareto Optimal Sets will be referred as \( P_{\text{known}} \). Similarly, in the computational domain we cannot represent \( \mathcal{P}^* \) nor \( \mathcal{P}^F \) due to their infinite nature. \( P_{\text{true}} \) and \( PF_{\text{true}} \) will be used respectively to represent computationally tractable approximations of these sets. Usually \( P_{\text{known}} \subset \mathcal{P}^* \) although in some MOPs \( P_{\text{known}} = \mathcal{P}^* \) can be achieved [7, p. 14].

The concept of Pareto Dominance is also restricted by the computational domain: solutions inside \( \mathcal{P}^F \) might never be reached. Obtaining a solution positioned within a "small" distance of the set might be enough. Thus, we introduce the concept of \( \epsilon \)-dominance, seen in Figure 2.3:

**Definition 2.2.4 (\( \epsilon \)-dominance [7]).** A vector \( u = (u_1, u_2, \ldots, u_k) \) is said to \( \epsilon \)-dominate another vector \( v = (v_1, v_2, \ldots, v_k) \) if and only if for some \( \epsilon > 0 \):

\[
\forall i \in \{1, \ldots, k\}, \ u_i \leq (v_i + \epsilon) \land \exists i \in \{1, \ldots, k\} : u_i < (v_i + \epsilon) \text{ (denoted as } u \preceq_{\epsilon+} v).\]

**Figure 2.3:** Solution \( P \) \( \epsilon \)-dominates the entire ABCD region instead of only the PFCE region under the regular domination definition. Source: [8].

### 2.3 Multi-Objective Evolutionary Algorithms

Early methods to solve multi-objective optimisation problems relied on transforming the problem into a single-objective one, causing the following problems [5, p. 75]:

- Each run of the algorithm only generates one solution, thus requiring multiple iterations to generate an approximation of a Pareto Front.
2.3. Multi-Objective Evolutionary Algorithms

- Depending on the method, non-convex Pareto Fronts cannot be found completely.

- All methods require problem specific knowledge about the importance of each objective according to the DM, to be set either before or after the execution of the algorithm [7, p. 31].

Evolutionary Algorithms (EA) use a population of solutions simultaneously on each iteration of the algorithm, rather than working with a single solution, thus allowing the approximation of a whole Pareto Front within a single run of the algorithm. Preserving diversity in the population is a natural feature of EA which makes them a perfect fit to solve MOPs [5, p. 8].

Thanks to this behaviour, Multi-Objective Evolutionary Algorithms (MOEA) do not require preference knowledge about the problem at hand. Moreover, no information about the characteristics of the problem is required as evolutionary algorithms can deal with concave or discontinuous Pareto Fronts.

2.3.1 Diversity in Multi-Objective Optimisation

The main goal in single-objective optimisation is to converge to solutions as close to the global optimum as possible. Multi-objective optimisation has a similar goal, but instead of converging to a single solution, the goal is to find a set of solutions as close to $\mathcal{PF}^*$ as possible. However, Multi-objective optimisation has an additional goal: the set of solutions has to be diverse. In other words, the set of solutions has to be well distributed along the whole optimal Pareto Front in order to portray an accurate estimation.

In a real situation, the DM will only be able to analyse and consider a few of the optimal solutions. Thus, it is important to obtain the maximum possible information about the optimal Pareto Front by distributing evenly the limited number of solutions which a MOEA is able to handle throughout all the Pareto Front, as shown in Figure 2.4. A small but concise selection of optimal solutions will ease the work of the DM.

Similarly to multi-objective solvers, EA have two goals which are at conflict:
exploitation (convergence) and exploration (diversity) [7, p. 24]. If an EA is pressured to focus on the convergence to an optimal solution, the algorithm will explore less of the search space. Analogously, if the EA manages to diversify the explored areas, the algorithm will converge to a mediocre solution.

This project is going to compare the distribution of solutions gathered by each of the algorithms. A good distribution has to extend throughout the whole $PF^*$ and the distance between neighbouring solutions has to be even, as shown in Figure 2.4(a). Additionally, by running the MOEA against MOPs with different characteristics, difficulties and biases that might cause the MOEA to not explore the solution space fully will be identified.

Figure 2.4: These figures represent a set of solutions in objective space where two objectives ($f_1$ and $f_2$) are at conflict. Figures (b) and (c) show bad distributions as the solutions are clumped in determined areas. Instead, Figure (a) shows a good distribution where all the solutions are evenly spread out throughout $P_{true}$. Source: [9]

2.3.2 Structures of the Tested MOEAs

The following sections will describe briefly the Multi-Objective Evolutionary Algorithms tested in this project. Readers not familiar with single-objective evolutionary algorithms are referred to [10, 11].

Comparing how good a solution is when using single-objective evolutionary algorithms is straightforward as each solution has a fitness value assigned. The main challenge faced by MOEA is to compare multiple objectives at once that might not be using the same scale. Additionally, MOEA incorporate different explicit mechanisms to preserve diversity. The descriptions will highlight the differences of each algorithm regarding these aspects as they are the main focus of the project.
2.3.2.1 VEGA

The Vector Evaluated Genetic Algorithm (VEGA) is considered the first multi-objective genetic algorithm. Proposed by [6], VEGA divides its population of size $M$ into $k$ fractions, where $k$ is the number of objectives of the MOP. Each individual in the $j$th ($j \in 1 \ldots k$) sub-population is evaluated using the $j$th objective function. The population is then shuffled and a regular genetic algorithm applies the crossover and selection operators over the new population.

VEGA uses a proportional fitness assignment in order to deal with the differences in scale that each objective might have. Using VEGA is equivalent to transforming the multi-objective problem into a single-objective one by minimising a linear combination of all the objective functions. This approach causes VEGA to usually converge to the local optimums of each of the objective functions [7, p. 72].

2.3.2.2 NSGA-II

The Nondominated Sorting Algorithm II, introduced by [12], is one of the most popular MOEAs and has become the de facto standard to test the performance of a new algorithm. NSGA-II starts by sorting the current population according to the number of solutions which dominate each solution. This results in a series of nondominated fronts with individuals of the same rank. The algorithm then sorts each of these fronts according to the distance between consecutive solutions (or crowding distance, shown in Figure 2.5), thus promoting solutions in low populated areas of the search space, before pushing them into next iteration’s population. Once the new population is filled, the algorithm applies selection, crossover and mutation to generate a child population, which will be merged with the parent population before starting the new iteration.

2.3.2.3 PAES

The Pareto Archived Evolution Strategy was introduced by [13]. PAES follows a (1+1) evolutionary strategy, meaning that a single parent generates a single offspring through mutation (crossover is not used). The offspring will be discarded if it is dominated by the parent or by any solution in the archive. Instead, if the offspring
dominates the parent, the child will replace it in the population and the parent will be inserted into the archive. If the archive is found full, parent and child are compared using an adaptive grid algorithm, which measures the number of solutions which are found inside a grid with as many dimensions as objectives the MOP has. The grid recursively adapts in order to separate both solutions without the need of a niching parameter. The most diverse solution will be kept into the population and pushed into the archive. Other implementations of PAES, such as (1+λ)-ES and (μ + λ)-ES, have been found to perform similarly as the (1+1) version.

2.3.2.4 SPEA2

The *Strength Pareto Evolutionary Algorithm 2*, introduced by [14], improves on the original SPEA algorithm. SPEA2 evaluates the fitness of a set of solutions comprised by the union of a parent and child population generated by selection and crossover operators. The fitness value of solution $i$ is computed as the summ of

$$R(i) = \sum_{j \in Pp \cup Pc, j \prec i} |\{w | w \in Pp \cup Pc \land j \preceq w\}|$$

(where $Pp$ and $Pc$ are the parent and child population, respectively) and the distance to the $k$-th nearest neighbour. The algorithm then selects greedily the nondominated individuals to form a new population of fixed size. If there are too many nondominated individuals, a truncation operator selects individuals according to the distance to the nearest neighbour of a solution. However, if more individuals are needed to fill the population, dominated individuals are added according to their fitness value. The algorithm then generates the appropriate child population and starts a new iteration.
2.3.2.5 PESA-II

The Pareto Envelope-based Selection Algorithm 2 was introduced by [15] and improves on the original PESA. PESA-II maintains an external archive apart from the current population where the nondominated solutions are stored. All the solutions in the archive are nondominated w.r.t each other, otherwise they will be deleted. The algorithm selects two parents from the external archive to generate a child via crossover and mutation. PESA-II divides the objective space into hyperboxes of fixed size in order to select these parents from low-populated hyperboxes. The solutions are added into the current population until it is filled and the next iteration starts.

2.3.2.6 $\epsilon$-NSGAII

When using NSGA-II, the user needs to set a population size, number of iterations, crossover probability and mutation probability. $\epsilon$-NSGAII, introduced by [16], seeks to improve the already mentioned NSGA-II by automating its parametrisation. The algorithm will run multiple iterations of NSGA-II, selecting into a fixed-size archive the nondominated solutions on each iteration according to an $\epsilon$-dominance criteria (See Definition 2.2.4). The updated archive is then injected in the next iteration, and the population size is doubled. The algorithm stops when a user-defined change rate between the previous result and the current one is achieved.

2.3.2.7 $\epsilon$-MOEA

$\epsilon$-MOEA, introduced by [8], uses a method to preserve diversity similar to PAES which consists in the division of the objective space in hyperboxes of size $\epsilon$. $\epsilon$-MOEA maintains an archive of $\epsilon$-nondominated solutions apart from the current population. The algorithm then selects one solution from the current population and one solution from the archive to create a single offspring. The offspring is swapped into the current population if it $\epsilon$-dominates any of its individuals. Similarly, the offspring will substitute an individual in the archive if the first $\epsilon$-dominates the later. In the case that the offspring is $\epsilon$-nondominated w.r.t the archive, the offspring will be accepted only if the hyperbox that contains it has no other solution already in it,
thus preserving a single solution on each hyperbox.

2.3.2.8 IBEA

The \textit{Indicator-Based Evolutionary Algorithm}, introduced by [17], approaches MOEA design differently and avoids the use of a diversity preserving technique by only using a quality metric to guide the search. The suggested metrics are the $I_{\epsilon}$, which quantifies the $\epsilon$ value with which one solution $\epsilon$-dominances another one, and the Hypervolume indicator [1] (introduced in Section 3.3.1.2). The algorithm starts by assigning a fitness value to the current population based on the the comparisons of each solution using the chosen indicator. The solution with the lowest score is deleted from the population and all the fitness scores are recomputed until the population size is lower than a user defined threshold. A mating pool is selected from the resulting population, over which combination and mutation operators are applied to generate a child population, later added to the current population.

2.3.2.9 MOEA/D

The \textit{Multi-Objective Evolutionary Algorithm based on Decomposition}, introduced by [18], optimises each objective independently according to a user defined decomposition approach. In this project, the MOEA/D variant using the Tchebycheff approach and a Differential Evolution operator is used. MOEA/D maintains an array with the minimum solution for each objective found so far. The algorithm then picks randomly three of these solutions in order to generate a single offspring. The offspring is objective-wise compared against randomly selected solutions from the algorithm’s population and substitutes solutions deemed worse by the Tchebycheff criterion. Diversity in MOEA/D is maintained due to the natural behaviour of the algorithm: by handling each objective separately the competing objectives will promote different solutions.

2.3.2.10 NSGA-III

NSGA-III, introduced by [19], seeks to improve on NSGA-II by modifying the selection operators based on the decomposition strategies seen in MOEA/D. NSGA-III thus replaces the crowding distance fitness assignment with an strategy which uses
a series of dynamically computed, well-spread, reference points to aid in the preservation of diversity in the population. The reference points are used similarly as the weight vectors in MOEA/D. The technique is referred to as *multiple targeted search*.

### 2.3.2.11 DBEA

The *Decomposition-Based Evolutionary Algorithm*, introduced by [20], uses a strategy based on decomposition similar to MOEA/D whereby each objective is handled separately. DBEA uses a unique mechanism to preserve diversity whereby two distances are computed for each individual in the population, shown in Figure 2.6: the first one measures the distance between the individual and the origin of the objective space. The second one measures the distance between the previously described line and the solution. DBEA selects two parents randomly from the population and applies a crossover operator to generate an offspring that will compete against the rest of the population. The offspring will replace the previous best solution found for each objective if it has lower values on the previously described distances.

![Figure 2.6: Distance measures used by DBEA (indicated as $d_1$ and $d_2$) to preserve diversity. Source: [20].](image)

### 2.3.2.12 OMOPSO

OMOPSO, introduced by [21], uses the Particle Swarm Optimisation technique inspired by the behaviour of flocks of birds: a population of solutions is "flown" through the search space, guided by a "leader" solution. OMOPSO selects the leaders from the non-dominated solutions of the current population. These solutions are
sent to an external archive, which will be returned as the result. If the number of leaders exceeds a user defined threshold, leaders with the highest crowding distance (used also by NSGA-II, described in Section 2.3.2.2) scores will be deleted. Each individual in the population will be updated using a leader selected using a evolutionary selection operator. A parent will be replaced by its offspring in the current population if the later dominates it.

2.3.2.13 SMPSO

The *Speed-constrained Multi-objective Particle Swarm Optimiser*, introduced by [22], is based on and seeks to improve OMOPSO. The main differences are a new formula to update each individual, which constraints the distance travelled by the solution on each update. Additionally, SMPSO now uses polynomial mutation over the generated offspring.

2.3.2.14 MOCell

MOCell, introduced by [23], is characterised by the usage of a cellular model: individuals are only able to breed with individuals found nearby. By splitting the search space in various overlapped neighbourhoods a diverse exploration of the search space is achieved while convergence takes place inside each neighbourhood independently. MOCell works by mating and mutating individuals in each neighbourhood to generate offspring. The offspring will replace its parents if it dominates them (Definition 2.2.1). Additionally, MOCell maintains an external archive to store nondominated solutions. The offspring will be inserted into the archive if they are nondominated. If the archive is full, the individual with the lowest crowding distance score (the same measure used by NSGA-II described in Section 2.3.2.2) will be replaced. MOCell also injects a fixed number of solutions from the archive into the current population before starting a new iteration.

2.3.2.15 CellDE

CellDE, introduced by [24], improves on the previously mentioned MOCell by using a Differential Evolution operator [25] instead of the regular selection, crossover and mutation operators seen in genetic algorithms. Additionally, CellDE uses the same
diversity criterion introduced by SPEA2 (which measures the distance to the kth nearest neighbour, Section 2.3.2.4) due to the underperformance of the crowding distance operator used in MOCell [26].

2.3.2.16 GDE3

The Generalized Differential Evolution 3 algorithm, introduced by [27], is an algorithm characterised by the usage of Differential Evolution (DE) as the mechanism to explore the search space. GDE3 applies a slightly modified DE operator to generate offspring from the current population. If the offspring dominates the parents, the child will replace them. If the offspring and parents are both nondominated, the offspring is still added to the current population. Thus, the population might exceed the user determined size of the population and will be pruned before starting the next iteration. The pruning criterion will eliminate dominated solutions from the population as well as solutions with a low crowding distance (the same criterion used in NSGA-II, seen in Section 2.3.2.2).

2.3.2.17 CMA-ES

The Covariance Matrix Adaptation Evolutionary Strategy, introduced by [28], is characterised by the usage of a covariance matrix to guide the search. The algorithm generates new offspring from sampling a normal distribution defined by the covariance matrix of each individual. CMA-ES uses a similar technique to GDE3 in order to check if the offspring are better than the parents: the nondominated solution will be chosen always, but in the case where both parent and child are nondominated to each other the ties are broken based on the value of an indicator. The author suggest either to use the crowding distance criterion used by NSGA-II (Section 2.3.2.2) or the Hypervolume indicator [1] (described in Section 3.3.1.2).

2.3.2.18 AbYSS

The Archive-Based hYbrid Scatter Search algorithm, introduced by [29], is characterised by the adaptation of the scatter search framework and the usage of various techniques seen in previous popular MOEAs to build a hybrid evolutionary algorithm. AbYSS maintains a fixed sized external archive of nondominated solutions
which prunes using the NSGA-II crowding distance criterion (see Section 2.3.2.2) in the case the archive becomes full. The current population is split between two sets: one which maintains the best individuals according to the same fitness formula used by SPEA2 (see Section 2.3.2.4), which not only takes into account the number of dominated individuals of each solution but also the distance to the $k$-th nearest neighbour. The second set stores individuals which are far from the individuals in the first set distance-wise to preserve diversity. Pairs of individuals from both sets have a crossover and mutation operator applied. The offspring will replace the parents if the offspring dominates any of them. If both parents and offspring are nondominated, the parents will be sent to the external archive and will be replaced by the offspring.

2.4 Previous Research

Comparisons in the literature usually tend to compare MOEA using either visualisations of the $PF_{known}$ found [30], metrics which independently measure convergence and diversity [2] or metrics that try to measure both convergence and diversity [20].

Many of these comparisons are found only as tests performed in the papers which introduce a new MOEA. These tests are made usually only against a small selection of a few past popular MOEA, proving to not be exhaustive tests. Many of these studies also use a small number of MOPs, which can bias the conclusions of the paper due to the tests not being exhaustive enough and lead to the cherry-picking of MOPs were the MOEA perform well.

Moreover, previous studies focused on the distribution of the solutions [3] use outdated MOEA.

Thus, in this project we are going to fill the gap left in the literature by gathering a comprehensive set of both MOEA and MOPs to test the distribution of solutions generated by these algorithms and bring past researched topics up to date.
Chapter 3

Methodology

This Chapter gives an overview of the whole process followed in order to develop the method used to compare the previously discussed MOEA.

Before starting to research and choose the MOPs and diversity metrics used in the comparisons, implementations of the selected MOEA must be researched and tested. Section 3.1 shows a brief comparison between the available Frameworks which support the software produced in this project. Next, Section 3.2 introduces the MOPs used in the comparisons. Lastly, Section 3.3 gives a brief overview of the available diversity metrics and presents the results obtained from a comparison of the two most promising ones.

3.1 MOEAFramework

Many software packages are available to ease both the development of MOEA and to compare them. In order to aid in the development of the software required to compare the algorithms, multiple frameworks were considered. A comparison between the most promising ones is shown in Table 3.1.

From the main candidates shown, MOEAFramework ¹ was chosen due to its numerous advantages over the more mature and tested candidates. Frameworks not shown in the table (like Opt4J ², Open-BEAGLE ³ and HeuristicLab ⁴) were discarded as these frameworks are focused on the development of Evolutionary

¹http://moeaframework.org
²http://opt4j.sourceforge.net/
³https://chgagne.github.io/beagle/
⁴http://dev.heuristiclab.com/
3.2 MOP Selection

In order to test the performance of the existing diversity metrics (shown in Section 3.3) a comprehensive set of problems with multiple characteristics needs to be compiled.

Over the last few years there have been multiple attempts at creating an exhaustive MOEA test suit. Out of those attempts the ones that have received more attention by the research community are the ZDT functions [4], the DTLZ suite [34] and the WFG Toolkit [35].

An ideal MOEA Test suite will contain MOPs with different characteristics and features in order to potentially expose situations in which the algorithm might not perform well enough. A MOEA performing well on a particular generic test suite only suggests that the algorithm will perform well in the type of problems included in the suite and does not assure good performance in "real-word" problems. In other words, tests performed with test suites will never give definitive answers.

The previously mentioned ZDT, DTLZ and WFG benchmark suites are not exhaustive and thus have shortcomings. The ZDT test suite misses scalable (i.e. a variable number of objectives or decision variables) and varied problems with different characteristics [35], DTLZ falls short on providing deceptive and non-separable problems [35], while the WFG Toolkit only provides hard problems which do not give the necessary information about the diversity of solutions that is needed [7].

The selected MOPs to be used in the comparisons can be seen in Table 3.2 and 3.3. The collection is formed by a varied set of MOPs which try to have easy to solve problems with plottable $P_{true}$ and $P F_{true}$ (Schaffer’s 1 and 2, Fonseca’s 1 and 2, Kursawe, DTLZ1 and ZDT1), hard problems (DTLZ4, DTLZ7 and WFG1-
<table>
<thead>
<tr>
<th>Framework</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
</table>
| MOEAFramework | • Great collection of MOEA and MOP implementations.  
• Multiple metrics implemented.  
• Focused on the comparison of MOEA.  
• Interfaces with JMetal and PISA.  
• Good code quality and well documented.  
• Actively developed. | • Newer software. |
| JMetal v4.5 [31] | • Great collection of MOEA and MOP implementations.  
• Multiple metrics implemented.  
• Easy visualisation of comparison results.  
• Well documented. | • Ongoing rewrite of the core, published as v5.0.  
• Poor code quality. |
| PISA [32] | • First framework published, has a mature userbase.  
• Has many official implementations of MOEA.  
• Widely used in the research community. | • No in-built methods to gather results to compare MOEA.  
• Fragmented codebase in multiple languages.  
• No metric implementations.  
• File poll based communication system between software modules. |
| Paradiseo [33] | • Written in C++.  
• Extensive documentation and tutorials. | • No metrics implemented.  
• Only four old MOEA implementations.  
• No support for comparison of algorithms.  
• Few MOPs implemented. |

Table 3.1: Comparison of Multi-Objective Evolutionary Software frameworks.
9) as well as problems with different features (provided by the DTLZ and WFG benchmarks). Additionally, the WFG and DTLZ functions have a scalable amount of objective functions, thus allowing us to observe the behaviour of MOEA on MOPs with more than 2 objectives. By selecting a varied set of MOPs we expect to gain deeper insight regarding the types of problems each MOEA has difficulty with.

Our research will give special attention to problems with disconnected $PF_{true}$ (Kursawe, ZDT3, DTLZ7), as they show the capacity of a MOEA to maintain a diverse set of solutions along the entirety of the search space. Additional emphasis will be made on MOPs with concave $PF_{true}$ as MOEA have difficulty finding a diverse set of solutions for them [7].

In Table 3.2, the number of objectives marked with an asterisk ("*") indicate that some additional parameters were omitted due to the authors of the problem stating recommended values for them, which were used in the tests.

Additionally, it should be noted that the DTLZ5 and DTLZ6 problems were omitted due to inconsistencies found in their definitions throughout the literature [35].

### 3.3 Metric Selection

Developing a single metric to compare approximation sets ($PF_{known}$) is not straightforward as it is difficult to exactly measure the qualities (mainly closeness to $PF_{true}$ and coverage of a diverse array of solutions) required by a solution set to be deemed "good". For example, in Figure 3.1 two comparisons between approximation sets can be observed: on the left, solution set $A$ is clearly better than $B$ as it dominates all of its solutions completely. However, on the right, judging whether $A$ is better than $B$ is not possible. These difficulties resulted in the usage of visualisations of the solutions along $PF_{true}$ in order to assess both the convergence and diversity due to the lack of metrics [30] during early MOEA development.

Since then, many metrics have been developed, which can be classified into **unary** and **binary**. Unary metrics map a single set of solutions into a real number, while binary metrics compare a solution w.r.t another solution set using also a single
### 3.3. Metric Selection

<table>
<thead>
<tr>
<th>Function</th>
<th># Decision Vars.</th>
<th># Objectives</th>
<th>(P_{\text{true}}) shape</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schaffer 1 [6]</td>
<td>1</td>
<td>2</td>
<td>Connected, convex curve</td>
<td>Plottable in both spaces</td>
</tr>
<tr>
<td>Schaffer 2</td>
<td>1</td>
<td>2</td>
<td>Disconnected, convex curves</td>
<td>Plottable in both spaces</td>
</tr>
<tr>
<td>Fonseca 1 [36]</td>
<td>2</td>
<td>2</td>
<td>Connected, concave curve</td>
<td>Plottable in both spaces</td>
</tr>
<tr>
<td>Fonseca 2</td>
<td>(N)</td>
<td>2</td>
<td>Connected, concave curve</td>
<td>Scaple decision variables</td>
</tr>
<tr>
<td>Kursawe [7]</td>
<td>(N)</td>
<td>2</td>
<td>Disconnected, concave curve</td>
<td>Scaple decision variables</td>
</tr>
<tr>
<td>OKA1 [37]</td>
<td>2</td>
<td>2</td>
<td>Connected, convex curve</td>
<td>Both MOPs use sparse probability distributions, hard problems</td>
</tr>
<tr>
<td>OKA2</td>
<td>3</td>
<td>2</td>
<td>Connected, concave curve</td>
<td></td>
</tr>
<tr>
<td>ZDT1 [4]</td>
<td>30</td>
<td>2</td>
<td>Connected, convex curve</td>
<td></td>
</tr>
<tr>
<td>ZDT2</td>
<td>30</td>
<td>2</td>
<td>Connected, concave curve</td>
<td></td>
</tr>
<tr>
<td>ZDT3</td>
<td>30</td>
<td>2</td>
<td>Disconnected, convex curves</td>
<td></td>
</tr>
<tr>
<td>ZDT4</td>
<td>10</td>
<td>2</td>
<td>Connected, convex curve</td>
<td>Deceptive problem with many local (PF)</td>
</tr>
<tr>
<td>ZDT5</td>
<td>11</td>
<td>2</td>
<td>Connected, convex curve</td>
<td>Uses binary bit string variables</td>
</tr>
<tr>
<td>ZDT6</td>
<td>10</td>
<td>2</td>
<td>Connected, concave curve</td>
<td>Non-uniform distribution of solutions along (PF_{\text{true}})</td>
</tr>
<tr>
<td>DTLZ1 [34]</td>
<td>(M + 4^*)</td>
<td>(M)</td>
<td>Connected and linear</td>
<td>Easy problem</td>
</tr>
<tr>
<td>DTLZ2</td>
<td>(M + 9^*)</td>
<td>(M)</td>
<td>Connected, concave</td>
<td></td>
</tr>
<tr>
<td>DTLZ3</td>
<td>(M + 9^*)</td>
<td>(M)</td>
<td>Connected, concave</td>
<td>Deceptive problem with many local (PF)</td>
</tr>
<tr>
<td>DTLZ4</td>
<td>(M + 9^*)</td>
<td>(M)</td>
<td>Connected, concave</td>
<td>Specially used to test MOEA distributions</td>
</tr>
<tr>
<td>DTLZ7</td>
<td>(M + 19^*)</td>
<td>(M)</td>
<td>Disconnected, convex surfaces</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2: Comparison of selected Multi-Objective Problems.
### Table 3.3: Comparison of selected Multi-Objective Problems. Continued from Table 3.2.

<table>
<thead>
<tr>
<th>Function</th>
<th># Decision Vars.</th>
<th># Objectives</th>
<th>$P_{true}$ shape</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFG1 [35]</td>
<td>$M + 9^*$</td>
<td>$M$</td>
<td>Connected, mixed</td>
<td>Flat areas of $PF$ bias the search</td>
</tr>
<tr>
<td>WFG2</td>
<td>$M + 9^*$</td>
<td>$M$</td>
<td>Disconnected, convex</td>
<td></td>
</tr>
<tr>
<td>WFG3</td>
<td>$M + 9^*$</td>
<td>$M$</td>
<td>Connected, linear</td>
<td>Uni-dimensional, even with three objectives</td>
</tr>
<tr>
<td>WFG4</td>
<td>$M + 9^*$</td>
<td>$M$</td>
<td>Connected, concave</td>
<td>Highly multimodal</td>
</tr>
<tr>
<td>WFG5</td>
<td>$M + 9^*$</td>
<td>$M$</td>
<td>Connected, concave</td>
<td>Deceptive problem</td>
</tr>
<tr>
<td>WFG6</td>
<td>$M + 9^*$</td>
<td>$M$</td>
<td>Connected, concave</td>
<td></td>
</tr>
<tr>
<td>WFG7</td>
<td>$M + 9^*$</td>
<td>$M$</td>
<td>Connected, concave</td>
<td></td>
</tr>
<tr>
<td>WFG8</td>
<td>$M + 9^*$</td>
<td>$M$</td>
<td>Connected, concave</td>
<td></td>
</tr>
<tr>
<td>WFG9</td>
<td>$M + 9^*$</td>
<td>$M$</td>
<td>Connected, concave</td>
<td>Combines the characteristics from 6 previous problems, highly difficult</td>
</tr>
</tbody>
</table>

**Figure 3.1:** Left: solution set A is better than B. Right: A is not comparable to B as both sets have mutually dominated solutions. Source: [38]
real number.

Binary metrics have been shown to provide more robust results than unary metrics [39]. However, due to the large amount of MOEA and MOPs selected in this study, using binary metrics would result in a quadratic number of comparisons. Unary metrics are going to be used instead since they have been used widely in the literature and have been shown to provide sufficient results.

3.3.1 Available Diversity Metrics

This Section introduces the most popular diversity metrics used throughout the literature. When measuring the diversity achieved by a MOEA it is important to observe the different trade-offs between the conflicting objectives. Thus, all unary metrics introduced in this Section measure the diversity of the solutions in *objective space* rather than in *decision space*. It should be noted that some problems have linear mappings between both spaces and thus, measuring diversities in either of them will produce the same results [5, p. 24].

Multiple diversity metrics are available. In Section 3.3.1.1 metrics which estimate evenness and spread of a diversity distribution exclusively are presented, while in Section 3.3.1.2 metrics which also estimate at the same time the convergence to PF_{true} are shown (referred to as mixed metrics).

3.3.1.1 Diversity Metrics

**Spacing (S):** Developed by [40], the metric simply measures the standard deviation of the Manhattan distances between neighbouring solutions. Equation 3.1 shows the computation of the Spacing metric for PF_{known}, where \( m \) is the number of objectives of the MOP. Note that a zero value indicates a perfectly uniform distribution.

\[
S = \sqrt{\frac{1}{|PF_{known}|} \sum_{i=1}^{|PF_{known}|} (\bar{d} - d_i)^2} \tag{3.1}
\]

where

\[
d_i = \min_{j \in PF_{known}} \left( \sum_{k=1}^{m} \left| f^i_k(x) - f^j_k(x) \right| \right) \tag{3.2}
\]

The metric was designed to be used in conjunction with other metrics since it
only conveys how well did the MOEA spread its solutions. Additionally, it is fast to compute and can be used in "real-world" problems where \( PF_{true} \) is not known beforehand.

**Spread** (\( \Delta \)): Developed by [12] for two objective problems and generalised to multiple objectives by [41], this metric expands on the previously seen Spacing metric in order to incorporate information about \( PF_{true} \): aside from computing the distance deviation between neighbouring solutions, the metric also measures how far are the most extreme solutions of \( PF_{known} \) to the closest solutions of \( PF_{true} \), thus measuring also if the solution set is widely spread throughout all \( PF_{true} \) (see Equation 3.3).

\[
\Delta = \frac{\sum_{k=1}^{m} d_{ek} + \sum_{i=1}^{\mid PF_{known} \mid} |d_i - \bar{d}|}{\sum_{k=1}^{m} d_{ek} + \mid PF_{known} \mid \cdot \bar{d}}
\]  
(3.3)

where \( d_i = \min_{f^i \in PF_{known}} (\|f^i(x) - f^j(x)\|_2) \)  
(3.4)

and \( d_{ek} = \min_{f^i \in PF_{known}} (\|e^k - f^j(x)\|_2) \)  
(3.5)

\( e_k = \max_{f^k \in PF_{true}} (f^k(x)) \)  
(3.6)

Note that the previous two metrics do not correctly identify if a solution set \( A \preceq B \); the metrics only give an idea about the distribution of the solutions and should be interpreted as such.

It should be noted that the objectives need to be normalised before computing these metrics as they are susceptible to changes in the scaling.

More recently other diversity metrics have been suggested such as the *Sigma Diversity Metric* [9], the *Entropy Measure* [42] and the *DCI* [43]. References have not been found outside the paper that introduced them and have been omitted due to not being researched enough. Other metrics such as the *Chi-Square-Like Deviation Measure* [5, p. 331] and the *Maximum Spread* [5, p. 330] have been also discarded since they have been shown to not perform as well as the introduced ones.
3.3. Metric Selection

3.3.1.2 Mixed Metrics

Due to the small number of metrics which solely measure diversity, mixed metrics which also measure convergence at the same time are also considered.

**Hypervolume (HV):** Introduced by [1], this metric adds up the volumes formed by the hyperboxes defined between a given reference point in objective space and each of the solutions in $PF_{\text{known}}$ before normalising the resulting value with the hypervolume formed by the solutions in $PF_{\text{true}}$. An HV value of 1.0 would thus indicate that $PF_{\text{known}} \approx PF_{\text{true}}$.

![Figure 3.2](image)

**Figure 3.2:** In the upper plots two solutions sets such that HV(A) $>$ HV(B) are shown. Below, only due to a change of the reference point, HV(A) $<$ HV(B). Source: [44]

Thus the HV metric can capture both the distance between $PF_{\text{true}}$ and $PF_{\text{known}}$ (convergence) as well as the difference in spread. In spite of its exponential computational complexity, the metric correctly identifies if a solution set $A \preccurlyeq B$. However, a badly chosen reference point might produce erroneous results, as seen in Figure 3.2. Additionally, the metric might yield misleading results when $PF_{\text{known}}$ in non-convex [1].
Inverted Generational Distance (IGD): Introduced by [45], the IGD metric measures the average Euclidean distance between each solution of \( PF_{true} \) and their corresponding closest solution in \( PF_{known} \). A poor value would be obtained, for example, in the case that the extreme solutions of \( PF_{true} \) are far apart from \( PF_{known} \).

Other more recent mixed metrics have been developed, such as the \( G\)-Metric [46] and the \( Averaged Hausdorff Distance \) [47] but, similarly, these metrics have not been adopted by the research literature and are thus not used.

### 3.3.2 Diversity Metric Evaluation

The comparison between the MOEA will be supported mainly by a single metric. Thus, either the Spacing or Spread metric, presented in Section 3.3.1.1, need to be selected.

As explained in Section 2.3.1, convergence and diversity are at conflict when solving a MOP. A mixed metric might improve due to an increase of the convergence and not due to an improvement in diversity, providing thus misleading results about the quality of the distribution of solutions. For example, in Figure 4.3(a), the HV metric indicates that CMA-ES and AbYSS generate \( PF_{known} \) with similar qualities. However, as seen in Figure 4.2, CMA-ES’s distribution is uneven and clumped in the middle of the set, while AbYSS’s distribution is evenly spread. The Mixed metrics presented in 3.3.1.2 are thus not considered for the tests as the inclusion of the evaluation of the convergence interferes with our objective of comparing the diversity. Instead, they will be used to support the results produced by our main metric during the MOEA comparisons.

In order to select a metric, a comparison will be performed by tuning the parameters of each MOEA against the Spacing and Spread metrics. Through the visualisation of the resulting distributions conclusions will be made regarding the accuracy of each of them. If none of the tested metrics prove to be good enough to assess the diversity of a solution set, a new metric will have to be designed.

Section 3.3.2.1 gives an overview of the process followed to implement the tests in MOEAFramework. Next, Section 3.3.2.2 explains the process followed and the set-up where the experiments were run on. Lastly, in Section 3.3.2.3, the
3.3. Metric Selection

effectiveness of both metrics is reviewed.

3.3.2.1 Spread Metric Implementation

MOEAFramework will be used to perform the comparison between the Spacing and Spread metrics. While the Spacing metric is already built into the software, the Spread metric needs to be implemented. Algorithm 1 shows the pseudocode used to implement it. The resulting Java code can be seen in Appendix A.4.1. For further details into the classes needed to introduce a new metric into MOEAFramework, see commit d9ef5 on the GitHub repository of the project.

Algorithm 1 Computation of the Spread metric.

```
1: procedure Spread
2:    sort PFknown lexicographically
3:    if |PFknown| = 0 or distance(PFknown[0], PFknown[end]) = 0 then
4:       return 2.0
5:    else
6:       m ← number of objectives
7:       extremeValues ← ∅
8:       for k ∈ 1...m do
9:          sort PFtrue according to the k-th objective
10:         extremeValues ← extremeValues ∪ PFtrue[end]
11:       end for
12:       dists ← ∅
13:       for x ∈ PFknown do
14:          dists ← dists ∪ minj∈PFknown {distance(x, j)}
15:       end for
16:       distExtremes ← 0
17:       for x ∈ extremeValues do
18:          distExtremes ← distExtremes + minj∈PFknown {distance(x, j)}
19:       end for
20:       sum ← 0
21:       for d ∈ dists do
22:          sum ← sum + |d − mean(dists)|
23:       end for
24:       return (distExtremes+sum)/(distExtremes+|PFknown|·mean(dist))
25:    end if
26: end procedure
```

To verify the implementation Unit Tests have been incorporated into

---

3 https://github.com/Gan0k/MOEAFramework/commit/d9ef5

6 Unit Test’s source code at GitHub
MOEAFramework using JUnit\(^7\). The tests run over some edge cases such as an empty Pareto Front, which should return 1.0, and simple cases such as approximation sets with solutions \( PF_{\text{known}} = \{(0, 1), (0.5, 0.5), (1, 0)\} \), which should return 0.0.

### 3.3.2.2 Tuning Process

For each MOEA, a program was developed to easily interact with the algorithm’s parameters from the command line. A snippet of the code can be seen in Appendix A.4.2 and indications on how to run it can be found in Appendix A.1. These programs output, for each indicated metric, the minimum, median and maximum values for achieved by the MOEA with default and tuned parameters, respectively. This setup is designed to exploit the fact that MOEAFramework uses the Kruskal-Wallis and Mann-Whitney methods to test for statistical significance between the two different set-ups in order to compare solutions gathered from the an stochastic process such as an EA. Inequality hypothesis are rejected if \( p \)-value < 0.05. Thus, MOEAFramework will indicate us when the values produced with the modified parameters are statistically indifferent from those gathered with the default parameters for a particular metric.

The MOEA will be tuned by first modifying a single parameter, fixing it to a value which produces the highest increase of the tested metric, and proceeding with the same process for the rest of the parameters.

A subset of MOEA and MOPs was selected from the previously introduced collections due to the time consuming nature of the described procedure. The tested MOEA are NSGA-II, MOEA/D, GDE3 and IBEA with Schaffer’s 1 and 2 functions, Fonseca’s first, DTLZ benchmark’s functions and the WFG Toolkit.

The visualisation of the found Pareto Fronts will be done through histograms of the solutions. Through these plots we will be able to observe both the distribution of solutions as well as the areas where solutions are clumped. Only two-dimensional spaces can be plotted using this method, thus the DTLZ and WFG Toolkit functions will be used only with two objectives.

\(^7\)http://junit.org/
In order to assure that each MOEA does the same amount of work, the number of function evaluations (that is, the number of times the algorithm evaluates any of the objective functions) was set to 10,000. Moreover, the population of each MOEA was set to 100 individuals. To obtain robust results we need to execute each MOEA with different random seeds in order to discard possible "lucky" results. Thus, each MOEA was executed 100 times with different random seeds.

The tuned MOEA parameters can be found in Appendix A.3.

3.3.2.3 Analysis and Conclusions

After tuning the MOEA against the Spread and Spacing metrics the following was concluded:

1. Unexpectedly, in few situations the Spread metric gave different assessments than the Spacing metric. One of these cases is shown in Figure 3.3, where a disagreement between the two metrics can be observed. The Spacing metric shows that the distribution of solutions might not be uniform on each run, yet the Spread metric manages to discern a better distribution. However, most situations show that both metrics agree on the improvement or decline in quality of the Front (see Figure 3.5).

2. It was expected that when tuning a MOEA, a metric would increase or decrease based on parameter changes. However, when changing them, an increase in the variance of the metrics throughout the multiple runs was observed instead, complicating the process.

3. The increase in variance was most notably observed in the Spacing metric, as the locations of solutions can vary greatly between runs and skew easily the metric.

4. The metrics have a hard time assessing non-convex Pareto Fronts (see Figure 3.4). Convex Fronts are assessed well (see Figure 3.5).

5. It is easier to compare MOEA using the Spread metric as it is bounded between zero and two. The Spacing metric can be easily mislead when the MOEA
does not fully converge to $P_{F_{true}}$.

6. In some cases, tuning against diversity metrics also improved convergence (see Figure 3.6).

Both metrics gave thus accurate assessments about the quality of the distribution of solutions. However, as stated, the Spread metric gave more consistent results and eases the comparison of results due to being bounded.

In the next chapter the Spread metric is going to be used to compare the MOEA. Nonetheless, the Spacing metric is still going to be used to gain insight on the uniformity of the distribution.

**Figure 3.3:** An improvement of the distribution of solutions is not detected by Spacing when IBEA is run against Fonseca, yet the Spread metric manages to identify it. Table (b) contains the median of each metric gathered throughout 50 runs. Note that Figure (b) contains plots in decision space.
3.3. Metric Selection

(a)

<table>
<thead>
<tr>
<th></th>
<th>HV</th>
<th>IGD</th>
<th>$I_{e_s}$</th>
<th>Spacing</th>
<th>Spread</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>0.153</td>
<td>0.0434</td>
<td>0.0605</td>
<td>0.0314</td>
<td>0.5922</td>
<td>0.405</td>
</tr>
<tr>
<td>Tuned</td>
<td>0.1545</td>
<td>0.0422</td>
<td>0.0563</td>
<td>0.027</td>
<td>0.555</td>
<td>0.4037</td>
</tr>
</tbody>
</table>

Figure 3.4: The WFG8 problem contains a concave Pareto Front. The Spread and Spacing metrics detect a better distribution although the histogram in Figure (a) shows that the solutions got clumped towards one extreme. Table (b) contains the median of the metrics over 50 runs.

(b)

Figure 3.5: The Spacing and Spread metrics agree that the distribution of solutions is better when running MOEA/D against Fonseca’s first function. Note that Figure (a) is plotted in objective space. Table (b) contains the median of the metrics over 50 runs.
3.3. Metric Selection

Figure 3.6: An improvement of both convergence and diversity metrics when running NSGA-II against the DTLZ3 problem with two objectives is shown. Note that $PF_{true} = \{(0,0,0)\}$. Once tuned, NSGA-II produced solutions much closer to $PF_{true}$. Plot (a) contains the histogram of solutions over 50 runs. Table (b) contains the medians of the metrics.

<table>
<thead>
<tr>
<th></th>
<th>IGD</th>
<th>$l_e$</th>
<th>Spacing</th>
<th>Spread</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>13.13</td>
<td>12.84</td>
<td>3.01</td>
<td>0.96</td>
<td>3.37</td>
</tr>
<tr>
<td>Tuned</td>
<td>8.164</td>
<td>7.32</td>
<td>0.87</td>
<td>0.87</td>
<td>1.96</td>
</tr>
</tbody>
</table>

Table (b)
Chapter 4

Experiments

To compare the quality of the distribution of solutions generated by the selected MOEA presented in Section 2.3.2, they will be executed against the MOPs described in Section 3.2 and the resulting $PF_{known}$ will be evaluated using the metrics presented in Section 3.3.

This Chapter introduces the experimental set-up followed to perform these tests. In Section 4.1, a series of convergence metrics are presented to further support our analysis. An overview of the data we expect to collect is also given. Additionally, the parametrisation used in each MOEA is stated along with the environment set-up used to run the tests. Section 4.2 contains an overview of the challenges faced when performing the tests. Lastly, in Section 4.3 the results of the tests are presented and in Section 4.4 the performance of each MOEA is analysed alongside the biases and difficulties that each MOEA showed.

4.1 Collected Metrics and Data

When mapping the characteristics of $PF_{known}$ into a single scalar some information will be inevitably lost. Thus researchers usually use multiple metrics which capture different characteristics of $PF_{known}$ in order to get more robust results when comparing MOEA. Aside from the metrics described in 3.3, the following convergence metrics will also be used.

It should be noted that a set of unary metrics which accurately describes the quality of $PF_{known}$ has been proven to be impossible by [39].
• Generational Distance (GD) [48], which measures the distance of each solution of $PF_{known}$ to the closest solution of $PF_{true}$, contrary to the IGD metric.

• Additive $\epsilon$-indicator ($I_{\epsilon^+}$) [39], which generalises the concept of $\epsilon$-dominance introduced in Definition 2.2.4 between $PF_{true}$ and $PF_{known}$ to a metric containing the value of $\epsilon$.

• $R_2$ Indicator [49], which similarly measures the distance of $PF_{known}$ to $PF_{true}$.

As stated in Section 2.3.1, convergence and diversity are at conflict in the execution of an EA. These metrics will allow us to assess if a good distribution of solutions is caused by a poor convergence to $PF_{true}$. They have been chosen due to their wide usage in the literature and availability inside MOEAFramework. The usage of the Hypervolume, $I_{\epsilon^+}$ and R2 indicator are specifically recommended by [38].

A program (see Appendix A.4.3 for the code and Appendix A.1 to find how to use it) was developed in order to automate the process of comparing the selected MOEA, producing the following data when given a series of MOPs on which to test the MOEA on:

• The found $PF_{known}$ by each MOEA through multiple runs.

• For each MOP, the minimum, median and maximum value obtained by each MOEA for the stated metrics. Additionally, as stated in Section 3.3.2.2, MOEA statistically indifferent from one another will be indicated by using the Kruskal-Wallis and Mann-Whitney methods. Inequality hypothesis are rejected when $p$-value $< 0.05$.

• Due to the stochastic nature of EA, each MOEA has to be executed multiple times for each MOP in order to gather consistent results. Box plots will be produce in order to observe side-by-side the different results of each MOEA. JFreeChart\(^1\) was used to produce them.

\(^1\)http://www.jfree.org/jfreechart/
• The average execution time of each MOEA for each MOP will be gathered in order to further support our comparison. MOEA which have significantly higher execution times will be held to higher standards.

Analysing the Pareto Fronts by visualising them directly might be inaccurate. However, as stated in Section 3.3, MOEA metrics do not fully convey all the useful information about the solution sets. Thus, aside from the already mentioned box plots, histograms of the union of the Pareto Fronts found by the MOEA through the multiple runs will be produced. A simple MATLAB script has been produced to plot them from the printed $PF_{known}$, additionally adding $PF_{true}$ into the plot for reference (information about it can be found in Appendix A.1).

4.1.1 MOEA Parametrisation

The default parameters of the MOEA shown in Table A.1 were used in all the tests. The values were gathered from the paper which introduced the algorithm when possible, as only some authors do give recommended values for a MOEA’s settings. When not available, the default parameters set by MOEAFramework were used.

The previously mentioned program to automatically run the tests also has the ability to read configuration files automatically in order to be able to modify the parametrisation of a MOEA with ease (see Appendix A.4.3).

4.1.2 MOP Parametrisation

A MOEA will have more difficulty solving a MOP as the number of objective function increases. Thus, the DTLZ and WFG Toolkit’s functions were run with both 2 and 3 objectives, allowing to observe the performance of the MOEA in situations where the number of objectives was increased.

Fonseca’s second function and Kursawe’s function were run with 3 decision variables each.

All the MOPs presented in Section 3.2 use Real Numbered-Variables in their problems. Therefore, the MOEA which require them, use the Simulated Binary Crossover Operator [50] and the Polynomial Mutation Operator to be able to emulate the crossover and mutation of bit-string variables seen in regular EA. The
configuration of each of these operators can be seen in Appendix A.2.

However, the ZDT5 problem required the usage of bit-string variables. As a result, the MOP could only be executed with MOEA which do support them: NSGA-III, SPEA2, MOCell, DBEA, PESA-II, PAES, $\epsilon$-NSGA-II, $\epsilon$-MOEA, NSGA-II and IBEA. When running on ZDT5, the MOEA used the Half-Uniform Crossover Operator with $p_c = 1.0$ and regular bit-flip mutation with $p_m = 0.01$.

### 4.1.3 Experimental Set-up

All MOEA were run 50 times with different random seeds for 10,000 objective function evaluations, assuring us that each MOEA performs the same amount of work. Using different seeds for the internal Java random number generator assures us that the results obtained for a particular MOEA are consistent and not a result of choosing by chance a seed that favours the algorithm.

The Amazon Web Services Compute Cloud was used to run the tests since it takes approximately 72 hours to generate all the results. The m3.medium 2 instance type, with one Intel Xeon E5-2670 v2 CPU core and 3.75GB of RAM, was used.

### 4.2 Difficulties Selecting MOEA

The MOEA presented in Section 2.3.2 were selected mainly based on the uniqueness of their diversity preservation mechanism. However, in principle, as many MOEA as possible should be tested. The implementation availability and stability ended up heavily influencing the MOEA which ended-up being selected.

Multiple times a particular MOEA would either raise an error or get stuck in a particular MOP. As a result, some MOEA were not run against them. In particular, CMA-ES crashed when running on Fonseca’s first function due to a negative eigenvalue calculation on the covariance matrix, while $\epsilon$-MOEA, $\epsilon$-NSGAI2 and CMA-ES hanged when solving the three-dimensional WFG problems and their executions were skipped. Moreover, JMetal’s (introduced in Section 3.1) implementation of IBEA had to be used instead of the one proportioned in MOEAFramework due to crashes in the ZDT2-7 problems.

[^2]: https://aws.amazon.com/ec2/instance-types/
4.3 Results

The selection of MOEA has also been considerably affected by faulty implementations. Originally the FastPGA algorithm was going to be included, but after several crashes the MOEA was omitted. The error causing the problem was debugged and reported on MOEAFramework’s issue tracker 3. The SMS-EMOA algorithm should also have been included, but after multiple tests the program hanged when running even the simplest functions, like Schaffer’s first MOP, and was thus discarded.

The HypeE algorithm was also to be included. An implementation of the algorithm is available in the PISA framework, introduced in Section 3.1, and MOEAFramework has the capacity to interface with it. The interface works using a file polling mechanism to check when PISA’s output files have been modified. After multiple attempts, its inclusion was abandoned as this mechanism seemed to induce multiple errors due to race conditions when accessing the files.

NSGA-III’s implementation has also been reported to be incomplete4 and, at the time of writing this report, a fully functioning implementation has yet to be released. NSGA-III was at first excluded from the comparisons. However, testing it might still bring further insight in the identification of high performing diversity-preserving mechanisms, as NSGA-III uses a similar strategy to MOEA/D (see Section 2.3.2.10). Nonetheless, NSGA-III results should not be taken as definitive.

It should be noted that none of the implementations used have been developed by the original authors of the algorithms. We trust nonetheless that the implementations of the MOEA given by MOEAFramework are correct.

Raw data is not included in the Appendix due to brevity, but is available and can be visualised through a web interface in the git repository of the project5. The data used to generate the box plots of the metrics is available in

---

3https://github.com/MOEAFramework/MOEAFramework/issues/40
4https://www.researchgate.net/post/Is_there_a_fully_functional_NSGA-III_implementation
5https://bitbucket.org/Ganok/moea-analysis/
4.4. Analysis

This Section outlines the quality of the distribution of solutions obtained by each of the tested MOEA. Examples are given in order to illustrate the given statements. Convergence scores are also included in order to verify if the underperformance or overperformance of a MOEA also translates into an increase or decrease in convergence.

CMA-ES produced really poor distributions throughout all the tested MOPs. Moreover, the evenness of the distribution, measured by the Spacing metric, always ranked among the worst, many times being magnitudes higher than other MOEA. These results can be partially attributed to the awful convergence that the algorithm showed on difficult problems (such as seen in Figure 4.1). By not converging to $PF_{true}$, CMA-ES produced solutions usually clumped in easy to reach areas of the search

\footnotesize{tests/compare/analysis\textsuperscript{6} folder at BitBucket
tests/compare/sets\textsuperscript{7} folder at BitBucket
tests/compare/boxplots\textsuperscript{8} folder at BitBucket
tests/compare/plot_sets\textsuperscript{9} folder at BitBucket
tests/compare/plot_times\textsuperscript{10} folder at BitBucket
tests/compare/time\textsuperscript{11} folder at BitBucket}
space. Additionally, CMA-ES showed convergence and distribution problems on MOPs with disconnected \( PF_{true} \) like Kursawe or ZDT3, as seen in Figure 4.4.

**Figure 4.1:** CMA-ES shows poor convergence on DTLZ1 with 2 objectives. Nonetheless, the Spread metric indicates a mediocre score due to the \( PF_{true} \) being a single point (shown in orange in (d)).

However, CMA-ES managed to achieve the best convergence on simple functions such as Schaffer’s first and second function and Fonseca’s second function. In Schaffer’s functions, these results translated also into a good distribution of solutions, although still surpassed by other MOEA (see Figure 4.2). Although it shows promising results in terms of convergence on simple problems, CMA-ES still needs far more function evaluations to converge than other MOEA and still, when it does,
produces uneven distributions. Moreover, the MOEA found difficulties converging in the ZDT1, ZDT2 and ZDT3 problems, showing extremely high execution times.

Figure 4.2: CMA-ES shows really good convergence on Schaffer 1, but the generated distribution of solutions is uneven. The Hypervolume scores for this MOP can be see in Figure 4.3. The execution times of each MOEA for this MOP can be seen in Figure 4.5(b).

AbYSS showed really promising results in regards of the distribution of solutions while also obtaining average results on the convergence metrics. AbYSS scored highly on the Spacing metric, showing even distribution of solutions across all
MOPs except the WFG functions with three objectives, as seen in Figure 4.13(b). While some MOEA fluctuate heavily regarding the quality of the distribution depending on the MOP they are run against, AbYSS produced good results on the Spread metric across all MOPs, as seen for example in Figure 4.14(a) or 4.10(a). Moreover, it does not seem that AbYSS has problems dealing with disconnected fronts. Albeit obtaining mediocre results regarding the convergence metrics (see Figure 4.7(c) or 4.10(c)), AbYSS proved to provide a consistent balance between high quality distributions and converging to the optimal solutions.

(a) Hypervolume values obtained during 50 runs (greater is better).
(b) Spacing values obtained during 50 runs (lower is better).

Figure 4.3: Results of AbYSS run against Schaffer 1. The distribution of solutions is much better than the one provided by CMA-ES in Figure 4.2, however the convergence is worse. The HV metric, however, indicates an average result for both MOEA when clearly AbYSS distribution is much better.

MOCell and CellDE showed really inconsistent results on all metrics, sometimes receiving top scores on some MOPs while scoring poorly on others. Both MOEA obtained average Spread scores, although in multiple MOPs MOCell obtained the lowest (i.e. best) score (see for example Figure 4.14(a)). It was observed that if MOCell performed well on the Spread metric, then CellDE would then not perform as good, while the contrary was also observed. This behaviour is completely MOP
dependant, suggesting that rather than CellDE outperforming MOCell in terms of diversity as the authors would have hoped (see Section 2.3.2.15), the algorithms rather complement one another. CellDE scored multiple times really poorly on the Spacing metric, while MOCell tended to average good scores. CellDE also struggled to converge to optimal solutions and was often surpassed by MOCell on the GD metric. Examples of these behaviours can be seen in Figure 4.4, where MOCell shows better a convergence and spread and spacing than CellDE.

Both MOEA seem to provide a good balance between diversity and convergence, although they lack the robustness seen in AbYSS: sometimes the MOEA produce the best solutions approximations across all algorithms, while sometimes score really poorly. Although both MOEA use the same underlying cellular strategy, they also showed significantly different results. Most of the time, MOCell performed better than CellDE regarding both diversity and convergence in spite of the fact that CellDE is supposed to be an improved version of the first one (see Figure 4.7). MOCell shows better distributions since CellDE tends to favour individual minimums of the objective functions. Only in WFG’s three-objective problems CellDE saw better results (see Figure 4.13). Moreover, the execution times of CellDE were magnitudes higher than any other MOEA in some particular MOPs such as Fonseca’s and Schaffer’s functions (see Figure 4.5(b) and 4.10(e)) and the DTLZ two-objective problems (see Figure 4.14(e)).

GDE3, on the other hand, performed quite similarly to CellDE, albeit obtaining considerably better scores in some problems. Both MOEA are related in their usage of Differential Evolution as the evolutionary operator. GDE3 showed really promising results regarding both the Spread and Spacing metrics, ranking multiple times among the best MOEA. For example, GDE3 scored the best results on ZDT3 across all metrics (see Figure 4.6). However, as it is expected from such algorithm, convergence scores were really average on the majority of the tests (see Figure 4.7). It is remarkable the low variance between the scores achieved by GDE3, suggesting that the results obtained with the algorithm are really consistent. GDE3
4.4. Analysis

(a) Spread values obtained during 50 runs (lower is better).

(b) Spacing values obtained during 50 runs (lower is better).

(c) GD values obtained during 50 runs (lower is better).

(d) Hypervolume values obtained during 50 runs (higher is better).

(e) Histogram of the solutions found by MOCell over 50 runs.

(f) Histogram of the solutions found by CellDE over 50 runs.

Figure 4.4: CellDE shows poor convergence on ZDT3 while MOCell obtains the best scores across all metrics. $P_{F_{true}}$ is marked on Figures (e) and (f) with orange.
4.4. Analysis

Figure 4.5: (a): $\epsilon$-NSGA-II produces an unbalanced distribution for Schaffer’s first function in Figure (a). CellDE and SPEA2 show extremely high execution times in Figure (b).

Also performed poorly regarding the diversity of solutions on all WFG problems. Nonetheless, the convergence of GDE3 on these MOPs were amongst the best. GDE3, together with AbYSS, seem to produce the most well spread and even distributions of all the MOEA.

$\epsilon$-NSGA-II did not perform remarkably well in terms of the quality of the distribution in any of the MOPs. $\epsilon$-NSGA-II additionally showed multiple times really inconsistent spacing results with high variances, similar to the ones seen in Figure 4.4(b). However, $\epsilon$-NSGA-II showed really promising results on the convergence metrics, as seen on Figure 4.2(c). The algorithm also tended to generate solutions away from the individual minimums of the objectives functions, as seen for example in Figure 4.5(a). The MOEA also showed problems solving the ZDT problems, converging to badly distributed solutions far away from $PF_{true}$. 
4.4. Analysis

(a) Spread values obtained during 50 runs (lower is better).

(b) Spacing values obtained during 50 runs (lower is better).

(c) GD values obtained during 50 runs (lower is better).

(d) Histogram of the solutions found by GDE3 over 50 runs. $PF_{true}$ is marked with orange.

(e) Average execution times in seconds of the ZDT4 function for each MOEA.

Figure 4.6: GDE3 obtains the best results on all metrics for ZDT3.
NSGA-II showed really average results on both Spacing and Spread metrics. All the scores obtained were really consistent as the variance between runs was much smaller than the ones seen in other MOEA. In Fonseca’s second, the DTLZ and WFG problems with three objectives, NSGA-II scored low on the Spacing metric due to not generating even distributions. NSGA-II was never found in the top performing algorithms in any of the tests nor the worst one, falling always on the average of the scores consistently. The same behaviour was observed for the convergence metrics as NSGA-II received mediocre scores on all tested MOPs, for example in Schaffer’s first function, seen in Figure 4.2. However, according to the Hypervolume metric in Figure 4.3(a), NSGA-II is one of the worst performing MOEA for this same MOP, indicating that the overall quality of the set might not be as high as indicated in the other metrics. Lastly, NSGA-II seems to generate an over-representation of solutions exactly at the individual minimums of each objective function, as seen for example if Figure 4.7(d).

OMOPSO and SMPSO have produced really promising results on the Spread and Spacing metrics. SMPSO seems to constantly obtain a better score on the Spread metric and place amongst the top MOEA, while OMOPSO obtains average scores on it (see for example Figure 4.14(a)). Regarding the Spacing metric, OMOPSO seems to produce really irregular results, sometimes obtaining a better than average result (see Figure 4.13(b)) while in other cases scoring amongst the worst algorithms. A behaviour similar to the one described in MOCell and CellDE was observed: when one of the MOEA would get better than average scores in the Spacing and Spread metrics, the other would then obtain mediocre results. For example, in Figure 4.10, OMOPSO obtains the best GD score amongst MOEA but received the worse diversity scores. On the other hand, SMPSO obtains the best diversity scores and average convergence results. An example can be seen in Figure 4.2 or Figure 4.7, where SMPSO receives favourable diversity scores but poor convergence scores, and OMOPSO produces just the contrary: better than average convergence but a poor distribution. It is interesting to see that despite the difference in performance,
4.4. Analysis

(a) Spread values obtained during 50 runs (lower is better).

(b) Spacing values obtained during 50 runs (lower is better).

(c) GD values obtained during 50 runs (lower is better).

(d) Histogram of the solutions found by NSGA-II over 50 runs.

Figure 4.7: NSGA-II obtains average results on all metrics for WFG5 with two objectives. The distribution is still quite good albeit skewed towards the minimum of each objective function.
the Hypervolume metric assigns similar scores to both MOEA in Figure 4.3(a) for Schaffer’s first function. A comparison of the distributions of solutions for WFG5 with two objectives can be seen in Figure 4.8.

OMOPSO also produced very inconsistent results regarding the convergence metrics, sometimes scoring really good scores with low variance (see Figure 4.10(c)) and in other situations obtaining the worst scores. On the other hand, SMPSO received average scores on the convergence metrics most of the time albeit also having problems solving three-objective WFG’s problems, as seen in Figure 4.13(c). This leaves us to conclude that OMOPSO produces really irregular results which are highly dependant on the MOP being solved, while SMPSO produces distributions which balance a good convergence rate and a high quality distribution.

Figure 4.8: The distribution of solutions of SMPSO is much better than the one obtained from OMOPSO, although it has a much better convergence as indicated by Figure 4.7(c). In the same Figure 4.7, Spread and Spacing values for both MOEA can be seen. $PF_{true}$ is marked with orange.

MOEA/D performed really poorly on both Spread and Spacing metrics, as shown for example in Figure 4.2(a). Moreover, the algorithm was not consistent as the variances of the results obtained in the Spacing metric were larger than average.
Convergence metrics also indicated that MOEA/D performs worse than the average score received by all MOEA in each MOP. In Figure 4.7, for example, shows that MOEA/D achieved worse than average Spread and Spacing scores, while scoring a high (i.e. worse) GD value. The histogram of solutions for this MOP can be found in Figure 4.9. Lastly, due to not having an explicit diversity-preserving mechanism, MOEA/D has shown problems distributing solutions on MOPs with disconnected $PF_{true}$ like Kursawe.

**Figure 4.9:** Histogram of solutions found by MOEA/D on the WFG5 problem with two objectives. Compared with SMPSO's distribution in Figure 4.8(b), the distribution is slightly unbalanced towards minimums of each objective function. Metrics results for this problem can be found in Figure 4.7.

IBEA's generated distributions were amongst the worst seen (see 4.10(a)). The algorithm consistently scored poorly in the Spread and Spacing metrics. Moreover, the variance seen on the values throughout the multiples runs was always greater than average, showing that IBEA produces inconsistent results. Histograms of the distributions showed that IBEA is heavily skewed towards generating solutions on the minimums of each objective functions, as seen on Figure 4.10(d). However, the algorithm was consistently ranked as one of the best in terms of convergence, as seen for example in Figure 4.14(c). IBEA also has shown problems distributing solutions among disconnected areas of the $PF_{true}$ in problems like Kursawe.
As described in Section 2.3.2.8, IBEA uses the Hypervolume metric to guide the search. Thus, IBEA scored highly constantly in this particular metric despite providing low quality distributions. However, as stated in Section 3.3, the Hypervolume is a computationally expensive metric to compute, thus making it one of the slowest MOEA. For example, in Figure 4.10(e), IBEA ranks amongst the slowest algorithms when running against Fonseca’s first function. Spending so many resources to only obtain solutions slightly closer to $PF_{true}$ shows that further development of this MOEA is needed.

**SPEA2** obtained remarkable scores on both Spread and Spacing metrics, ranking consistently amongst the top algorithms according to these metrics. Moreover, on the majority of MOP tested, the variance of the results was really low, indicating that the algorithm provides consistent results. Surprisingly, SPEA2 consistently provided the best distribution of solutions for the DTLZ and WFG three-objective problems, as seen for example in Figure 4.13. However, SPEA2 did not show high convergence rates on any of the MOPs. The MOEA obtained consistent average values in the GD metric, a similar behaviour also seen in NSGA-II. An example can be seen in Figure 4.10 where SPEA2 did score highly on the Spread and Spacing metrics on Fonseca’s first function, but did not do as good on the GD metric. The resulting distribution of the MOP can be seen in Figure 4.11.

SPEA2 and NSGA-II were released on approximately the same time-frame (the year 2001). However, NSGA-II ended-up being considered the "standard" MOEA by the research community up to this day, while SPEA2 did not receive as much attention. Although producing uniform distributions and slightly better results than NSGA-II, SPEA2 showed some of the slowest computational times across all MOEA, as seen for example in Figure 4.10(e), Figure 4.6(e) or Figure 4.5(b). Thus, it can be inferred that the superior performance yielded by NSGA-II is the main reason that drove SPEA2 out of the spotlight.
4.4. Analysis

(a) Spread values obtained during 50 runs (lower is better).

(b) Spacing values obtained during 50 runs (lower is better).

(c) GD values obtained during 50 runs (lower is better).

(d) Solutions found by IBEA over 50 runs on Fonseca’s first MOP. $PF_{true}$ is marked in orange.

(e) Average time of a run in seconds of Fonseca’s first function.

Figure 4.10: IBEA obtains poor Spacing and Spread scores while obtaining high GD values. Figure (d) shows an unbalanced distribution. Figure (e) shows that IBEA is amongst the slowest MOEA.
4.4. Analysis

Figure 4.11: SPEA2 shows a much better distribution than IBEA (Figure 4.10(d)) while obtaining top results in the Spread and Spacing metrics, although it shows mediocre convergence. The box plots for the metrics can be found in Figure 4.10.

DBEA, although being the newest MOEA tested, obtained really mediocre scores on the diversity metrics, as seen for example in Figure 4.1(a) or 4.4(a). Moreover, it showed really irregular results between MOPs, sometimes achieving better than average scores and others placing amongst the worst MOEA. Multiple times DBEA also obtained poor spacing scores (see Figure 4.2(b)) suggesting that the algorithm has difficulties generating uniform distributions. Regarding the convergence of the algorithm, DBEA performed really poorly in comparison to other older MOEA, achieving less than average results in all the test cases. DBEA also showed higher than average execution times throughout all MOPs, as shown for example in Figure 4.10(e).

The distribution of solutions generated by DBEA in Fonseca’s first function is shown in Figure 4.12. The distribution, albeit better than the one gathered with IBEA shown in Figure 4.10(d), still does not produce enough solutions in the middle of the front. DBEA was one of the worst MOEA in terms of convergence, as shown in Figure 4.10(c), while receiving average scores on the Spacing and Spread metrics.

NSGA-III showed slightly better Spread scores than DBEA. However, the Spread values were, except in WFG5-7 problems with three objectives (see Figure 4.13),
really mediocre, always falling within the average. Using reference points in order to divide the search space into sub-problems, as explained in Section 2.3.2.10, seemed to produce poor Spacing scores, indicating that the distributions were not uniform. Convergence wise, NSGA-III obtained similar scores to NSGA-II, indicating that the added strategy does not improve convergence at all (see Figure 4.7). Lastly, NSGA-III had problems distributing solutions along disconnected areas of the $PF_{true}$ of the DTLZ7 and Schaffer 2 problems.

The added computational time resulting from the modified selection operator does not seem to justify the slight improvements in the diversity scores that NSGA-III receives with respect to NSGA-II.

**PESA-II** showed really poor performance on both Spacing and Spread metrics, never surpassing the average score on each MOP. On the other hand, convergence rates on PESA-II were amongst the best seen throughout all MOPs. For example, in Figure 4.13 or Figure 4.14, PESA-II is shown to receive poor Spread scores and exceptional convergence scores. However, as seen in Figure 4.13(d), PESA-II showed high execution times when running against MOPs with three objectives, indicating that the MOEA has difficulty handling problems with more than two
objectives. Lastly, the algorithm seemed to favour generating scores away from the objective function’s individual minimum.

**Figure 4.13:** PESA-II achieves remarkable convergence scores on WFG7 with three objectives. However, the scores of the Spread metric are amongst the worst and the Spacing scores are inconsistent. PESA-II’s execution time is really high for three-objective MOPs, as seen in Figure (e).

PAES, the second oldest MOEA tested, provided competitive results despite being one of the first developed MOEA. In both convergence and diversity metrics, PAES showed results with huge variances, causing the algorithm to sometimes achieve better than average scores and worse than average scores at the same time. The results show that the algorithm is unreliable as it is, but with potential to achieve
well distributed $PF_{\text{known}}$. However, the median of the scores achieved by PAES is always worse than average results of the other MOEA, as seen for example in Figure 4.13. An example of this behaviour can be seen in Figure 4.14, where the Spread and Spacing scores obtained by PAES are really high while also varying greatly. Moreover, PAES obtains really irregular results on the GD metric over the tested runs.

Judging by the histograms of $PF_{\text{known}}$ found by PAES, the algorithm has problems converging on complicated problems and thus, clumps solutions in easy to reach areas of the search space. This results in disconnected areas of the front not being fully reached in some MOPs.

Additionally, PAES showed high execution times on MOPs with three objectives. This might mean that PAES might not be a good choice to solve MOPs with an arbitrary number of objectives, as seen for example in Figure 4.13(e). Otherwise, PAES reports average execution times in bi-objective MOPs.

$\epsilon$-MOEA behaved similarly to PESA-II: both Spread and Spacing metrics scored average results, never showing uniform distributions in any of the tested MOPs. However, $\epsilon$-MOEA showed exceptional convergence rates, scoring consistently among the best algorithms in the GD metric. Moreover, the convergence scores obtained showed low variance. This behaviour can be observed in Figure 4.10 and Figure 4.15, where $\epsilon$-MOEA obtained the best GD score while showing poor performance in both diversity metrics. Similarly to $\epsilon$-NSGA-II, the algorithm seems to tend to generate solutions away from the minimums of the individual minimums of the objective functions.

VEGA, as expected, did not perform well in any of the tests and, as previously stated, had to be removed from the presented box plots in order to improve the visualisation of the results. VEGA, as explained in Section 2.3.2.1 is the first MOEA ever developed, and thus showed problems converging to the optimal solutions, resulting in distributions clumped in other areas of the search space far away from $PF_{\text{true}}$. 
Figure 4.14: PAES shows great variances on the GD, Spread and Spacing metrics on the DTLZ1 problem with two objectives. However, PAES showed acceptable execution times, as seen in Figure (e). $PF_{true}$ is marked orange on Figure (d).
As a result, VEGA provided the worst scores across all MOPs. In Figure 4.16 an example of the solutions generated by VEGA on the ZDT3 problem is shown. It can be clearly observed that VEGA did not converge to the optimal solution as MOCell and CellDE did in Figure 4.4. Not converging results in a solution set with low diversity of solutions.

Figure 4.15: \( \epsilon \)-MOEA shows poor diversity due to clumping the solutions in areas away from the minimum of each objective function on Fonseca’s first function. However, as seen in Figure 4.10, \( \epsilon \)-MOEA obtained the second best convergence score. \( PF_{true} \) is marked in orange.

Figure 4.16: VEGA did not fully converge to \( PF_{true} \) of the ZDT3 problem, resulting in an uneven distribution of solutions.
Chapter 5

Conclusions

In this report, a comparison between various MOEA was performed. The objectives outlined in Section 1 were all accomplished:

• In Section 2.3.2 a selection of popular and state-of-the-art MOEA were presented. The MOEA were chosen based on the availability of their implementation and their diversity preserving mechanism. These MOEA were: NSGA-II, DBEA, MOEA/D, GDE3, OMOPSO, SMPSO, SPEA2, VEGA, $\epsilon$-MOEA, $\epsilon$-NSGAII, AbYSS, PAES, PESA-II, MOCell, CellDE, IBEA, CMA-ES and NSGA-III.

• In Section 3.1 MOEAFramework, the software supporting the implementation of the selected MOEA, was presented alongside the considered alternatives.

• In Section 3.2 the selected Multi-Objective Problems were presented.

• In Section 3.3 the Spread and Spacing metrics were presented alongside the Hypervolume and Inverted Generational Distance metrics. While the first two measure the diversity of a solution set, the second ones also try to estimate the convergence at the same time. In Section 3.3.2.1, the algorithm to implement the Spread metric into MOEAFramework was presented.

• In Section 3.3.2 the previously introduced diversity metrics were evaluated in order to test their accuracy. By tuning the parameters of NSGA-II, MOEA/D, GDE3 and IBEA algorithms against the Spread and Spacing metrics improve-
ments in the distribution of the found solutions were observed, assuring us that both metrics are able to identify diverse distributions.

- In Section 4.1 the program used to automate the testing of the MOEA was presented. The software automatically generates box plots and histograms of the $PF_{true}$ found by the MOEA in order to ease the process of comparison.

- A summary of the comparisons obtained through running the selected MOEA against the chosen MOPs is presented in Section 4.4. When appropriate, biases of each MOEA have been highlighted alongside MOPs which the MOEA had difficulty solving.

From the previously mentioned analysis of the results the following conclusions were gathered:

- $\epsilon$-MOEA, $\epsilon$-NSGA-II and IBEA generated really poor distributions across all the tested problems. This leads us to conclude that selecting solutions based on a $\epsilon$-dominance criterion or using a metric such as the Hypervolume in the case of IBEA to guide the search favours convergence versus diversity.

- CMA-ES and MOEA/D, although using novel approaches, achieved mediocre results on the diversity metrics and did not produce better distributions than NSGA-II, which is considered to be the baseline. CMA-ES saw extremely high execution times on many problems and, as a result, did not fully converge in many MOPs.

- NSGA-II produced average but consistent distributions with really low execution times. SPEA-2, on the other hand, produced much better results than NSGA-II on diversity as well as convergence. However, the execution times of SPEA2 are much higher than the average.

- PESA-II produced consistently poor distributions throughout all MOPs. However, the convergence scores of the algorithm, which were amongst the best, are not able to justify the extremely high execution times on MOPs with three objectives.
• PAES and VEGA, the two oldest tested MOEA, are known to be outdated and achieved extremely poor diversity results across all MOPs. While VEGA’s results show that in the majority of the cases no true Pareto Optimal solutions are found, PAES produces high variances across all metrics, indicating that the approach is not robust enough.

• DBEA and NSGA-III, albeit being the most recently released MOEA out of the ones selected, produced only slightly better than average results, not justifying their high computational times. Both MOEA are explicitly designed to solve MOPs with more than three objectives, which other MOEA might have problems with, and thus perform poorly on the bi-objective and tri-objective MOPs tested.

• GDE3 showed really promising results by achieving the top scores on the diversity metrics on some of the tested MOPs.

• AbYSS proved to be a robust algorithm, obtaining better than average distribution scores across the majority of MOPs.

• OMOPSO and SMPSO, albeit showing inconsistent results throughout MOPs (in some occasions achieving top scores while in others scoring amongst the worst MOEA), showed potential in generating high quality distributions. In particular, SMPSO did not score very high on convergence metrics but achieved the best distributions in many MOPs.

• CellDE and MOCell results were also unpredictable across the MOPs. CellDE performed, on average, worse than MOCell although CellDE is supposed to improve on the latter. MOCell produced uniform distributions in some of the MOPs, however, it did not achieve a better performance than NSGA-II in many other MOPs.

We expected that MOEA with similar diversity preserving mechanisms would perform similarly. However, we can gather from the results that this is not the case. For example, CellDE uses SPEA2’s diversity estimator but does not see the same
success as the later. Nonetheless, the diversity mechanisms used by PAES, PESA-II, MOEA/D (which, incidentally, uses none) and \(\epsilon\)-MOEA and IBEA, each of them described in Section 2.3.2, have been shown to perform worse than the other tested strategies and should be avoided.

Regarding the diversity metrics, our introductory statement suggested the usage of a single metric to measure the quality of the distribution of solutions. However, multiple metrics ended-up being used. We can conclude that measuring the quality of a solution set might be even a more difficult task than MOEA design itself due to the difficulties found when mapping information about a solution set into a single number.

From the tests conducted in Section 3.3.2.2 we could gather that both Spacing and Spread metrics are able to accurately assess the quality of a distribution of solutions. Although the Spacing metric only assesses the uniformity of the distances between solutions rather than the spread of the set, both characteristics are correlated and thus, both metrics produced, in most cases, similar judgements. Nonetheless, the Spread metric is able to convey more information about the distribution and thus, was given more importance when comparing the MOEA.

Lastly, rather than seeing an increase or decrease in a metrics’s value when tuning a MOEA against it, changes in the variance were observed. In other words, tuning the algorithm produced more consistent results rather than lower values.

5.1 Future Work

Possible experiment extensions could include the DTLZ and WFG test functions with a higher number of objectives, as only MOPs with two and three objectives were tested.

Additionally, transforming MOPs with rotation matrices allows us to create a problem with similar characteristics to the original but with increased difficulty. Thus, testing the selected MOPs with rotations might bring further insight into the performance of MOEA.

Other more recently introduced test suites such as the CEC2009 [51] and
BBOB2016 [52] have been published and could be included. Additionally, the inclusion of constrained problems such as the C-DTLZ test functions [53] could be considered in order to widen the problem pool even more and observe the behaviour of MOEA in constrained spaces.

More versions and configurations of the tested MOEA could be performed. For example, CMA-ES allows the usage of any type of quality indicator to select solutions. In our tests, NSGA-II’s crowding distance indicator was used, but the Hypervolume indicator or $I_x$ are also suggested in the introductory paper. Similarly, other indicators could have been used instead of the Hypervolume metric in IBEA. Furthermore, other versions of MOEA/D such as MOEA/DFD [54] and MOEA/DD [53] could also be tested.

Other promising MOEA left out due to having problems with their implementations (see Section 4.2) such as HypE and SMS-EMOA could also be included.

NSGA-III was a last addition to the collection of MOEA tested due to the implementation of the algorithm not being fully functioning \(^1\). However, NSGA-III was included in order to identify if the strategy used by the algorithm outperforms significantly any of the other tested algorithms. When a complete implementation of NSGA-III is available, tests should be re-run.

Using a binary indicator, although more time consuming, might also provide accurate answers to decide if a MOEA is "better" than another one. Binary metrics would also eliminate the need to use multiple indicators to measure different characteristics of the set.

Lastly, the Spread metric was implemented using Euclidean distances (see Algorithm 1). It is suggested in the literature [5, p. 328] that Manhattan distances or the crowding distance used in NSGA-II (see Section 2.3.2.2) could be used instead. Implementations using this distances could be tested as well in order to possibly strengthen the accuracy of the metric.

\(^1\)https://www.researchgate.net/post/Is_there_a_fully_functional_NSGA-III_implementation
Appendix A

Appendices

A.1 User Manual

This section gives a brief overview of the code written throughout the project, along with its location. Indications on how to compile and run the code are also given.

The code for the project is found in two separate code bases: modifications made to the source of MOEAFramework are found in the directory MOEAFramework/, while the rest of the code used to perform the tests and build the plots is found in the directory comparisons.

A.1.1 MOEAFramework/

The MOEAFramework directory contains all the code and tests created to incorporate the Spread metric into MOEAFramework. The code, also available in its GitHub repository¹, can be compiled with:

$ ant build-binary

The command, which requires Ant², will produce a in the folder MOEAFramework/dist/ a file named MOEAFramework-2.9.jar, which can be copied to the directory comparisons/lib/ in order to be used in the tests. When running any of the tests, the Spread metric will be automatically included in the results.

To compile and run the Unit Tests use the command:

¹https://github.com/Gan0k/MOEAFramework
²https://ant.apache.org/
$ ant run-tests

The code relevant for the Spread metric can be found on the Appendix A.4.1 or in the folder:

MOEAFramework/src/org/moeaframework/core/indicator/
GeneralizedSpread.java

and the Unit Tests on:

MOEAFramework/test/org/moeaframework/core/indicator/
GeneralizedSpreadTest.java

All the code changes to include the Spread metric can be viewed in commit d9ef5³. The rest of the code found in the folder has not been developed in this project.

A.1.2 Comparisons/

The folder comparisons contains all the programs and scripts used to tune the MOEA and generate the plots and data to support the comparisons. All the code is mirrored in its git repository⁴. Its contents are the following:

- pf/ Contains all the $P_{F_{true}}$ from the tested MOP. These files were obtained from MOEAFramework.
- lib/ Contains all the .jar files which MOEAFramework depends on.
- tests/EvalDefParams.java: Program to generate a solution set of a MOEA with the default parametrisation and evaluate it using all the metrics supported by MOEAFramework. Run the program with the -h flag to see its usage.
- tests/Gen_PFss.java: Program to print the $P_{F_{known}}$ found by a given MOEA. Run the program with the -h flag to see its usage.

³https://github.com/Gan0k/MOEAFramework/commit/d9ef5
⁴https://bitbucket.org/Ganok/moea-analysis/
• tests/plot_funcs.m: MATLAB script to visualise Fonseca’s first and second function and Schaffer’s first function using three-dimensional plots.

• tests/plot_tuning.m: MATLAB script to plot the histograms of the $PF_{known}$ found in the tests/param_tuning/ folder generated after tuning a MOEA using the programs found in the same folder.

• tests/plotHisto.m: MATLAB script to plot the histograms of the $PF_{known}$ found in the tests/compare/sets folder generated after running each MOEA against the selected MOPs with the CompareDef.java program. The generated plots are saved in tests/compare/plot_sets.

• tests/plotTimes.m: MATLAB script to generate the bar plots out of the average execution times found in tests/compare/times, created through the CompareDef.java program. The generated plots are saved in tests/compare/plot_times.

• tests/param_tuning/{IBEAParam.java, GDE3Param.java, MOEADParam.java, NSGAIIParam.java, SPEA2Param.java}: Programs used to tune the respective MOEA and print the $PF_{known}$ generated by the MOEA over the given runs along with a summary of the scores achieved in the selected metrics. Run the programs with the `-h` flag to see its usage. The code of NSGAIIParam.java can be found in Appendix A.4.2.

• tests/param_tuning/create_pic.sh: Bash script used to generate a side-by-side image of the plots of a tuned and untuned MOEA. The script has to be run on the root of the project (comparisons/ folder) and requires the imagemagick\(^5\) software package to be installed. To run the script with NSGA-II against Schaffer’s first function, for example, use the following command:

```
$ ./create_pic.sh NSGAII Schaffer -sbxr 1.0 \ 
   -sxbd 20.0 -pmdx 1.0
```

\(^5\)https://www.imagemagick.org/
• tests/param_tuning/histograms/: Contains all of the side-by-side plots generated when tuning the MOEA using the create_pic.sh. These plots are used in Section 3.3.2.3.

• tests/compare/CompareDef.java: Program used to generate the box plots, $PF_{known}$ found and execution time data produced when testing the selected MOEA. Run the program with the -h flag to see its usage. The source code can be found in Appendix A.4.3.

• tests/compare/alg_config/: Contains the parametrisation of the MOEA to be used when running the tests.

• tests/compare/analysis/: Contains the summaries of the analysis of the results obtained by the MOEA when running the tests with the CompareDef.java program. This data is used to plot the box plots found in the folder tests/compare/boxplots/.

• tests/compare/sets/: Raw data containing the $PF_{known}$ generated by each MOEA when run against each MOP with the CompareDef.java program. These files are not included in the .zip file attached.

• tests/compare/time/: Raw data containing the execution times achieved by each MOEA when run with the CompareDef.java program.

All the aforementioned code can be compiled with the following command, which will save all the .class files into the class/ folder:

```
$ javac -cp "lib/*" -d class tests/compare/*.java \
   tests/param_tuning/*.java tests/*.java
```

To run any of the classes use the following command and substitute the name of the class by any the program we wish to run along with its parameter flags:

```
$ java -cp "lib/*:class" $NAME_CLASS [$PARAMETERS]
```
A.2 MOEA Default Parametrisation

Table A.1 and A.2 contain the default parametrisation of the MOEA used in the comparisons.

A.3 Tuned Parameters

Table A.4 and A.5 contain the tuned parameters of each MOEA used to perform the tests in Section 3.3.2.2. Note that due to brevity, parameters which obtained worst results when modifying their default value are omitted. Thus, only parameters which saw changes from the ones given in Table A.1 are included.

A.4 Code Listings

This section contains the relevant code written for the project. Note that in Section A.4.2 only the class to tune NSGA-II is included as the other classes used to tune the rest of the MOEA only differ on the parameters relevant to each MOEA.

MATLAB scripts to produce plots (see Appendix A.1), test programs to easily visualise Pareto Fronts and shell scripts used to automate some task are not included due to being irrelevant to the project. Additionally, the implementation of the Spread metric can be found in the corresponding GitHub repository⁶, and the programs and plots used to compare the MOEA can be found in the following git repository⁷.

A.4.1 Spread Metric

```java
package org.moeaframework.core.indicator;

import org.apache.commons.math3.stat.StatUtils;
import org.moeaframework.core.NondominatedPopulation;
import org.moeaframework.core.Solution;
import org.moeaframework.core.comparator.ObjectiveComparator;
import org.moeaframework.core.comparator.LexicographicalComparator;
import org.moeaframework.core.Problem;

public class GeneralizedSpread extends NormalizedIndicator {

/* Generalized Spread metric. */

//
```

⁶https://github.com/Gan0k/MOEAFramework/
⁷https://bitbucket.org/Ganok/moea-analysis/
<table>
<thead>
<tr>
<th>MOEA</th>
<th>Parameter</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMA-ES</td>
<td>$\lambda$</td>
<td>100.</td>
<td>where $N$ is the number of decision variables</td>
</tr>
<tr>
<td></td>
<td>$c_c$</td>
<td>$\frac{2}{N+1}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$P_{\text{succ}}$</td>
<td>$\frac{5+\sqrt{12}}{2}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\text{damps}$</td>
<td>$1 + \frac{N}{2\lambda}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$c_{\text{cov}}$</td>
<td>$\frac{N^2+6}{P_{\text{succ}}\lambda}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$c_p$</td>
<td>$\frac{2+P_{\text{succ}}\lambda}{\lambda}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\sigma$</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>diagonalIterations</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>indicator</td>
<td>Crowding Distance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>initialSearchPoint</td>
<td>Unset</td>
<td></td>
</tr>
<tr>
<td>DBEA</td>
<td>population Size</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>divisions</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sbx.rate</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sbx.distributionIndex</td>
<td>15.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pm.rate</td>
<td>$1/N$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pm.distributionIndex</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>$\epsilon$-MOEA</td>
<td>populationSize</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\epsilon$</td>
<td>Problem dependent</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sbx.rate</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sbx.distributionIndex</td>
<td>15.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pm.rate</td>
<td>$1/N$ where $N$ is the number of decision variables</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pm.distributionIndex</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>$\epsilon$-NSGAII</td>
<td>population Size</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>epsilon</td>
<td>Problem dependent</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sbx.rate</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sbx.distributionIndex</td>
<td>15.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pm.rate</td>
<td>$1/N$ where $N$ is the number of decision variables</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pm.distributionIndex</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>injectionRate</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>windowSize</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>maxWindowSize</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>minimumPopulationSize</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>maximumPopulationSize</td>
<td>10000</td>
<td></td>
</tr>
<tr>
<td>IBEA</td>
<td>population Size</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sbx.rate</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sbx.distributionIndex</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pm.rate</td>
<td>$1/N$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pm.distributionIndex</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>indicator</td>
<td>hypervolume</td>
<td></td>
</tr>
</tbody>
</table>

Table A.1: Parametrisation of the MOEA.
<table>
<thead>
<tr>
<th>MOEA</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOEA/D</td>
<td>populationSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>de.crossoverRate</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>de.stepSize</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>pm.rate</td>
<td>1/N</td>
</tr>
<tr>
<td></td>
<td>pm.distributionIndex</td>
<td>20.0</td>
</tr>
<tr>
<td></td>
<td>neighborhoodSize</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>delta</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>eta</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>updateUtility</td>
<td>Unset</td>
</tr>
<tr>
<td>NSGA-II</td>
<td>populationSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>sbx.rate</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>sbx.distributionIndex</td>
<td>15.0</td>
</tr>
<tr>
<td></td>
<td>pm.rate</td>
<td>1/N</td>
</tr>
<tr>
<td></td>
<td>pm.distributionIndex</td>
<td>20.0</td>
</tr>
<tr>
<td>NSGA-III</td>
<td>populationSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>divisions</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>sbx.rate</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>sbx.distributionIndex</td>
<td>15.0</td>
</tr>
<tr>
<td></td>
<td>pm.rate</td>
<td>1/N</td>
</tr>
<tr>
<td></td>
<td>pm.distributionIndex</td>
<td>20.0</td>
</tr>
<tr>
<td>OMOPSO</td>
<td>populationSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>archiveSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>maxEvaluations</td>
<td>25000</td>
</tr>
<tr>
<td></td>
<td>mutationProbability</td>
<td>1/N</td>
</tr>
<tr>
<td></td>
<td>perturbationIndex</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>epsilon</td>
<td>Problem dependent</td>
</tr>
<tr>
<td>PAES</td>
<td>archiveSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>bisections</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>pm.rate</td>
<td>1/N</td>
</tr>
<tr>
<td></td>
<td>pm.distributionIndex</td>
<td>20.0</td>
</tr>
<tr>
<td>PESA-II</td>
<td>populationSize</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>archiveSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>bisections</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>sbx.rate</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>sbx.distributionIndex</td>
<td>15.0</td>
</tr>
<tr>
<td></td>
<td>pm.rate</td>
<td>1/N</td>
</tr>
<tr>
<td></td>
<td>pm.distributionIndex</td>
<td>20.0</td>
</tr>
</tbody>
</table>

Table A.2: Parametrisation of the MOEA.
## Table A.3: Default parametrisation of the MOEA.

<table>
<thead>
<tr>
<th>MOEA</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPEA-2</td>
<td>populationSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>offspringSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>k</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>sbx.rate</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>sbx.distributionIndex</td>
<td>15.0</td>
</tr>
<tr>
<td></td>
<td>pm.rate</td>
<td>1/N</td>
</tr>
<tr>
<td></td>
<td>pm.distributionIndex</td>
<td>20.0</td>
</tr>
<tr>
<td>VEGA</td>
<td>populationSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>sbx.rate</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>sbx.distributionIndex</td>
<td>15.0</td>
</tr>
<tr>
<td></td>
<td>pm.rate</td>
<td>1/N</td>
</tr>
<tr>
<td></td>
<td>pm.distributionIndex</td>
<td>20.0</td>
</tr>
<tr>
<td>AbYSS</td>
<td>populationSize</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>archiveSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>refSet1Size</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>refSet2Size</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>improvementRounds</td>
<td>1</td>
</tr>
<tr>
<td>CellDE</td>
<td>populationSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>archiveSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>feedBack</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>de.crossoverRate</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>de.stepSize</td>
<td>0.5</td>
</tr>
<tr>
<td>MOCell</td>
<td>populationSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>archiveSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>feedBack</td>
<td>20</td>
</tr>
<tr>
<td>SMPSO</td>
<td>populationSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>archiveSize</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>pm.rate</td>
<td>1/N</td>
</tr>
<tr>
<td></td>
<td>pm.distributionIndex</td>
<td>20.0</td>
</tr>
<tr>
<td>GDE3</td>
<td>population Size</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>de.crossoverRate</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>de.stepSize</td>
<td>0.5</td>
</tr>
<tr>
<td>MOEA</td>
<td>MOP</td>
<td>Parameter</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>NSGA-II</td>
<td>Schaffer</td>
<td>pm.distributionIndex</td>
</tr>
<tr>
<td></td>
<td>Schaffer2</td>
<td>pm.distributionIndex</td>
</tr>
<tr>
<td></td>
<td>Fonseca</td>
<td>sbx.distributionIndex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pm.distributionIndex</td>
</tr>
<tr>
<td></td>
<td>DTLZ1_2</td>
<td>sbx.distributionIndex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pm.distributionIndex</td>
</tr>
<tr>
<td></td>
<td>DTLZ3_2</td>
<td>sbx.distributionIndex</td>
</tr>
<tr>
<td>MOEA/D</td>
<td>Schaffer</td>
<td>pm.distributionIndex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>delta</td>
</tr>
<tr>
<td></td>
<td>Schaffer2</td>
<td>de.crossoverRate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pm.distributionIndex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>delta</td>
</tr>
<tr>
<td></td>
<td>Fonseca</td>
<td>de.crossoverRate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pm.distributionIndex</td>
</tr>
<tr>
<td></td>
<td>DTLZ1_2</td>
<td>sbx.distributionIndex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>delta</td>
</tr>
<tr>
<td></td>
<td></td>
<td>eta</td>
</tr>
<tr>
<td></td>
<td>DTLZ2_2</td>
<td>de.crossoverRate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pm.distributionIndex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>eta</td>
</tr>
<tr>
<td></td>
<td>DTLZ4_2</td>
<td>de.crossoverRate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>delta</td>
</tr>
<tr>
<td></td>
<td>DTLZ7_2</td>
<td>de.stepSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pm.distributionIndex</td>
</tr>
<tr>
<td>IBEA</td>
<td>Schaffer</td>
<td>pm.distributionIndex</td>
</tr>
<tr>
<td></td>
<td>Schaffer2</td>
<td>pm.distributionIndex</td>
</tr>
<tr>
<td></td>
<td>Fonseca</td>
<td>sbx.rate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sbx.distributionIndex</td>
</tr>
<tr>
<td></td>
<td>DTLZ1_2</td>
<td>sbx.rate</td>
</tr>
<tr>
<td></td>
<td>DTLZ2_2</td>
<td>pm.distributionIndex</td>
</tr>
<tr>
<td></td>
<td>DTLZ3_2</td>
<td>sbx.rate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sbx.distributionIndex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pm.distributionIndex</td>
</tr>
<tr>
<td></td>
<td>DTLZ7_2</td>
<td>sbx.rate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pm.distributionIndex</td>
</tr>
</tbody>
</table>

Table A.4: Tuned parameters of each MOEA w.r.t each MOP.
<table>
<thead>
<tr>
<th>MOEA</th>
<th>MOP</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDE3</td>
<td>Schaffer</td>
<td>de.stepSize</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Schaffer2</td>
<td>de.rate</td>
<td>0.5</td>
</tr>
<tr>
<td>DTLZ1_2</td>
<td>de.rate</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DTLZ4_2</td>
<td>de.stepSize</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>de.rate</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table A.5: Tuned parameters of each MOEA w.r.t each MOP. Continued from Table A.4.

```java
/**
 * Constructs a maximum Pareto front error evaluator for the specified
 * problem and corresponding reference set.
 *
 * @param problem the problem
 * @param referenceSet the reference set for the problem
 */
public GeneralizedSpread(Problem problem, NondominatedPopulation referenceSet) {
    super(problem, referenceSet);
}

@Override
public double evaluate(NondominatedPopulation approximationSet) {
    return evaluate(problem, normalize(approximationSet),
                     getNormalizedReferenceSet());
}

/**
 * Computes the maximum Pareto front error for the specified problem given
 * an approximation set and reference set. While not necessary, the
 * approximation and reference sets should be normalized. Returns @code
 * Double.POSITIVE_INFINITY if the approximation set is empty.
 *
 * @param problem the problem
 * @param approximationSet an approximation set for the problem
 * @param referenceSet the reference set for the problem
 * @return the generational distance for the specified problem given an
 *         approximation set and reference set
 */
static double evaluate(Problem problem, NondominatedPopulation approximationSet, NondominatedPopulation referenceSet) {
    if (approximationSet.isEmpty()) {
```
return 2.0;
}

int numberOfObjectives =
    approximationSet.get(0).getNumberOfObjectives();

Solution[] extremeValues = new Solution[numberOfObjectives];
for (int i = 0; i < numberOfObjectives; i++) {
    referenceSet.sort(new ObjectiveComparator(i));
    extremeValues[i] = referenceSet.get(referenceSet.size()-1);
}

approximationSet.sort(new LexicographicalComparator());

if (IndicatorUtils.euclideanDistance(problem, approximationSet.get(0),
    approximationSet.get(approximationSet.size()-1)) == 0.0) {
    return 2.0;
} else {

double[] distances = new double[approximationSet.size()];
for (int i = 0; i < approximationSet.size(); i++) {
    distances[i] =
        IndicatorUtils.distanceToNearestNeighbour(problem, i, approximationSet);
}

double meanDistance = StatUtils.mean(distances);

double distExtremes = 0.0;
for (int i = 0; i < extremeValues.length; i++) {
    distExtremes +=
        IndicatorUtils.distanceToNearestSolution(problem, extremeValues[i], approximationSet);
}

double normalizedSum = 0.0;
for (int i = 0; i < approximationSet.size(); i++) {
    normalizedSum += Math.abs(distances[i] - meanDistance);
}

return (distExtremes + normalizedSum) /
    (distExtremes + (approximationSet.size()*meanDistance));
}
A.4.2 Parameter tuning for NSGA-II

```java
import java.util.List;
import java.io.File;

import org.moeaframework.Executor;
import org.moeaframework.Analyzer;
import org.moeaframework.core.NondominatedPopulation;
import org.moeaframework.core.Solution;
import org.moeaframework.core.variable.EncodingUtils;

public class NSGAIIParam {

    private static void usage() {
        System.err.println("Usage: NSGAIIParam " +
                "-p PROBLEM_NAME -s POPULATION_SIZE " +
                "-m MAX_FUNCS_EVAL -r NUMBER_SEEDS " +
                "-sbxr SBX.RATE -sbxd SBX.DISTRIDX " +
                "-pmidx PM.DISTRIDX -printvar -printobj");
        System.exit(-1);
    }

    public static void main(String[] args) {
        String algName = "NSGAII";
        String problemName = "Schaffer";
        int seeds = 100, mfe = 10000, populSize = 100;
        double sbxrate = 1.0, sbxidx = 15.0, pmidx = 20.0;
        boolean printvar = false, printobj = false;

        // parse arguments to set options
        int n = args.length;
        for (int i = 0; i < n; ++i) {
            if (args[i].equals("-sbxr") && i+1 < n)
                sbxrate = Double.valueOf(args[++i]);
            else if (args[i].equals("-sbxd") && i+1 < n)
                sbxidx = Double.valueOf(args[++i]);
            else if (args[i].equals("-printvar"))
                printvar = true;
            else if (args[i].equals("-printobj"))
                printobj = true;
            else if (args[i].equals("-pmidx") && i+1 < n)
                pmidx = Double.valueOf(args[++i]);
            else if (args[i].equals("-p") && i+1 < n)
```
problemName = args[++i];
else if (args[i].equals("-s") && i+1 < n)
    populSize = Integer.parseInt(args[++i]);
else if (args[i].equals("-m") && i+1 < n)
    mfe = Integer.parseInt(args[++i]);
else if (args[i].equals("-r") && i+1 < n)
    seeds = Integer.parseInt(args[++i]);
else usage();
}

File f = new File("./tests/param_tuning/" + algName + "_
    problemName + "." + String.valueOf(seeds));

Analyzer analyzer = new Analyzer()
    .withProblem(problemName)
    .includeInvertedGenerationalDistance()
    .includeSpacing()
    .includeGeneralizedSpread()
    .includeR2()
    .includeAdditiveEpsilonIndicator()
    .includeHypervolume()
    .showStatisticalSignificance();

Executor newparams = new Executor()
    .withProblem(problemName)
    .withProperty("populationSize", populSize)
    .withProperty("sbx.rate", sbxrate)
    .withProperty("sbx.distributionIndex", sbxidx)
    .withProperty("pm.distributionIndex", pmidx)
    .withMaxEvaluations(mfe)
    .withAlgorithm(algName)
    .distributeOnAllCores();

Executor defaultparams = new Executor()
    .withProblem(problemName)
    .withProperty("populationSize", populSize)
    .withMaxEvaluations(mfe)
    .withAlgorithm(algName)
    .distributeOnAllCores();

List<NondominatedPopulation> result = newparams.runSeeds(seeds);

if (printvar) {
    for (NondominatedPopulation pop : result) {
        for (Solution solution : pop) {
            System.out.format("%.10f", \// decimals can modify results!};
A.4. Code Listings

88    EncodingUtils.getReal(solution.getVariable(0));
89    for (int i = 1; i < solution.getNumberOfVariables(); ++i) {
90        System.out.format("\t%.10f",
91                EncodingUtils.getReal(solution.getVariable(i)));
92    }
93    System.out.println();
94}
95}
96}
97}
98    else if (printobj) {
99        for (NondominatedPopulation pop : result) {
100            for (Solution solution : pop) {
101                System.out.format("%.10f",
102                        solution.getObjective(0));
103                for (int i = 1; i < solution.getNumberOfObjectives(); ++i) {
104                    System.out.format("\t%.10f",
105                        solution.getObjective(i));
106                }
107                System.out.println();
108            }
109        }
110        analyzer.addAll("New " + algName, result);
111        analyzer.addAll("Default " + algName, defaultparams.runSeeds(seeds));
112        try {
113            analyzer.saveAnalysis(f);
114        } catch (Exception e) {
115            e.printStackTrace();
116        }
117    }

A.4.3 Comparison of MOEA

import java.util.List;
import java.util.ArrayList;
import java.io.File;
import java.io.FileInputStream;
import java.io.PrintWriter;
import java.util.Properties;
import java.util.HashMap;
import java.util.Map;
import java.io.PrintStream;
```java
import org.moeaframework.Executor;
import org.moeaframework.Analyzer;
import org.moeaframework.core.NondominatedPopulation;
import org.moeaframework.core.Solution;
import org.moeaframework.core.variable.EncodingUtils;

import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.CategoryAxis;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.labels.BoxAndWhiskerToolTipGenerator;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.renderer.category.BoxAndWhiskerRenderer;
import org.jfree.data.statistics.BoxAndWhiskerCategoryDataset;
import org.jfree.data.statistics.DefaultBoxAndWhiskerCategoryDataset;
import org.jfree.chart.ChartUtilities;

public class CompareDef {

    private static void usage() {
        System.err.println("Usage: CompareDef " + "-m MAX_FUNCS_EVAL -r NUMBER_SEEDS ");
        System.exit(-1);
    }

    private static void printSolutions(List<NondominatedPopulation> result,
                                       String algName, String problemName, int seeds) {

        String objSol = "";
        String variableSol = "";
        boolean printvar = true, printobj = true;

        for (NondominatedPopulation pop : result) {
            for (Solution solution : pop) {
                if (solution.getNumberOfVariables() > 3) {
                    printvar = false;
                    break;
                }

                variableSol += String.valueOf(EncodingUtils.getReal(solution.getVariable(0)));
                for (int i = 1; i < solution.getNumberOfVariables(); ++i) {
                    variableSol += "\t" +
                                   String.valueOf(EncodingUtils.getReal(solution.getVariable(i)));
                }
                variableSol += "\n";
            }
        }
    }
```
for (NondominatedPopulation pop : result) {
    for (Solution solution : pop) {
        if (solution.getNumberOfObjectives() > 3) {
            printobj = false;
            break;
        }

        objSol += String.valueOf(solution.getObjective(0));
        for (int i = 1; i < solution.getNumberOfObjectives(); ++i) {
            objSol += " \t " + String.valueOf(solution.getObjective(i));
        }
        objSol += "\n";
    }
}

if (printobj) {
    try (PrintWriter out = new PrintWriter("./tests/compare/sets/" +
    problemName + "_" + algName + "_def_obj_" +
    String.valueOf(seeds) + ".sets")) {
        out.println(objSol);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

if (printvar) {
    try (PrintWriter out = new PrintWriter("./tests/compare/sets/" +
    problemName + "_" + algName + "_def_var_" +
    String.valueOf(seeds) + ".sets")) {
        out.println(variableSol);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private static void saveBoxplots(Analyzer problem, String problemName, int seeds) {
    Analyzer.AnalyzerResults results = problem.getAnalysis();
    HashMap<String, DefaultBoxAndWhiskerCategoryDataset> datasetmap =
        new HashMap<String, DefaultBoxAndWhiskerCategoryDataset>();

    for (String namealgo : results.getAlgorithms()) {
        // produces nicer boxplots
        if (namealgo.equals("VEGA")) continue;
    }
A.4. Code Listings

```java
Analyzer.AlgorithmResult algores = results.get(namealgo);
for (String nameindicator : algores.getIndicators()) {
    Analyzer.IndicatorResult indres = algores.get(nameindicator);
    List<Double> values = new ArrayList<Double>();
    for (double d : indres.getValues()) values.add(d);

    if (!datasetmap.containsKey(nameindicator)) {
        datasetmap.put(nameindicator,
            new DefaultBoxAndWhiskerCategoryDataset());
    }
    datasetmap.get(nameindicator).add(values, "",
        namealgo.equals("IBEA-JMetal") ? "IBEA":
            (namealgo.equals("OMOPSO") ? "omopso" : namealgo));
}
}
Analyzer.AlgorithmResult algores = results.get("NSGAII");
for (String nameindicator : algores.getIndicators()) {
    CategoryAxis xAxis = new CategoryAxis("Algorithms");
    xAxis.setLowerMargin(xAxis.getLowerMargin() * 0.2);
    xAxis.setUpperMargin(xAxis.getUpperMargin() * 0.2);
    NumberAxis yAxis = new NumberAxis(nameindicator + " Value");
    BoxAndWhiskerRenderer renderer = new BoxAndWhiskerRenderer();
    renderer.setMeanVisible(false);
    renderer.setUseOutlinePaintForWhiskers(true);
    yAxis.setAutoRangeIncludesZero(false);
    CategoryPlot plot =
        new CategoryPlot(datasetmap.get(nameindicator), xAxis, yAxis,
            renderer);
    JFreeChart chart = new JFreeChart(problemName + " " +
        nameindicator, plot);

    File img = new File("./tests/compare/boxplots/" + problemName +
        "_def_" + nameindicator + "_" + String.valueOf(seeds) +
        ".png");
    try {
        ChartUtilities.saveChartAsPNG(img, chart, 960, 640);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
public static void main(String[] args) {
    String[] algNames = {"NSGAII", "IBEA", "MOEAD", "GDE3", "OMOPSO"};
```
"SMPSO", "SPEA2", "VEGA", "eMOEA", "eNSGAII", "Abyss", "PAES",
"PESA2", "MOCell", "CellDE", "IBEA−JMetal", "CMA−ES", "NSGAII";

String [] problemNames = { "Schaffer", "Schaffer2", "Fonseca",
"Fonseca2", "Kursawe", "OKA1", "OKA2", "ZDT1", "ZDT2", "ZDT3",
"ZDT4", "ZDT5", "ZDT6", "DTLZ1_2", "DTLZ2_2", "DTLZ3_2",
"DTLZ4_2", "DTLZ7_2", "DTLZ1_3", "DTLZ2_3", "DTLZ3_3", "DTLZ4_3",
"DTLZ7_3", "WFG1_2", "WFG2_2", "WFG3_2", "WFG4_2", "WFG5_2",
"WFG6_2", "WFG7_2", "WFG8_2", "WFG9_2", "WFG1_3", "WFG2_3",
"WFG3_3", "WFG4_3", "WFG5_3", "WFG6_3", "WFG7_3", "WFG8_3",
"WFG9_3"};

Analyzer [] problems = new Analyzer[problemNames.length];
int seeds = 100, mfe = 10000;

// parse arguments to set options
int n = args.length;
for (int i = 0; i < n; ++i) {
    if (args[i].equals("-m") && i+1 < n)
        mfe = Integer.parseInt(args[++i]);
    else if (args[i].equals("-r") && i+1 < n)
        seeds = Integer.parseInt(args[++i]);
    else usage();
}

// initialize analyzers
for (int i = 0; i < problemNames.length; ++i) {
    problems[i] = new Analyzer()
        .withProblem(problemNames[i])
        .includeGenerationalDistance()
        .includeInvertedGenerationalDistance()
        .includeSpacing()
        .includeGeneralizedSpread()
        .includeR2()
        .includeAdditiveEpsilonIndicator()
        .includeHypervolume()
        .showStatisticalSignificance();
}

// store execution time of algorithms
HashMap<String,HashMap<String,Double>> avgTime =
    new HashMap<String,HashMap<String,Double>>();

// evaluate each algorithm for each problem
for (String algName : algNames) {
    Properties properties = new Properties();
}
try {
    File propfile = new File("./tests/compare/alg_config/" + algName + "_def.config");
    if (propfile.exists() && !propfile.isDirectory()) {
        FileInputStream inp = new FileInputStream(propfile);
        properties.load(inp);
        inp.close();
    }
}

} catch (Exception e) {
    e.printStackTrace();
}

for (int i = 0; i < problemNames.length; ++i) {
    // skip problems that do not support binary variables
    if (problemNames[i].equals("ZDT5") &&
        (algName.equals("GDE3") || algName.equals("MOEAD") ||
        algName.equals("SMPSO") || algName.equals("Abyss") ||
        algName.equals("CMA-ES") || algName.equals("OMOPSO") ||
        algName.equals("CellDE"))) continue;

    if (problemNames[i].equals("Fonseca") &&
        algName.equals("CMA-ES")) ||
    (problemNames[i].contains("WFG") &&
        problemNames[i].contains("_3") &&
        algName.equals("eMOEA")) ||
    (problemNames[i].contains("WFG") &&
        problemNames[i].contains("_3") &&
        algName.equals("CMA-ES")) ||
    (problemNames[i].contains("WFG") &&
        problemNames[i].contains("_3") &&
        algName.equals("eNSGAII"))) continue;

    Executor exec = new Executor()
        .withProblem(problemNames[i])
        .withMaxEvaluations(mfe)
        .withAlgorithm(algName)
        .distributeOnAllCores();

    // loop through set properties and set them
    Enumeration<? extends Property> e = properties.propertyNames();
    while (e.hasMoreElements()) {
        String key = (String)e.nextElement();
        exec.withProperty(key, properties.getProperty(key));
    }
}
```java
long start = System.nanoTime();
List<NondominatedPopulation> result = exec.runSeeds(seeds);
long elapsedTime = System.nanoTime() - start;
double seconds = ((double)elapsedTime / 1e9);
if (!avgTime.containsKey(problemNames[i]))
    avgTime.put(problemNames[i],
        new HashMap<String,Double>());
avgTime.get(problemNames[i]).put(algName, seconds / seeds);

printSolutions(result, algName, problemNames[i], seeds);
problems[i].addAll(algName, result);

System.out.println(algName + " " + problemNames[i] + " completed");
}

// save analysis
for (int i = 0; i < problemNames.length; ++i) {
    saveBoxplots(problems[i], problemNames[i], seeds);

    // save file analysis
    File f = new File("./tests/compare/analysis/" + problemNames[i] + 
        "_def_" + String.valueOf(seeds) + ".metrics");
    try {
        problems[i].saveAnalysis(f);
    } catch (Exception e) {
        e.printStackTrace();
    }

    // save file average elapsed time
    try (PrintWriter out = new PrintWriter("./tests/compare/time/" + 
        problemNames[i] + 
        "_def_" + String.valueOf(seeds) + 
        ".time") ){
        for (Map.Entry<String,Double> entry : avgTime.get(problemNames[i]).entrySet()) {
            out.println(entry.getKey() + " " + entry.getValue());
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

A.5 Project Plan
Project Title

Analysis of the Distribution of Pareto Optimal Solutions on various Multi-Objective Optimisation Genetic Algorithms

Aims

Evaluate the spread and diversity of solutions over the Pareto Front that the most popular multi-objective optimisation genetic algorithms produce. Analyse the characteristics of the distributions of this solutions and evaluate the evenness of each of them. Additionally, learn about state-of-the-art research in the multi-objective optimisation field and, more specifically, about the advancements achieved using genetic algorithms.

Objectives

1. What are the biases that each multi-objective genetic algorithm has when producing Pareto Optimal solutions?

2. How does a change of a parameter in the configuration of an algorithm affect the distribution of the solutions over the Pareto Front of a problem given by each of the analysed algorithms?

3. Can we control the areas over the Pareto Front where solutions will be generated by each algorithm by modifying its configuration?

4. How well the standard metrics for multi-objective genetic algorithms do on measuring the spread and evenness of solutions over the Pareto Front?

5. If none of the standard metrics succeed in producing a satisfactory estimation of the spread and evenness of solutions over the Pareto Front, a new metric needs to be developed. This new measure should not only allow us to compare the evenness of the distribution of solutions between algorithms, but also aid us in the tuning of the configuration of an algorithm to generate even distributions of solutions.

Expected Outcomes

1. Brief survey of the state-of-the-art multi-objective genetic algorithms, widely used test problems for this type of algorithms and the standard metrics used to measure the perfor-
2. Results of the experiments performed using multiple multi-objective genetic algorithms on various test functions with two sets of configuration parameters: the standard configuration described in the paper that introduced the algorithm and the ones that offer the best performance of a given metric.

3. In order to generate the previously mentioned results, various extensions to the MOEA Framework\(^1\) will be produced in addition to multiple scripts to automate the generation of the solutions.

4. Analysis of the solutions produced by the aforementioned experiments. The analysis will look at the evenness of the distributions of the solutions over the Pareto Front. Moreover, the distributions of each algorithm will be analysed in order to identify which areas of the Pareto Front the algorithms tend to focus the solutions on.

5. If the analysis does not show that a commonly used metric can be used to measure the evenness of a solution distribution, a new one will be developed.

**Work Plan**

- Project start to end of October (4 weeks) Literature search and review about multi-objective optimisation and genetic algorithms. Evaluation of the most popular test suites for this type of genetic algorithms and selection of the metrics to be tested.

- End October to mid-November (2 weeks). Familiarisation with the available implementation of the various algorithms and its usage.

- Mid-November to End-December (6 weeks) Development of the necessary scripts in order to obtain the data from the necessary experiments and gathering of said data.

- January to Mid-February (6 weeks) Analysis of the results. Depending on the conclusions obtained, it will be necessary to develop more scripts to perform additional experiments or create a new metric to measure evenness.

- Mid-February to end of March (6 weeks) Development of the Final Report.

\(^1\)More information available at: [http://moeaframework.org/](http://moeaframework.org/)
A.6 Interim Report
Previous and Current Project Title

Analysis of the Distribution of Pareto Optimal Solutions on various Multi-Objective Optimisation Genetic Algorithms

Progress made

In order to validate and evaluate the effectiveness of the current most used standard measures of diversity for a non-dominated set the configuration parameters of the evaluated algorithms were tuned against a series of standard test problems. The Spacing and Spread metrics were optimised while trying not to worsen the convergence metrics, as it is usually the case that as an algorithm gets closer to the Pareto Front, the diversity of the solutions worsens.

After running the tests we concluded that the standard metrics are sufficient in order to measure the spread on the tests problems, as a visualisation of the distribution of solutions gathered for the multiple problems showed that the evenness of solutions increased.

The Spread metric, originally defined to evaluate the spread of two-objective problems, is extended by multiple researchers to support many-objective problems. While tuning the algorithms, both versions of the metric were compared when running on two-objective problems. We concluded that in order to only use the Generalised Spread metric in our tests when using two-objective problems, we need to run the algorithms against problems with a Pareto Front "parallel" to the axis, otherwise errors can be introduced. In any other case, both metrics are quite similar, usually one scoring consistently higher than the other one.

When tuning the algorithms it was expected that the Spread metric will show better consistency than the Spacing metric due to its increased complexity. However, a case where these metrics diverged from one another could not be found, meaning the spacing metric, although being relatively simple, estimates the evenness of a distribution correctly.

The Generalised Spread metric was implemented into MOEAFramework, the software used to evaluate the algorithms. Additionally, while performing the tests a bug was found in the code and reported.
Remaining work

It is expected that each algorithm will tend to generate solutions over specified areas of the Pareto Front. In order to find this biases, problems with two-objectives will be used to plot a histogram of the non-dominated set produced by each algorithm in objective space.

In a second phase, more complicated problems with more than two objectives will be used to compare algorithms by using the Spacing and Generalised Spread metrics together with convergence metrics. We will try to find the type of problem on which each algorithm excels.
Bibliography


Bibliography


Bibliography


