MULTI- AND MANY-OBJECTIVE FACTORED EVOLUTIONARY ALGORITHMS

by

Amy Peerlinck

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

Doctor of Philosophy

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

May 2023

DEDICATION

This dissertation is dedicated to every little girl who was told she was not good enough and to my parents because they told me I was.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. John Sheppard, for his continued support and making this dissertation possible. I am grateful to have been pushed to become a better researcher, teacher, and person. Thanks to all of my committee members over the years for the help you have given: Dr. Stephyn Butcher, who provided me with code and assistance; Dr. Sean Yaw for his insightful comments and questions; Dr. Mary-Ann Cummings for her assistance in all aspects of life; and Dr. David Millman for his patience with my mathematical equations. I would also like to thank my labmates, with a special thanks to Jordan Schupbach, for his endless patience and vast statistical knowledge, Elliott Pryor, for his contribution to the variable grouping research, and Giorgio Morales Luna, for the creation of the Hyper3DNet. I extend my thanks to all the people I have had the pleasure of working with, notably, Dr. Shane Strasser, who laid the groundwork for my research in his own dissertation; and Drs. Bruce Maxwell and Paul Hegedus, for answering my questions on all aspects of agriculture. I could not have made it through this dissertation without my family and friends. Specifically, I want to thank my parents, Luc Peerlinck and Iris Scheirs, for their support of my journey despite its many twists and turns, including a move to a different continent. I am deeply grateful to Rani Van Cauwenbergh for her listening ear whenever I needed it, and to Matteo Björnsson for being my voice of reason. Lastly, I would not have made it without the love and support of my surrogate pets and their owners.

This work was funded in part by NSF grant 1658971 and USDA Grant NR213A750013G021.

TABLE OF CONTENTS

TABLE OF CONTENTS – CONTINUED

TABLE OF CONTENTS – CONTINUED

TABLE OF CONTENTS – CONTINUED

LIST OF TABLES

LIST OF TABLES – CONTINUED

Table Page

LIST OF TABLES – CONTINUED

Table Page

LIST OF TABLES – CONTINUED

Table Page

LIST OF FIGURES

LIST OF FIGURES – CONTINUED

Figure Page

LIST OF FIGURES – CONTINUED

Figure Page

LIST OF FIGURES – CONTINUED

Figure Page

LIST OF ALGORITHMS

ABSTRACT

Multi-Objective Optimization (MOO) is the problem of optimizing two or more competing objectives, where problems dealing with more than three competing objectives are termed as Many-Objective (MaOO). Such problems occur naturally in the real world. For example, many engineering design problems have to deal with competing objectives, such as cost versus quality in product design. How do we handle these competing objectives? To answer this question, population-based meta-heuristic algorithms that find a set of Pareto optimal solutions have become a popular approach. However, with the increase in complexity of problems, a single population approach may not be the most efficient to solve MOO problems. For this reason, co-operative co-evolutionary algorithms (CCEA) are used, which split the population into subpopulations optimizing over subsets of variables that can now be optimized simultaneously. Factored Evolutionary Algorithms (FEA) extends CCEA by including overlap in the subpopulations.

This dissertation extends FEA to MOO, thus creating the Multi-Objective FEA (MOFEA). We apply MOFEA to different problems in the MOO family with positive results; these problems include combinatorial and continuous benchmarks as well as problems in the real-world domain of Precision Agriculture. Furthermore, we investigate the influence of different grouping techniques on continuous large-scale, MOO, and MaOO problems to help guide research to use the appropriate techniques for specific problems. Based on these results, we find that some MaOO problems lead to large sets of non-dominated solutions. From this, an Objective Archive Management (OAM) strategy is presented that creates separate archives for each objective based on performance and diversity criteria. OAM successfully reduces large solution sets to a more manageable size to help end-users make more informed decisions.

The presented research makes four main contributions to the field of Computer Science: the creation of a new Multi-Objective framework to create and use subpopulation in a co-operative manner including the ability to use overlapping populations, the analysis of different grouping strategies and their influence on continuous optimization in both large-scale and multi-objective optimization, the introduction of a post-optimization solution set reduction approach, and the inclusion of an environmental objective into a real-world Precision Agriculture application.

CHAPTER ONE

INTRODUCTION

Many quantitative problems that occur in disciplines such as physics, engineering, and economics can be defined through mathematical principles; solving these problems is commonly referred to as "optimization" [161]. When problems have more than one goal to be solved, we say we are solving multiple "objectives". Multi-Objective Optimization (MOO) is the area of research that aims to find solutions for problems with multiple, competing objectives [33]. For example, an engineer designing a car needs to take different aspects into account such as comfort, speed, cost effectiveness, and safety; but increasing speed could come with a decrease in safety, thus creating a trade-off between the objectives. When we deal with problems that have more than three objectives, the problems are termed as Many-Objective Optimization (MaOO) problems [84]. MaOO is a subset of MOO, therefore when we refer to MOO, this includes MaOO problems.

Solving MOO problems means finding a set of potential solutions, since, due to their competing nature, the best solution for one objective will not necessarily be the best solution for the other objective(s). One class of commonly used algorithms to solve MOO problems are meta-heuristic algorithms, which keep a solution or set of solutions and adjust these solutions in an attempt to improve them [134]. More specifically, population-based algorithms have become the norm to solve MOO problems [33]. The use of a population of potential solutions offers a natural way to keep track of a set of solutions, where different parts of the population can explore different parts of the objective space, thus exploring the trade-off between objectives. However, there are many unanswered questions in the field of MOO. With the need to explore the objective space to find a good spread of solutions for all objectives,

comes the potential of slow convergence towards the set of optimal solutions. Balancing this trade-off is a prominent problem in MOO, especially as the number of variables and objectives increases.

To deal with a large number of decision variables, known as Large-Scale Optimization (LSO), Co-operative Co-Evolutionary Algorithms (CCEAs) have been proposed [22, 97, 117]. CCEAs divide the variables into variable groups and optimize the variable groups separately before combining them again to form a complete solution. The chosen variable grouping strategy can have a strong influence on the optimization process of CCEAs, and has become its own research area [85, 140]. In this dissertation we aim to explore different aspects of CCEAs in the context of Large-Scale MOO and MaOO through empirical analysis.

1.1 Motivation

Industrial problems, such as engineering design, economics, and electricity distribution, have been at the forefront of multi-objective optimization (MOO) research since it manifested itself as a research field of interest in the 80s [32]. However, since then, many more real world applications have entered the realm of multi-objective optimization. This includes medical decision making, land use planning, and supply chain management [136]. Furthermore, with the increase in computational power and the prevalence of computers in every day life, the dimensionality of the problems as well as the amount of available data keeps growing. This increase in data can mean an increase in decision variables to be solved, resulting in large scale optimization (LSO) problems. Population-based algorithms have become the primary way MOO and LSO problems are solved [32, 170]. Current research has been focusing on developing new multi-objective evolutionary algorithms (MOEAs) and applying them to specific problems, such as autonomous vehicle navigation and city planning [24, 27].

When looking at LSO from the perspective of single-objective optimization, a common approach, when using population-based algorithms, is to decompose the problem into smaller

groups that are then optimized separately and combined to find the final solution [170]. Such approaches are widely known as co-operative co-evolutionary algorithms (CCEA) [117]. Research has indicated that the decomposition strategy is vital in the performance of the algorithm for continuous optimization. A popular approach for problem decomposition is to apply random grouping, which dynamically assigns variables to a random group of fixed size at each iteration [169]. This results in groups of equal size and an equal division of variables into the subpopulations, which may not be the most effective representation for a given problem. A popular alternative to random grouping that alleviates the equal splitting of variables is known as variable interaction learning. In this case, variables are determined to be interacting by calculating interaction effects (e.g., by using derivatives) and variables that are found to be interacting are then grouped together. Based on these interactions, problems can be categorized depending on the level of separability of the interaction. If none of the variables interact, a problem is fully separable. Conversely, if interaction is found between all variable pairs, this is a fully non-separable problem. Lastly, if some variables are found to be interacting, but others are not, this is known as partial separability.

One of the most popular approaches to perform variable decomposition is known as Differential Grouping (DG), which uses the derivative of a function to determine interaction between variables [107]. This results in disjoint subpopulations, which indicates DG may not be capable of accurately representing problems with indirect variable interactions , i.e., variables that are not interacting directly, but that have a third, shared variable they each interact with [140]. If a variable is only optimized in relation to the variables it directly interacts with, the influence of indirect interactions may not be taken into account. This is not a problem if the representation of the problem is an accurate depiction of the corresponding variable interaction; however, if the grouping is not accurate, this could result in a lack of exploration of the solution space [132]. To this end, the Factored Evolutionary Algorithm (FEA) was introduced, which includes overlap in the subpopulations, meaning a

single variable can belong to more than one group [138].

DG inspired many extensions [91, 109, 141]; however, limited research has been performed on which grouping approach is useful in which scenarios. Even though there has been research looking at the effectiveness of grouping strategies on different functions, it is as of yet unclear how important the accuracy of variable interaction representation actually is. For example, if an overlapping architecture ensures that all variables are connected, is this sufficient to explore the feasible solution space? And what type of functions benefit from which decomposition approach? Even though the variable decomposition approach is widespread in LSO, comparatively, it has seen limited research in the field of MOO [145]. How does variable grouping affect large-scale multi-objective optimization? Could adding overlap to subcomponents improve MOO results? When is a decomposition approach the right choice?

There has been a lot of work done to apply evolutionary algorithms to MOO and LSO problems. There have been several surveys on different aspects of these issues [97, 114, 145]. While some of them focus on specific subproblems in the field of MOO, others try to create a more comprehensive overview. However, these surveys generally agree that as the number of objectives increases (resulting in Many-Objective Optimization), appropriate exploration of the objective space becomes more difficult. If we consider the added difficulty of having multiple objectives on top of dealing with a large decision space, applying a decomposition strategy has two important benefits: the ability to apply distributed computing and an improved exploration of the subspaces [132]. Furthermore, the objective space suffers from the curse of dimensionality: as the number of objectives increases, the number of non-dominated solutions increases exponentially [70]. This can result in solution sets with thousands of solutions, which are nigh impossible for a human end-user to process. To help the end-user make an informed decision, there are two common approaches: using reference directions to guide the algorithm's search early on [34] or reduce the solution-

set post-optimization [142]. However, both of these approaches require knowledge of the problem, either to define the appropriate reference directions or to choose the appropriate reduction algorithm to apply. Therefore, we propose a new post-optimization solution set reduction technique that does not require such knowledge.

Lastly, we apply these concepts to fertilizer prescription optimization in the field of Precision Agriculture (PA) and evaluate the effectiveness of the different algorithms in this real-world setting. PA is a subfield of agricultural research that uses different technologies to help improve crop and livestock management while increasing sustainability practices. More specifically, we investigate site-specific farming using variable rate application technology. This means that the field is divided into smaller plots and different rates of inputs (e.g., fertilizer, herbicide, seeding) are prescribed and applied to these smaller plots to optimize yield. There are two different types of prescriptions: experimental and optimal prescriptions. Experimental prescriptions are intended to gather data on a specific field, and this data can then be used to create optimized fertilizer prescriptions. PA research has found that site-specific optimization is beneficial for farming profit [79], but yield is not the only variable to be taken into account when creating these prescriptions. Furthermore, agriculture is the primary source of nitrogen pollution, due to the use of nitrogen as crop fertilizer and the presence of nitrogen in animal manure [30]. We aim to address this environmental issue by including fertilizer minimization as an objective when creating fertilizer prescription maps. Additionally, we also wish to minimize rate jumps between consecutive cells to reduce wear and tear on machinery, effectively reducing waste. All of these different problems and objectives makes it a prime candidate for applying a MOO approach, and allows us to show that environmental concerns can be addressed using MOO.

1.2 Research Questions

We recap the core research questions of the thesis; chapters will provide more details on the research performed.

1. Many-objective optimization
    - How does the use of overlapping subpopulations in co-operative co-evolutionary methods affect Many-Objective Optimization?
    - How can we reduce large non-dominated solution sets to a more manageable size for the end-user?
2. Influence of different grouping methods on optimization
    - How do different grouping strategies influence optimization?
    - In which situations is a variable-decomposition approach the appropriate choice (e.g., number of objectives, size of the problem, nature of the problem)?
3. Real world applications
    - How do we apply different MOO techniques to the real-world problem of creating fertilizer prescription maps?
    - How does adding in environmental concerns, such as nitrogen seeping into the soil and thus the waterways, affect the optimization of the fertilizer prescription maps?

1.3 Contributions

The contributions of our work to the field of evolutionary computation and multi-objective optimization are as follows:

- We introduce the Multi-Objective Factored Evolutionary Algorithm (MOFEA). MOFEA is a co-operative co-evolutionary approach to multi-objective optimization that not only allows for distinct subpopulations, but includes the use of overlapping subpopulations. When defining MOFEA, we enable the use of co-operative co-evolutionary MOEAs as a part of the general framework. In other words, using distinct subpopulations does not impact MOFEA. Furthermore, we show how the proposed framework is applicable to any MOEA by applying it to three popular MOEAs.

- We apply MOFEA to the classic multi-objective knapsack (MOKS) problem and find that using overlapping subpopulations improves results as compared to NSGA2 and CC-NSGA2. We also propose a different MOKS problem that maximizes profit while minimizing weight, volume, and the difference in weight/volume of the items in the knapsack. This problem was found to be more difficult to solve than the classic problem. We apply the same algorithms and once again find that overlap is beneficial for the optimization process. These experiments show the benefit of using overlapping subpopulations in MOO through empirical analysis.

- We study different factor architectures for large-scale continuous optimization problems. More specifically, we extend DG to create overlapping groups and we create a random grouping approach that uses a tree-graph to create connected groups through overlap. We confirm the benefit of using overlapping subpopulations with single-objective optimization, and we also find that identifying variable interaction may not be necessary when creating overlapping groups, as long as the overlap connects all the variables.

- Based on the results for LSO and the benefit found by applying MOFEA to the MO-KS, we examine variable grouping for continuous LSMOO to gain deeper insight into the effects of variable grouping. In our empirical analysis, we find that overlap

often improves results compared to the single population MOEAs and CCMOEAs. Furthermore, we notice certain function-specific trends that can help future research identify the right approach to use for problems with specific characteristics.

- We also propose using MOFEA to solve the following problem posed by Li *et al.* [85]: they found that when applying DG along different objectives, different groups were created for each objective, thus creating groups that contain the same variables. The authors note that no appropriate solution to this problem had been proposed. The capability of using overlapping subpopulations in MOFEA directly addresses this issue.

- During our variable grouping experiments for LSMOO, we found that the non-dominated solution set can contain thousands of solutions as the number of objectives increases to five and ten. Because of this, we propose a novel approach to help reduce the solution set size. The Objective Archive Management (OAM) strategy provides a multi-faceted approach to solution set reduction.

- Finally, we show the benefit of the MOFEA framework in a real-world application by including environmental objectives when creating fertilizer prescription maps. We use MOO to help mitigate negative environmental impacts of fertilizer application in agriculture. This furthers the field of multi-objective optimization by showing, through empirical analysis, that MOO approaches have real-world benefit.

1.4 Organization

The remainder of this dissertation is organized as follows. We start by giving fundamental background information on the fields of multi- and many-objective optimization, multi-objective combinatorial optimization, and co-operative co-evolutionary algorithms in Chapter 2. Chapter 3 introduces the multi-objective factored evolutionary algorithm

(MOFEA), a framework for co-operative co-evolution in multi-objective optimization that allows for overlapping subpopulations. Chapter 4 explores the effects of variable grouping in large-scale optimization, while Chapter 5 looks at variable decomposition for multi- and many-objective optimization problems. Chapter 6 addresses the issue of large non-dominated solution sets in many-objective optimization through the Objective Archive Management strategy. Lastly, we apply MOFEA to generate fertilizer prescription maps in Precision Agriculture (Chapter 7) before summarizing our main contributions and proposing future work in Chapter 8.

CHAPTER TWO

BACKGROUND

This chapter lays out the foundational work and concepts that are used in this dissertation. We start by exploring basic definitions used in large-scale and multi-objective optimization as well as popular approaches and common evaluation metrics. We then move on to many-objective optimization and multi-objective combinatorial optimization. Lastly, we provide background information on the co-operative co-evolutionary and factored evolutionary algorithms.

2.1 Population-Based Algorithms

Before diving into multi-objective optimization (MOO) and its specific algorithms, we describe three commonly used population-based algorithms: Genetic Algorithm [62], Differential Evolution (DE) [137], and Particle Swarm Optimization [41]. Several years after the introduction of DE in 1997, there was a boom in the creation of novel nature-inspired meta-heuristic algorithms [103, 168, 180]. However, most research still uses classic approaches to solve different problems, where the above three algorithms are currently the most commonly used population-based approaches (based on their impact score) [45].

2.1.1 Genetic Algorithm

The Genetic Algorithm (GA) is motivated by the concept of "survival of the fittest" [62]. The GA is a form of stochastic search, in that it uses randomness to explore the search space. A population of candidate solutions is initialized and subsequently modified using selection, crossover, and mutation operators to create offspring, which can then replace all or part of the population. This general flow can be seen in Figure 2.1.

The population consists of a set of chromosomes, each representing a possible solution, and a chromosome consists of genes, which represent variables or sets of variables, where the actual values of the variables are known as alleles. The simplest form of representation for genes in chromosomes is a bit string, but a gene can also be represented by integers, real numbers, permutations, trees, etc. A chromosome, or solution, is evaluated by calculating its fitness. Once the fitness for each solution has been calculated, the selection of parents for offspring production is performed. There are different parent selection procedures available, but generally, fitter individuals are preferred to be parents. The selected parents can then be recombined via crossover to create two new candidate solutions. A simple implementation of crossover, known as one-point crossover, picks a random point in the chromosome and combines the first part of the first parent with the second part of the second part and vice versa. The second operator, mutation, only adjusts a small part of the chromosome to help introduce diversity into the population. It randomly decides which genes of a chromosome to change; the exact implementation of different mutation operators depends on the variable representation. Both of these genetic operators are performed based on an operator rate, determined by the user as a parameter.

2.1.2 Differential Evolution

DE is a simple parallel direct search method that also uses mutation and crossover operations but in a different way than the GA [137]. A population of vectors is initialized randomly across the entire decision space using a uniform probability distribution. For each vector in the population, where the chosen vector is called the target vector X_t, the following process takes place. DE performs mutation by adding the weighted difference between two randomly chosen vectors, X_a and X_b to a third randomly chosen vector X_c ($X_a \neq X_b \neq X_c$):

$$X_m = X_a + F \cdot (X_b - X_c), \tag{2.1}$$

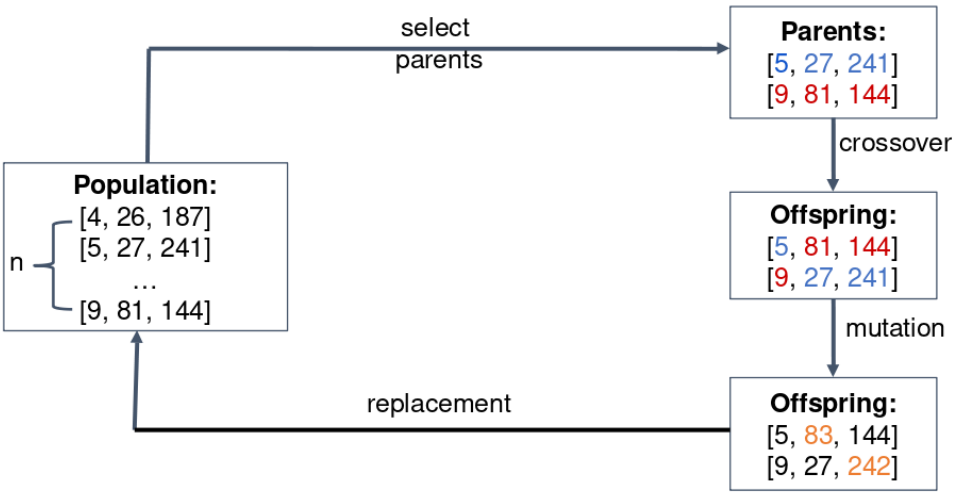


Figure 2.1: High-level flowchart of the genetic algorithm using one-point crossover and randomized mutation.

where F is a user chosen scale factor. The resulting mutated vector X_m is then combined with the target vector X_t through crossover to create the trial vector X_r. DE sometimes refers to this procedure as parameter mixing. Crossover chooses certain values of each vector to switch, or cross over, between the two vectors, thus resulting in two new vectors. However, in DE only the altered trial vector is kept as offspring. After mutation and crossover are performed, selection decides whether the new trial vector is to be kept. Its fitness is compared to the target vector's to determine whether or not it should replace the target vector. This entire process can be seen in Figure 2.2.

2.1.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is based on the social behavior of flocking birds [41]. Here, solutions are called particles, which belong to a swarm and whose state is updated using velocity vectors. A particle is updated using the following general equation:

$$x_i(t+1) = x_i(t) + v_i(t+1),$$

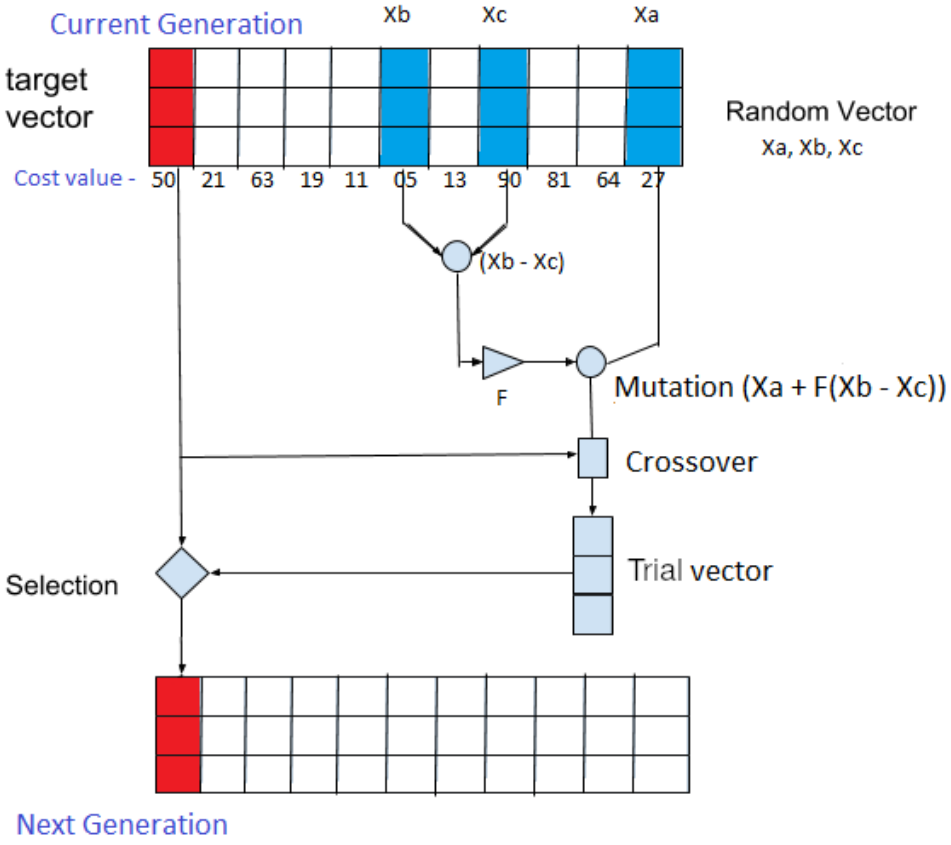


Figure 2.2: Differential Evolution [112].

where $v_i(t+1)$ refers to the velocity at timestep $t+1$ and $x_i(t)$ is the particle's current position. This velocity vector is dependent on the fitness of previous states of the particles, effectively guiding them to the "fittest solution." The global best velocity implementation of PSO is calculated as follows:

$$v_i(t+1) = \omega v_i(t) + c_1 r_1(x_{pbest} - x_i(t)) + c_2 r_2(x_{gbest} - x_i(t)), \tag{2.2}$$

where ω is the inertia weight, which was introduced to adjust how much influence the previous velocity has on the velocity in the next time step. Parameters c_1 and c_2 are the cognitive and social acceleration coefficients respectively, and r_1 and r_2 are randomly generated values

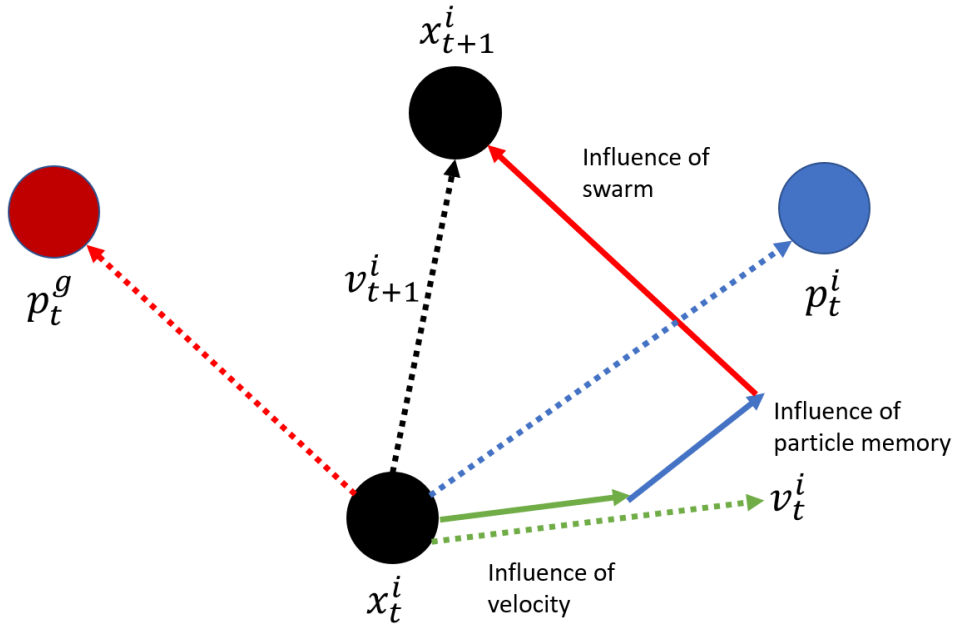


Figure 2.3: Particle x^i velocity (v^i) update process using local (p^i) and global (p^g) best at iteration t to find the particle position at iteration $t+1$ [155].

between 0 and 1. These values determine the influence of the local (x_{pbest}) and global (x_{gbest}) best positions on the current particle's velocity. The initial particle population, or swarm, is initialized randomly across the search space. The position of each particle is then updated based on the previously mentioned velocity vector (Figure 2.3).

To control stability and deter a particle from rapid acceleration, velocity clamping or constriction is implemented. Velocity clamping can be used to limit the potential velocity of a particle by assigning a minimum and maximum value. The previously mentioned inertia weight also adjusts the particle velocity. Lastly, constriction uses a coefficient to adjust the entire velocity equation. These strategies limit the scope of the search and balance exploration and exploitation in the search process.

2.2 Multi-Objective Optimization

Multi-Objective Optimization is the process of optimizing multiple objectives simultaneously [70]. Formally, without loss of generality, assume we wish to minimize M objectives. Then MOO consists of solving

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) = \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})\}$$

with $M \geq 2$ objective functions $f_i : \mathbb{R}^n \to \mathbb{R}$ that have conflicting goals, $f_i \in F^M$ where F^M represents the objective space, and $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$ denotes the decision variables, where $\mathcal{X} \in \mathbb{R}^n$ is the solution space.

Like any optimization problem, MOO problems can be subject to constraints. The general constrained MOO can then be defined as follows:

$$\begin{aligned}
&\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) = \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})\} \\
&\text{s.t. } g_j(\mathbf{x}) \leq 0, j = 1, \ldots, J, \\
&\quad h_m(\mathbf{x}) = 0, m = 1, \ldots, M, \\
&\quad \mathbf{x}_i^L \leq \mathbf{x}_i \leq \mathbf{x}_i^U, i = 1, \ldots, N.
\end{aligned}$$

Many MOEA's are able to handle constraints without needing to be adjusted for said constraints [6, 175]. On top of this general capability, several frameworks have been proposed to handle constrained MOO with any MOEA as the base algorithm [73, 119].

2.2.1 Classic Approaches

A simple approach to solving MOO problems is to transform the multiple objectives into a single objective [33]. As a result, single-objective methods can be applied directly. Two common approaches are weighted sum and the ϵ-constraint method, where the former

transforms the problem through aggregation and the latter transforms all objectives except one into constraints. The weighted sum approach assigns weights to each objective function, where the weights sum to 1. For example, if there are three objective functions ($M = 3$) $f_1(x_1, x_2)$, $f_2(x_1, x_2)$, and $f_3(x_1, x_2)$, these can be combined into a single objective as follows,

$$f(x_1, x_2) = \alpha_1 f_1(x_1, x_2) + \alpha_2 f_2(x_1, x_2) + \alpha_3 f_3(x_1, x_2),$$

where, $\sum_{i=1}^{3} \alpha_i = 1$. The simplicity of the weighted sum method makes it a popular choice; however, it may not be the best approach for more complex methods. The ϵ-constraint method is more robust. The general strategy is to take one objective while adding limits, defined by ϵ, to the other objectives. The downside of the ϵ-constraint method (as well as the weighted sum) is that there is no guarantee to find a uniformly distributed set of solutions across the different objectives. It is also difficult to determine the optimal ϵ values; this usually requires prior knowledge or extensive tuning. Plus, in order to obtain a set of solutions using the weighted sum or ϵ-constraint methods, the algorithms need to be run several times with different settings.

2.2.2 Pareto Optimization

When exploring algorithms that are generalized for any MOO problem, a trend has emerged that favors Pareto-based solutions [145, 153, 171]. This means tracking the Pareto front of possible solutions along the different dimensions in the solution space [135]. Pareto dominance and Pareto optimality are then used to determine solution quality. Some of the core concepts dealing with Pareto fronts are defined as follows.

Definition 1 (Pareto-Dominance) *Assuming minimization, a point* $\mathbf{x}^* \in \mathcal{X}$ *dominates another point* $\mathbf{x} \in \mathcal{X}$ *(*$\mathbf{x}^* \succ \mathbf{x}$*) if* $\forall f_i \in F^M, \forall \mathbf{x} \in \mathcal{X}, \mathbf{x} \neq \mathbf{x}^*, f_i(\mathbf{x}^*) \leq f_i(\mathbf{x})$*, and* $\exists f_j \in F^M, f_j(\mathbf{x}^*) < f_j(\mathbf{x})$.

It then follows that $\mathbf{x}^*$ is said to be non-dominated or Pareto optimal if no other feasible solution dominates it. The implication of this definition is that, when improving an objective component of an "optimal" solution, at least one other component will be degraded.

Definition 2 (Pareto Optimal Set) *The Pareto optimal set (PS^*) is the set of non-dominated solutions with respect to the solution space $\mathcal{X}$.*

Definition 3 (Pareto Optimal Front) *The Pareto optimal front (PF^*) is the set of points mapped from the Pareto optimal set onto the objective space F^M to form the boundary of the set of non-dominated solutions.*

MOO-focused algorithms that use Pareto techniques search for the Pareto optimal front; the resulting Pareto front is called the approximate Pareto front.

2.2.3 Evaluation Metrics

How do we measure the quality of the solution set found by different MOO algorithms? In general, MOO metrics can be categorized in two ways: 1) by looking at which aspects of a solution set the metric is addressing, or 2) by how many solution sets a metric evaluates, i.e., does it look at a single solution set (unary) or does it compare two solution sets (binary) [122]. In the first category, there are three aspects of a solution set that can be evaluated:

- Cardinality: Number of proposed solutions.
- Accuracy: How close the approximation solution set is to the actual Pareto optimal front.
- Diversity: Distribution (relative distance) and spread (range of values) of the solution set.

A survey by Riquelme *et al.* found the following six metrics to be the most commonly used across MOO articles published from 2005-2013 [122]. We list the six metrics with

information on the two categories of evaluation: first, whether it is a unary or binary metric, and second, which aspects of a solution the metric investigates.

1. Hypervolume (HV): Unary, cardinality, accuracy, diversity
2. Generational Distance (GD): Unary, accuracy
3. Inverse Generational Distance (IGD): Unary, accuracy, diversity
4. Δ-indicator: Unary, diversity
5. Epsilon family (ϵ): Binary, cardinality, accuracy, diversity
6. Coverage (C): Binary, cardinality, accuracy, and diversity

By far the most popular metric is the HV indicator. Its popularity is likely because it is the only unary metric that explores all three aspects of the given solution set. Given M objectives, a set of points $\mathbf{X} \in \mathbb{R}^M$, representing the approximate Pareto front, and a reference point $\mathbf{r} \in \mathbb{R}^M$, the HV of $\mathbf{X}$ is the measure of the region dominated by $\mathbf{X}$ and bound by $\mathbf{r}$ [11]. This is in contrast to measures such as Generational Distance or Inverse Generational Distance, which require the true Pareto Front to be known. Since we do not necessarily know the true Pareto Front when dealing with real-world problems, the HV is a natural choice to gain insight in the size of the covered objective space [183]. The biggest downside to HV is its computational cost; to address this issue a recursive strategy to calculate HV was introduced, known as the Walking Fish Group (WFG) HV [157]. For k points in the Pareto front $\mathbf{X}$, let

$$HV(\mathbf{X}) = \sum_{i=1}^{k} ExcHyp(p_i, \{p_{i+1}, \ldots, p_k\})$$

where $ExcHyp(p\mathbf{S})$ is called the "exclusive hypervolume" and is defined as

$$ExcHyp(p, \mathbf{S}) = HV(\mathbf{S} \cup \{p\}) - HV(\mathbf{S}).$$

GD and IGD are unary in the sense that they only look at one approximation set; however, both metrics require knowledge of the Pareto optimal front, or a reference set as a substitute [68]. GD considers the average distance between the members of the approximation set $\mathbf{X} = \{x_1, x_2, \ldots, x_{|\mathbf{X}|}\}$ and their closest solutions in the optimal front or reference front $\mathbf{R} = \{r_1, r_2, \ldots, r_{|\mathbf{R}|}\}$:

$$GD(\mathbf{X}) = \frac{1}{|\mathbf{X}|} \left(\sum_{i=1}^{|\mathbf{X}|} d_i^p \right)^{1/p},$$

where d_i is the distance from x_i to the nearest reference point in $\mathbf{R}$, and p is an integer parameter, often set to $p = 1$. If $p = 2$, the Euclidean distance is used as the distance metric. IGD looks at the minimum distance from the approximate set to all solutions in the optimal front:

$$IGD(\mathbf{X}) = \frac{1}{|\mathbf{R}|} \left(\sum_{j=1}^{|\mathbf{R}|} \hat{d}_j^p \right)^{1/p},$$

where $\hat{d}_j$ is the distance from r_j to the nearest objective vector in $\mathbf{X}$.

The last of the unary metrics is the spread indicator S, which is defined as follows [69]:

$$S = \sum_{i=1}^{M} \left[\max_{\mathbf{x} \in \mathbf{X}} \{f_i(\mathbf{x})\} - \min_{\mathbf{x} \in \mathbf{X}} \{f_i(\mathbf{x})\} \right].$$

Thus S corresponds to the sum of the width for each objective, indicating how wide the solutions are spread across the objective space, i.e., a measure of distance instead of volume.

There are two metrics that compare two Pareto front approximations directly: the ϵ

and the coverage metric. Calculating ϵ gives a factor by which one approximation set $\mathbf{X}'$ is worse than another $\mathbf{X}''$ considering all objectives. If trying to minimize a problem with k objectives, an objective vector $\mathbf{x}' \in \mathbf{X}'$ ϵ-dominates another objective vector $\mathbf{x}'' \in \mathbf{X}''$ if and only if

$$\forall i \leq M : x'_i \leq \epsilon \cdot x''_i,$$

where $\epsilon > 0$ [182]. By this definition, $I_\epsilon(\ \mathbf{X}',\ \mathbf{X}'')$ determines the minimum value of ϵ by which each objective vector in $\mathbf{X}''$ is ϵ-dominated by at least one objective vector in $\mathbf{X}'$.

The coverage C of two fronts, denoted as $\mathbf{X}'$ and $\mathbf{X}''$, can be calculated as follows [184]:

$$C(\mathbf{X}', \mathbf{X}'') = \frac{|\{\mathbf{x}' \in \mathbf{X}' : \exists \mathbf{x}'' \in \mathbf{X}'' : \mathbf{x}'' \succeq \mathbf{x}'\}|}{|\mathbf{X}'|}.$$

This returns a value between 0 and 1, where 0 indicates that no solutions in $\mathbf{X}'$ are dominated by or equal to any solutions in $\mathbf{X}''$, and 1 indicating that all solutions in $\mathbf{X}'$ are dominated by $\mathbf{X}''$. Since the reverse is not a symmetric measure, the metric is calculated for both combinations: $C(\mathbf{X}', \mathbf{X}'')$ and $C(\mathbf{X}'', \mathbf{X}')$.

2.2.4 Pareto-Based Approaches

The idea of using Pareto-optimal solutions to find a non-dominated solution set has been growing in popularity, and several different algorithms have been proposed. Generally viewed as the first MOEA, the Vector Evaluated Genetic Algorithm (VEGA) was introduced by Schaffer in 1984 [128]. It evaluates solutions for a single objective at a time, which leads to locally non-dominated solution for each objective. However, evaluating solutions in this way ignores non-dominated solutions that are not optimal for any of the objectives [1]. Since then, approaches that better explore the objective space have been proposed. In general, four different types of algorithms are distinguished: Pareto-based sorting, indicator based sorting, decomposition-based approaches, and reference direction-based exploration. Each

of these approaches have at least one popular algorithm that has been shown to work well on different types of problems.

2.2.4.1 Pareto-Based Sorting Pareto-based sorting algorithms order individuals based on Pareto dominance; this ordering is algorithm dependent. There are two popular algorithms that perform such an ordering: Strength Pareto Evolutionary Algorithm (SPEA) [183] and Non-Dominated Sorting Genetic Algorithm (NSGA) [135]. Originally, SPEA used a population P_t and an archive of non-dominated solutions P'_t, where t indicates the generation; each member i in the population is assigned a strength value S_i based on the number of population members that are dominated by or equal to the individual, divided by the population size plus one [187]. This strength value also serves as the fitness value for archive members. Each population member is then assigned a fitness F_i by summing over the strength values of the archive members that dominate the individual, plus one. However, this means that individuals in the population that are dominated by the same archive members, have the exact same fitness. To address this issue, SPEA was adjusted in 2001 to create SPEA2 [185].

In SPEA2, the strength value is adjusted to include both the archive and the population and is based on how many solutions an individual dominates across these two solutions sets:

$$S_i = \frac{|\{j | j \in P_t \cup P'_t \wedge i \succ j\}|}{|P_t| + 1}.$$

The "raw" fitness value R_i sums over its dominators' strength values in the archive and general population:

$$R_i = 1 + \sum_{j \in P_t \cup P'_t, j \succ i} S_j.$$

In addition to this adjusted strength value, density information is included based on the k-nearest neighbors to distinguish between individuals with the same raw fitness value. The

k-th element's distance is represented by σ_i^k, and $k = \sqrt{|P_t| + |P'_t|}$. An individual's density D_i is then calculated as follows:

$$D_i = \frac{1}{\sigma_i^k + 2}.$$

The final fitness value is then calculated as $F_i = R_i + D_i$.

Furthermore, the non-dominated archive now has a fixed size. This means three different scenarios can occur based on the number of non-dominated solutions. Either the number of non-dominated solutions is exactly the size of the archive, the number is smaller, or the number is larger. If it is smaller, the archive is filled out with dominated solutions based on their fitness value. If the archive needs to be truncated this is done through an iterative process based on the distance between individuals, removing the individuals with the smallest distances.

The concept of Non-Dominated Sorting was introduced by Srinivas *et al.* in 1994 [135], and improved in 2002 by Deb *et al.* to create the Non-Dominated Sorting Genetic Algorithm II (NSGA2) [36]. NSGA2 is an elitist GA that uses a crowding distance measure to maintain diversity in the next generation. Figure 2.4 shows how the population is sorted to perform selection for the next generation. The parent population P_t and the offspring population Q_t are combined into one population $R_t = P_t \cup Q_t$. R_t is sorted based on the non-domination principle, and individuals are assigned to different solution sets F_j based on how good the solution is. If an entire set of solutions is larger than the remaining slots for the next population, a second elimination is performed for that set based on crowding distance. In Figure 2.4, this is shown for the third solution set F_3. The crowding distance is calculated for each individual in F_3, and the solutions with the largest crowding distance are chosen for the next generation. The crowding distance measure for an individual CD_i is based on the cardinality of the solution set and its distance to the solution boundary, where a boundary

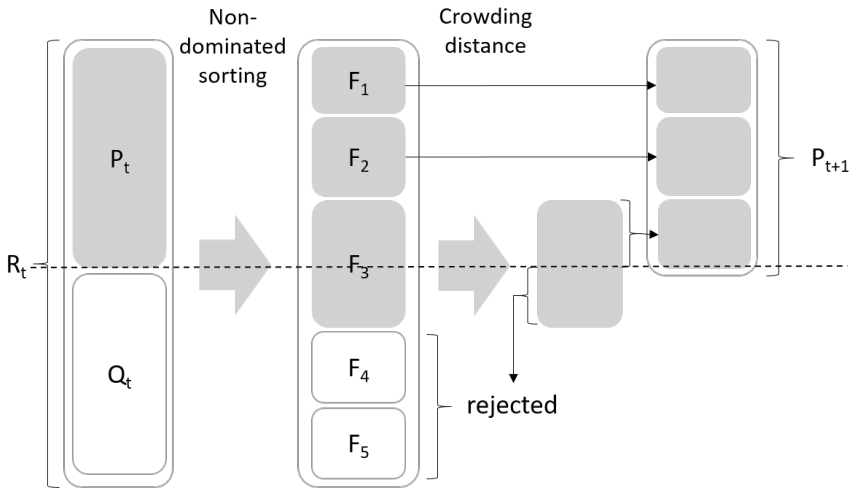


Figure 2.4: NSGA2 selection procedure [36].

solution is defined as having the highest ($F_j^{\max}$) and lowest ($F_j^{\min}$) objective values.

$$CD_i = \sum_{n=1, n\neq i}^{|F_j|} \frac{||F_j^i - F_j^n||}{F_j^{\max} - F_j^{\min}}$$

Boundary solutions' crowding distance is set to infinity so they will always be selected.

Lastly, Particle Swarm Optimization (PSO) was adjusted to the multi-objective case to create multi-objective PSO (MOPSO) [23]. The algorithm starts by initializing a population in the same way regular PSO does but evaluates each particle based on the multiple objectives. The non-dominated solutions from the initial population are put into an archive. After the position evaluation, hypercubes representing the explored search space are created. The particles are then located within these hypercubes to define particle coordinates according to the values of the objective functions. Then, the algorithm iterates through velocity updates, and positional adjustments. A particle's velocity is updated using Equation 2.2, except that the global best is replaced by a particle from the non-dominated repository. The repository particle is chosen in the following way. Each hypercube is evaluated by counting the number of particles it contains. Hypercubes with more than one particle get a

fitness value by dividing a chosen value z by the number of particles the hypercube contains. Roulette-wheel selection based on these fitness values selects the hypercube from which a random particle is selected. These particle update steps are repeated until a stopping criterion is met.

2.2.4.2 Indicator Based Search In this type of algorithm, an MOO evaluation measure is used to sort individuals. The hypervolume-based algorithm Hypervolume Estimation Algorithm for Multi-objective Optimization (HypE) is the most broadly used example of this type of algorithm [4]. The idea is to use the HV indicator to guide the search through selection of the population (mating selection) and of the archive (environmental selection) by assigning hypervolume-based fitness values.

HypE slices the solution space recursively into hyperrectangles to determine which solutions are non-dominated. The number of recursions is decided by the number of objectives. For each objective, the solutions are ordered by their fitness value for the objective at hand and the space is sliced according to the objective axis. Within this hyperrectangle subspace, solutions are filtered out based on their dominance levels, and slicing is performed along the next objective axis, until each objective has been processed. Once recursion is finished, the HV is calculated for the filtered solutions. However, the HV calculations themselves are still expensive, so the authors propose a Monte Carlo simulation to approximate the HV value for problems with more than three objectives. This is done by defining a sampling space from which a set number of objective vectors are sampled uniformly at random. For each partition, the number of samples that lie within it are counted and this number is multiplied by the volume of a sampling box. The authors define the sampling box as the axis-aligned minimum bounding box containing the previously defined subspaces. The resulting HV-based fitness values are then used to select parents for mating as well as to select the next population of individuals.

2.2.4.3 Decomposition-based Approaches Zhang proposed the first multi-objective evolutionary algorithm based on decomposition (MOEA/D). MOEA/D uses a set of weight vectors to guide the search by decomposing the MOO problem into several scalar optimization subproblems [175]. The algorithm was created such that any problem decomposition approach can be applied, e.g., weighted sum (Section 2.2.1), Chebyshev, or Boundary Intersection (BI). Chebyshev scalarization uses the following approach:

$$\min g^{ch}(x|\lambda, z^*) = \max_{1 \leq i \leq M} \{\lambda_i | f_i(x) - z_i^*|\}, \tag{2.3}$$

where λ is a weight vector, x are the variables to be optimized, and z^* is a reference point [175]. Each optimal solution for Equation 2.3 is a Pareto optimal solution for the original objective functions. The weight vector λ can then be optimized in lieu of the variables xx, since there exists a weight vector λ that optimizes Equation 2.3 for each Pareto optimal point x^*. MOEA/D initializes N uniformly spread weight vectors $\lambda^1 \ldots \lambda^N$. Each weight vector λ^i has a neighborhood of its closest weight vectors, measured using Euclidean distance. Chebeshev scalarization does not create a smooth aggregation function for continuous MOO problems. To alleviate this issue, several different BI approaches have been proposed. BI approaches try to find intersection points of the top-most boundary and a set of lines; i.e., if the set of lines is spread evenly, the intersection points should provide a good approximation to the total Pareto front. BI uses an equality constraint $z^* - f(\mathbf{x}) = d\lambda$ which ensures that the objectives $f(\mathbf{x})$ fall within the attainable objective set while still pushing $f(\mathbf{x})$ to the boundary of said objective set (Figure 2.5). Zhang *et al.* added a penalty factor θ to handle this equality constraint, thus creating Penalty-based Boundary Intersection (PBI).

Each generation of MOEA/D performs the following steps. For each weight vector λ^i (representing a scalarized subproblem), two parents are chosen randomly from its neighborhood to mate, after which mutation is performed. Then, the resulting solution

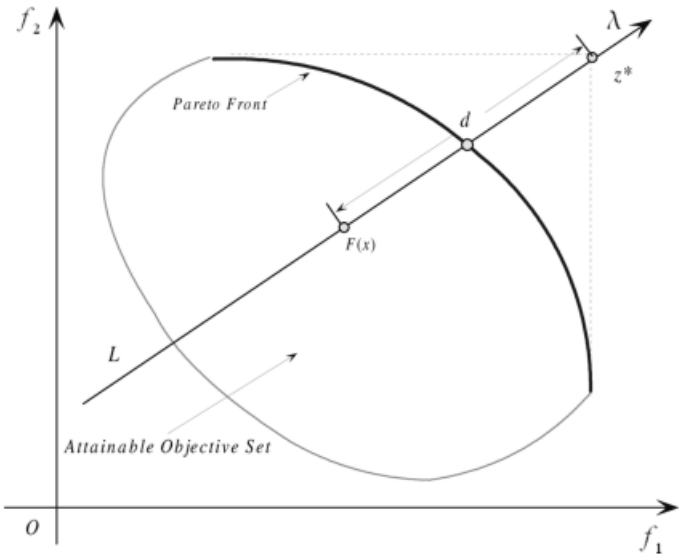


Figure 2.5: Boundary intersection approach [175].

is repaired if necessary based on problem-specific heuristics. The fitness values of the resulting offspring are compared to the currently saved fitness values, and if the fitness for any objective is better, the saved fitness value is replaced. The variable values (corresponding to the replaced fitness values) are then used to update the neighboring solutions. The found solutions for each subproblem are added to the non-dominated archive if it is not dominated by any other members of the archive.

2.2.4.4 Reference Direction Based Approaches Similar to MOEA/D, NSGA3 is an extension of NSGA2 that uses reference directions to guide the search through the objective space. However, weight vectors are now used to guide the search as opposed to transforming the objectives. This approach was created to better handle many-objective optimization problems [34] (see Section 2.3). In NSGA3, predefined reference points are specified, i.e., weight vectors, where points corresponding to these reference points become the focus of the algorithm. More specifically, NSGA3 replaces the crowding distance calculation of NSGA2 with these reference points. The idea of non-domination levels is still applied as

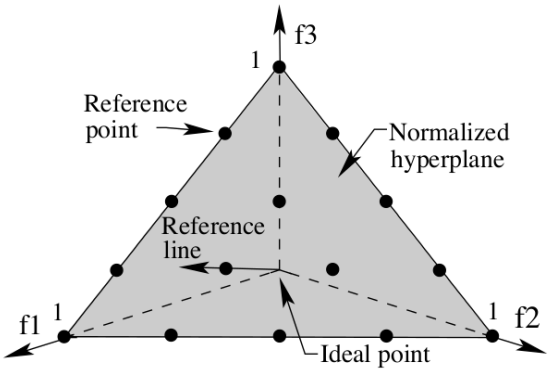


Figure 2.6: Example of a normalized reference plane for a three-objective function with four divisions ($p = 4$) using Das-Dennis [29] on each axis and the resulting 15 reference points [34].

the first step; however, the selection of points from the lowest included non-domination level is now based on reference points, which are placed on a normalized hyperplane. The most common approach to create these reference directions is called the Das-Dennis approach, a normal-boundary intersection approach that places uniformly spread reference points on a normalized hyperplane based on a pre-defined number of partitions p [29]. If there are M objectives, the hyperplane represents an $M-1$-dimensional unit simplex. The number of reference points H depends on the number of divisions p for each objective axis, where $H = \binom{M+p-1}{p}$ (Figure 2.6).

The resulting reference points are used to associate population members. Figure 2.7 shows the association process. A reference line is drawn from the origin to a reference point (indicated by the dotted lines); the reference line is then used to calculate the distance to the solution points, as shown by the solid lines drawn from the solution points to the reference line. These distances determine which solutions are associated with which reference points, which can then be used to make the selection of which solutions should be included in the next generation.

The selection procedure uses a niche-preservation operation. For each reference point,

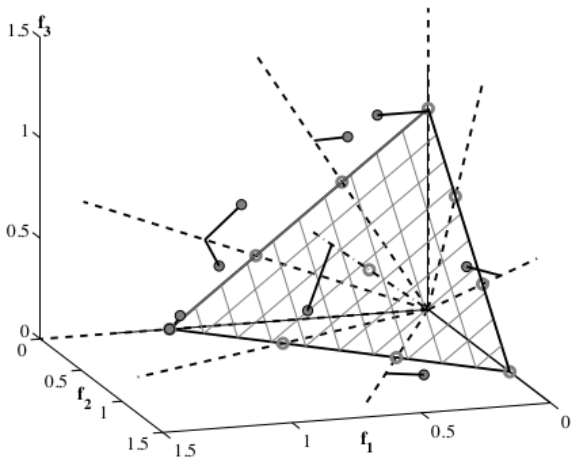


Figure 2.7: Association of reference points to population members [34].

the number of new population members associated with it are counted. The reference point with the lowest count is identified. If this count is zero, the solutions in the last non-domination front are checked to see if some are associated with this reference point, and the closest one is added to the new population. If there are no associated solutions, the reference point is taken out of consideration for the current generation. If there is at least one solution already associated with the lowest-count reference point, a randomly associated solution from the last front is chosen to be added to the new population. The niche count for the reference point is updated, and the procedure is repeated until all empty solution slots in the new population are filled. Once the population has been filled, the algorithm proceeds as a normal evolutionary algorithm (EA), by performing crossover and mutation.

2.2.5 MOO Benchmark Problems

In MOO, several benchmark problems have been established by different research groups, each with different characteristics. Table 2.1 gives an overview of some of the most commonly used multi-objective continuous optimization problems, their notable features, and which paper they were first introduced in. These functions are not scalable in the

Name	Features	Papers
dMOP1	dynamic	Goh and Tan [54]
dMOP2	dynamic	Goh and Tan [54]
dMOP3	dynamic	Goh and Tan [54]
FDA1	dynamic	Farina, Deb, and Amato [47]
FON	severe parameter interaction, disconnected	Van Veldhuizen [152]
KUR	severe parameter interaction, disconnected	Van Veldhuizen [152]
ZDT test suite	2-objective	Zitzler, Deb, and Thiele [184]

Table 2.1: List of multi-objective optimization benchmark functions in alphabetical order with the corresponding papers they were used in and their most notable features.

objective space. Our research aims to investigate the effects of many-objective spaces, therefore we do not use these benchmark functions.

2.3 Many-Objective Optimization

Generally, problems with more than three objectives are referred to as many-objective optimization (MaOO) problems [5]. This distinction is made, since problems with over three objectives seem to increase in difficulty rapidly as the objectives keep growing, requiring more sophisticated methods [67]. Such problems are becoming more prominent in real-world applications; for example, search-based software engineering [120], hybrid car control [106], and automotive engine calibration [95].

2.3.1 Identified Problem Areas

With an increase in competing objectives, the number of non-dominated solutions in the Pareto front increases as well, complicating the search process and resulting in large Pareto fronts. More specifically, the three main identified problems of interest are as follows

[70]:

1. Convergence and diversity are compromised. An increase in objectives often means that almost all solutions in a population are non-dominated, leading to a deterioration in selection pressure and thus convergence.

2. The curse of dimensionality arises in the objective space. As the number of objectives increases, the number of solutions required to approximate the Pareto front increases exponentially.

3. With a large number of solutions, visualization of solutions becomes more difficult or even impossible, as does making a final solution choice (from the perspective of the end-user).

The first two problem areas have been addressed in many different ways, mostly focusing on adjusting algorithms to increase diversity or by adjusting the selection procedure [84]. A commonly used approach is to adjust the Pareto dominance relationship, for example, by increasing the dominance degree. In other words, a non-dominated solution will now dominate more solutions than classic Pareto dominance [174]. Such approaches offer ways to balance convergence and diversity as the objectives increase, but they do not address the issue of large non-dominated solution sets for the end-user to inspect. Most of the research focusing on helping the decision maker in their choice for a final solution focuses on dimensionality reduction to aid in visualization [26] or by incorporating preferences directly into the search processes [56, 86, 154].

2.3.2 Scalable MaOO Benchmark Problems

The benchmark problems presented here are scalable in terms of both the number of variables and the number of objectives to be optimized, lending themselves well to MaOO and LSMOO research. Table 2.2 gives an overview of two of the most commonly used

many-objective continuous optimization benchmarks: the Deb, Thiele, Laumanns, & Zitzler (DTLZ) functions and the Walking Fish Groups (WFG) functions. We summarize their most important features and which paper they were first introduced in.

The following characteristics are used to describe each function:

1. Modality: Functions are multimodal when there are multiple local optima and unimodal when there is a single global optimum. Different objectives in the same function can have different modality.

2. Separability: Functions where all variables interact are non-separable, if there is no interaction between variables, it is considered a separable problem. A question mark indicates that the level of separability is unknown.

3. Bias: Functions where there is a large discrepancy between the distribution of solutions in the search space and the distribution of solutions in the objective space [65].

4. Geometry: Shape of the Pareto optimal front, concave, convex, or linear (both concave and convex); a front can also be disconnected or unknown (indicated through a question mark).

The complete formulation of the DTLZ benchmark problems can be found in Appendix A. The WFG benchmarks are made using the WFG toolkit, where a problem is defined using a vector of parameters x, which are derived from a set of working parameters through different transition vectors, as chosen by the user. These transition vectors are what add complexity to each benchmark problem; Huband *et al.* [65] define two different types of functions: shape functions and transformation functions. There are five different shape functions: linear, convex, concave, mixed convex/concave, and disconnect; and there are three types of transformations: bias, shift, and reduction.

Name	Features	Paper
DTLZ1	separable, multimodal, linear	Deb, *et al.*[38]
DTLZ2	separable, unimodal, concave	Deb, *et al.*[38]
DTLZ3	separable, multimodal, concave	Deb, *et al.*[38]
DTLZ4	separable, unimodal, concave, biased	Deb, *et al.*[38]
DTLZ5	unknown separability, unimodal, unknown geometry	Deb, *et al.*[38]
DTLZ6	unknown separability, unimodal, unknown geometry, biased	Deb, *et al.*[38]
DTLZ7	multimodal, seperable, disconnected	Deb, *et al.*[38]
WFG1	separable, unimodal, convex, biased	Huband, *et al.* [65]
WFG2	non-separable, multimodal, disconnected	Huband, *et al.* [65]
WFG3	non-separable, unimodal, linear	Huband, *et al.* [65]
WFG4	separable, multimodal, concave	Huband, *et al.* [65]
WFG5	separable, deceptive modality, concave	Huband, *et al.* [65]
WFG6	non-separable, unimodal, concave	Huband, *et al.* [65]
WFG7	separable, unimodal, concave, biased	Huband, *et al.* [65]
WFG8	non-separable, unimodal, concave, biased	Huband, *et al.* [65]

Table 2.2: List of multi-objective optimization benchmark functions in alphabetical order with the corresponding papers they were used in and their most notable features.

2.4 Multi-Objective Combinatorial Optimization

Multi-Objective Combinatorial Optimization (MOCO) is a subdomain of MOO looking at discrete variable spaces [25]. In the single-objective space, there are three popular combinatorial optimization problems: traveling salesperson (TSP) [75], knapsack [123], and quadratic assignment (QAP) [82]. Each of these problems can be transformed into different real-world applications. In order to demonstrate their importance, we provide an example of such a real-world problem for each of these benchmarks.

1. TSP is a representation of the order picking problem, which collects a set of products in a warehouse in a minimum amount of time [111].

2. The knapsack problem can be translated to solve traffic congestion in telecommunication, i.e., how we allocate available bandwidth to services which have different characteristics and quality requirements [48].

3. Hospital layout planning is an example of the QAP, where specific locations in the hospital are allocated to clinics by minimizing the total distance travelled by patients [43].

However, the single-objective versions of these problems are limiting, since there may be other relevant objectives to be taken into account, resulting in multi-objective combinatorial optimization. For example, in the case of the order picking problem, reducing the cost of the picking process could be another relevant objective [25]. Due to its broad real-world applicability, MOCO has become a specific area of interest in the MOO community.

2.4.1 Problem Areas

Many single-objective combinatorial optimization problems, including the three problems discussed above, have been proven to be NP-hard [159]. But even with their NP-hard nature, exact solutions can be found for these problems if certain criteria are met (e.g., size limitations). The same does not hold true for the multi-objective instances. Due to its discrete nature, a linear programming weighted sum approach will not be able to provide all feasible solutions within the objective space for MOCO [42]. Since the classic weighted sum approach is unable to provide a complete picture of efficient solutions even in lower objective spaces, it becomes even more important to explore different avenues. As discussed in the previous section, meta-heuristic approaches are a popular method to solve MOO problems. There are two different general classes of meta-heuristic approaches: local search in the

objective space and population-based search, where the population-based approach is more common [7].

2.4.2 MOCO Benchmark Problems

In this section we define the three most common benchmark problems in MOCO [5]:

- Multi Objective Knapsack [67]
- Multi Objective Quadratic Assignment [77]
- Multi Objective Traveling Salesperson [94]

.

The Multi Objective Knapsack problem is defined as follows [67]

$$\max f(x) = (f_1(x), f_2(x), \ldots, f_M(x))$$

$$s.t. \sum_{j=1}^{D} b_{ij} x_j \leq c_i, i = 1, 2, \ldots, M$$

$$x_j \in \{0, 1\}, j = 1, 2, \ldots, D$$

where $f_i(x) = \sum_{j=1}^{D} a_{ij} x_j, 1, 2, \ldots, M$, x is a D-dimensional binary vector, b_{ij} represents the weight of item j inside knapsack i, a_{ij} is the profit of item j inside knapsack i, and c_i is the capacity of knapsack i. This means that the multi-objective part of the problem is defined as having M knapsacks across which D items need to be assigned according to capacity and value.

Knowles and Corne [77] define the Multi Objective Quadratic Assignment Problem:

$$\min_{c^k}(\pi) = \sum_{i=1}^{D} \sum_{j=1}^{D} a_{ij} b^k_{\pi_i \pi_j}, k = 1, 2, \ldots, M$$

where D represents the number of facilities and a_{ij} is defined by an $N \times N$ matrix that contains the distance between locations i and j. Matrix $B = (B^1, \ldots, B^M)$ indicates an mQAP with M flows where $B^o = (b^o_{ij})$, and b^o_{ij} denotes the k-th flow matrix from facility i to j. Lastly, π is the permutation of D facilities, and π_i is the i-th element of π and represents an objective function $o \in \{1, 2, \ldots, M\}$.

Lastly, the Multi Objective Traveling Salesperson problem [94] is defined as finding a tour to minimize cost:

$$\min_{c^k}(\rho) = \sum_{i=1}^{D-1} c^k_{\rho(i),\rho(i+1)} + c^k_{\rho(D),\rho(1)}, k = 1, 2, \ldots, M$$

where D denotes the number of cities visited, $c^k_{i,j}$ represents the cost k for traveling from city i to city j, and ρ is the cyclic permutation of cities, also defined as a tour.

2.5 Co-operative Co-evolutionary Algorithms

Co-evolutionary algorithms divide the population into subpopulations; these subpopulations either can represent part of the solution, or they can represent the entire solution space but only optimize a single objective, thus lending themselves well to solving MOO problems. Two different types of co-evolution exist: competitive and cooperative. The competitive model generally follows the biological predator-prey or host-parasite model [81]. The predator (or host) tries to improve itself to better attack its prey (or to conquer the parasites). In turn, the prey (or parasites) evolve to better protect themselves against these attacks. This is done to set up an evolutionary arms race such that both populations continue to improve, rather than stagnating into a "mediocre stable state" [81].

Cooperative co-evolutionary algorithms (CCEAs) were initially introduced by Potter and De Jong [117]. CCEAs are based on symbiotic relationships found in nature, where different species live together and improve each others' standard of life. To mimic this,

a problem is divided into smaller components, each represented by a different population. In this first version of CCEA, problems with n dimensions are decomposed into n one-dimensional subproblems. These subpopulations are then evolved separately and recombined after evolution to form a complete solution. An individual's fitness is not only based on how well it solves its own part of the problem; it also takes its ability to cooperate with other solutions into account. This is done by injecting the subsolution's variables into a global solution and evaluating the fitness of this full solution.

2.6 Factored Evolutionary Algorithms

Classic CCEA only creates subpopulations that have disjoint sets of variables, i.e., there is no overlap between subgroups (or factors). Haberman and Sheppard proposed including overlap in subpopulations [58], which was then generalized by Strasser *et al.* to create the Factored Evolutionary Algorithm (FEA), which has been shown to perform well on combinatorial optimization problems such as NK-landscapes [138] and Bayesian network abductive inference [50].

FEA initializes subpopulations based on a pre-defined factor architecture, where a factor architecture represents the decomposition of the variables into subgroups. A global solution representing all decision variables is initialized randomly. Then, the following three steps are repeated until a stopping criterion is met: subpopulation optimization using the base-algorithm, competition between overlapping subpopulations, and, finally, sharing of the best sub-solutions. The compete step is executed to decide which variable values from the overlapping population will be added to the global solution, where the current global solution is used to evaluate the subpopulations. Evaluation of the subpopulation happens by injecting the optimized partial solution into the global solution. The competition starts by iterating over the function variables. Each subpopulation containing that variable is then considered, where the relevant variable is substituted into the global solution and evaluated.

The subpopulation resulting in the best fitness for the global solution is saved. Once the algorithm has iterated through the subpopulations for a specific variable, the global solution is updated, and the process is repeated for the next variable.

Butcher *et al.* showed that the compete step in FEA can be considered as a form of Pareto improvement [15]. In single population and disjoint subpopulation approaches, the "hitchhiking" phenomenon is known to occur when trying to find the current best solution. Hitchhiking refers to the scenario where the improved solution has a better fitness score overall than the previous best, but in doing so, individual variables' values could be deteriorating. In terms of Pareto optimality, this means that one aspect is improving while another declines, potentially negatively impacting the overall score and thus not achieving Pareto improvement. The authors argued that when performing competition, each variable is considered separately, and the global solution is only adjusted if the fitness score improves, thus avoiding the hitchhiking pitfall and achieving a Pareto improvement. However, Pareto improvements do not guarantee Pareto optimality, since Pareto optimality implies that there are no more Pareto improvements to be made.

In the last step, the sharing function, the following are repeated for each subpopulation. The variables R_i that are not included in the partial solution S_i are set to be equal to those of the global solution, i.e. $R_i = G \setminus S_i$. The worst individual from the subpopulation is chosen and its values are replaced by those found in the global solution. The worst individual's fitness is then reevaluated using these new values. These steps are repeated until a certain number of FEA iterations have been completed or a convergence criterion is met.

CHAPTER THREE

MULTI-OBJECTIVE FACTORED EVOLUTIONARY ALGORITHM

This chapter introduces the novel Multi-Objective Factored Evolutionary Algorithm Framework (MOFEA), which is based on the ideas presented by Strasser *et al.* in FEA [138]. We apply the proposed framework with NSGA2 [36] and present results on two variations of the multi-objective knapsack (MOKS) benchmark with 1000 variables and three and five objectives.

3.1 Related Work and Motivation

Our MOO research focuses on improving exploration in high-dimensional objective and variable spaces. We use the classic MOKS problem, as well as a more complex variation of MOKS, to evaluate the effectiveness of our proposed framework. Since a large amount of research has been performed to improve exploration in different types of multi- and many-objective optimization [7, 97, 145], we focus our related work discussion on two aspects: the use of co-operative co-evolution in MOO and research looking at the MOKS problem specifically.

Before diving into these specific aspects, we would like to note that there has been work done that includes the idea of Pareto optimality to improve single-population optimization, inspired by the FEA algorithm [15]. The authors proposed altering the g-best strategy of PSO through the idea of Pareto efficiency. Instead of selecting the best individual from the population to represent the g-best solution, the g-best solution was constructed variable by variable. This was accomplished as follows: for each variable, each solution in the population's value for that variable was injected into the current g-best, and the value was only changed if the fitness score improved, thus achieving Pareto improvement.

The resulting g-best solution is not an existing particle in the population, and was also not injected back into the population. In this approach, the g-best solution served as a communication mechanism to share information between the particles in the swarm, similar to how information is shared in FEA.

3.1.1 Subpopulations in Multi-Objective Optimization

CCEA has been applied to MOO in several different studies, and the resulting algorithms are called cooperative co-evolutionary multi-objective evolutionary algorithms (CCMOEA) [100]. The first combination of the Multi-Objective Genetic Algorithm (MOGA) [49] and CCEA [117] was presented by Keerativuttitumrong *et al.* [76]. They found that a co-operative approach can have beneficial results for finding a well spread out Pareto front when compared to the single population MOGA. This finding was confirmed in a follow-up study, where CCEA was applied to four different base algorithms (MOGA, NSGA2, a controlled elitist NSGA, and a niched Pareto genetic algorithm) and compared to single population alternatives of each algorithm [100].

Another study had similar results when applying CCEA to a multi-objective particle swarm optimizer (MO-PSO) [55]. Goh *et al.* found that their approach maintained diversity while finding good approximations to the Pareto front, with the exception of multi-modal problems. Dorronsoro *et al.* applied the co-operative co-evolutionary versions of NSGA2, SPEA2, and the Multi-objective Cellular Genetic Algorithm (MOCGA) to the combinatorial optimization problem of grid computing [39]. The authors found that CC-SPEA2 outperformed CC-NSGA2 and CC-MOCGA as well as their single-population alternatives. All of the above research confirms the benefit of using co-operative co-evolution in MOO; however, groups generated for the CC methods are still disjoint, which could lead to problems when dealing with partially separable functions.

A different subpopulation strategy used a one-on-one relationship between a subpopula-

tion and scalarized objectives [175]. The multi-guide particle swarm optimization (MGPSO) algorithm was introduced by Scheepers *et al.* which introduced objective decomposition into MOPSO [129]. They created subswarms, where each subswarm optimizes a single objective. They found their approach to be competitive with other MOEA's and provided a theoretical stability analysis of their algorithm.

Liang *et al.* also used the idea of evolving a single subpopulation for a single objective, using the Chebycheff approach to decompose the multi-objective problem into m single-objective problems [90]. An external archive was kept that keeps track of how the best solutions for each subpopulation are performing on all objectives. This external archive was then used to create offspring for all subpopulations, thus ensuring that solutions from one subpopulation are shared across all others. They applied their approach to two and three objective MOO benchmarks and found that the proposed method performs well, outperforming regular MOEA/D, NSGA2, SPEA2, CC-PSO, and CC-based Differential Evolution on 25 out of 31 test problems.

Lastly, a competitive co-evolutionary algorithm based on objective decomposition was proposed by Vu *et al.* [147]. The Dual-Population Competitive Co-Evolutionary Approach (which the authors abbreviate as DPPCP) used each population to perform one of two tasks: Pareto-based ranking to push the algorithm to convergence or objective-wise decomposition to increase diversity. In other words, one subpopulation applied NSGA2, while the other subpopulation applied MOEA/D. Each population kept their own archive and these archives were used to create offspring. The common offspring were then used to update each subpopulation using their respective approaches. DPPCP was applied to the standard continuous MOO benchmark suite using two and three objectives and was compared to NSGA2 and MOEA/D. DPPCP outperformed the other algorithms on 26 out of 32 problems for hypervolume and inverted generational distance.

These objective decomposition approaches offer a way to solve many-objective opti-

mization problems more efficiently, but the results are dependent on the applied objective decomposition strategy [175]. Additionally, when evolving subpopulations along different objective directions, this could result in parts of the search-space remaining entirely unexplored and in the algorithm getting stuck in local optima [146]. Furthermore, these approaches can work with the proposed framework; MOEA/D can be used as the base-algorithm, thus combining objective decomposition and variable decomposition.

3.1.2 The Multi-Objective Knapsack Problem

In an adjustment of the NSGA3 approach, Sahinkoc and Bilke used a fixed hyperplane and used subpopulations to evolve along the different objectives [124]. The fixed hyperplane was introduced to address the problem of the evenly spread reference points guiding the solutions in the wrong direction. To accomplish this, an optimal solution for each single objective was calculated and used as the fixed edge points of the hyperplane. They evaluated their method on the many-objective knapsack, solving a 500 item knapsack problem with 6, 8, 10, 15, 20, and 30 objectives. They found that including a fixed hyperplane significantly improved results for all NSGA3 implementations. The co-operative approach evolving along the different objectives improved the results further, and their proposed algorithm had the best performance on instances with a large number of objectives. The authors did note that finding the optimal solutions beforehand may not always be feasible, but they claimed that near optimal solutions would suffice.

Zouache *et al.* proposed a novel "cooperative" swarm intelligence algorithm for MOO; however, the term "cooperative" here does not refer to CCEA but to the combination of the firefly algorithm with particle swarm optimization (MOFPA) [188]. They applied their approach to a knapsack with 250, 500, and 750 items, optimizing two, three, and four objectives, resulting in $3 \times 3 = 9$ different knapsack problems. Their results indicated that their proposed hybrid algorithm performed better in terms of coverage on all instances of the

knapsack problem studied when compared to NSGA2, MOEA/D, and SPEA-II. However, the inverse generational distance metric, which compares the found Pareto front to the known optimal Pareto front, was not significantly different across any of the algorithms.

Ishibuchi *et al.* looked at four MOEA's performance on the multi-objective knapsack problem [67]. NSGA2, MOEA/D, SMS-EMOA, and HypE were applied to the knapsack problem with 2-10 objectives. NSGA2 was further adjusted to perform what the authors call a focused search. This means that there is a separate population exploring each objective, i.e., objective-wise decomposition, and the algorithm looks for non-dominated solutions near the best solution for the objective at hand. However, there is one more population assigned to look for non-dominated solutions around the center of the Pareto front as well. Two different versions of this focused search were implemented: F100 and F90. In F100, NSGA2 optimized each of the subpopulations entirely separately and merged the found solutions into one population at the very end. Alternatively, F90 used 90% of the computation time to optimize the subpopulations. The found solutions were then merged into one population, and NSGA2 was applied to this merged population for the remaining 10% of computation. They found that while regular NSGA2 performed well on two-objective knapsack, the focused search alternative of NSGA2 and MOEA/D performed well on the many-objective 10,000 item knapsack problems. The authors specifically noted that more focused research applying NSGA2 to subpopulations is a worthwhile endeavor.

3.2 Multi-Objective Factored Evolutionary Algorithm

Instead of designing a novel meta-heuristic algorithm, we proposed an approach to increase exploration of the search space that can be applied to any MOEA: the multi-objective factored evolutionary algorithm (MOFEA). MOFEA is a framework that divides the population into subpopulations with overlapping variables, it uses the chosen MOEA to optimize the subpopulations and combines the resulting sets of non-dominated solutions

through competition and sharing. MOFEA is an extension of the FEA framework introduced by Strasser *et al.* [138]. Furthermore, the use of sub-populations lends itself to parallellization and can thus be used to reduce computation time, which is desirable when dealing with large-scale MOO problems. We do not apply parallellization to the research in this dissertation, so we cannot talk about the effects of distributed computing in regards to MOFEA.

Classic CCEA only creates subpopulations that have distinct variables, i.e., there is no overlap between subpopulations. FEA uses overlapping subpopulations, which requires using principles from both co-operative and competitive co-evolution. We define an individual $\mathbf{X} \leftarrow \{x_0, \ldots, x_n\}$, where n is the dimensionality of the problem. We define a set of subpopulation $\mathcal{S}$, $\forall S \in \mathcal{S} : S \leftarrow \{\mathbf{U_0}, \ldots, \mathbf{U_p}\}$. Where p is the population size, and $\mathbf{U}$ represents a single population member, where $\mathbf{U} \subset \mathbf{X}$ as defined by the factor architecture. We have extended FEA to use any MOEA algorithm; we show the pseudocode in Algorithm 3.1. Originally, FEA kept a single global solution G; however, in MOO, the goal is to obtain an approximate Pareto front. Therefore, we keep a set of global solutions $\mathcal{G}$ as well as an archive of all non-dominated solutions. Initially, each subpopulation is assigned the same global solution G to evaluate the total fitness of its individuals; however, as the algorithm progresses, a random global solution $G \in \mathcal{G}$ out of the solution set is chosen for each subpopulation. We denote the specific global solution of a subpopulation S as G_S. Each subpopulation is optimized using the MOEA of choice, which returns a set of non-dominated solutions $\mathcal{N}''$. In FEA, subpopulations representing the same variable compete to represent that variable in the global solution G, which represents the full set of variables. In MOFEA, the "Compete" step (lines 9–20) is altered by allowing each representing subpopulation to contribute three non-dominated solutions to a temporary archive. All representing subpopulations are analysed for each variable. First, a random non-dominated solution from $\mathcal{N}''$ is selected and added to the temporary archive. Next, the "best" solution according to the criteria ($\arg\min_{\text{criteria}}$) of the chosen MOEA is selected from $\mathcal{N}''$. For NSGA2, SPEA2, and MOEA/D, the sorting

Algorithm 3.1 Multi-Objective Factored Evolutionary Algorithm

Input: number of variables n, population size p, number of objectives M, objective functions $F \leftarrow \{f_0, \ldots, f_M\}$, subpopulations $\mathcal{S}$
Initialize: global solution $G_0 \leftarrow \{x_0, \ldots, x_n\}$, global solution set $\mathcal{G} \leftarrow \{G_0\}$, non-dominated archive $\mathcal{N} \leftarrow \{\}$

while stopping criterion is not met **do**
  $it = 0$
  for all $S \in \mathcal{S}$ **do**
    if $it = 0$ **then**
      $G_S \leftarrow G_0$
    // Optimize subpopulation
    $\mathcal{N}''_S \leftarrow \text{MOEA}(S, G_S, F)$
  // Compete step
  $\mathcal{N}' \leftarrow \{\}$
  for all $i = 1$ to n **do**
    for all $S \in \mathcal{S}$ where $\mathbf{U} \in S : x_i \in \mathbf{U}$ **do**
      $\mathcal{N}' \leftarrow \mathcal{N}' \cup \text{random}(\mathcal{N}''_S))$
      $X \leftarrow G_S$
      $N'' \leftarrow \mathcal{N}''_S[\arg\min_{\text{criteria}}])$
      $\mathcal{N}' \leftarrow \mathcal{N}' \cup N''$
      $X_i \leftarrow N''_i$
      $\mathcal{N}' \leftarrow \mathcal{N}' \cup X$
  $\mathcal{N}' \leftarrow \text{non-dominated}(\mathcal{N}')$
  $\mathcal{G} \leftarrow \mathcal{N}'$
  $\mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{N}'$
  // Share step
  for all $S \in \mathcal{S}$ **do**
    // update global solution
    $G_S \leftarrow \text{random}(\mathcal{G})$
    $S[\arg\max_{\text{criteria}}] \leftarrow G_S$
  // Update archive
  $\mathcal{N} \leftarrow \text{non-dominated}(\mathcal{N})$
  $it++$
return $\mathcal{N}$

criteria are as follows:

- NSGA2: crowding distance,
- SPEA2: strength value,
- MOEA/D: decomposed fitness value.

The decision variable in G_S of the relevant subpopulation is replaced by the decision variable of the selected solution and is saved in a temporary solution set $\mathcal{N}'$. Furthermore, the original chosen solution is also added to $\mathcal{N}'$. The procedure that adjusts G_S is visualized in Figure 3.1. When dealing with disjoint subpopulations, i.e., there is no overlap, this means that solutions from only one subpopulation will be added to $\mathcal{N}'$. We claim adding these three different solutions to $\mathcal{N}'$ for evaluation helps improve exploration of the solution and objective spaces.

$\mathcal{N}'$ is evaluated for non-dominance once all the variables and corresponding subpopulations have been processed (line 18 in Algorithm 3.1), and $\mathcal{N}'$ is now used as the new set of global solutions. For each subpopulation, a randomly chosen $G \in \mathcal{G}$ is injected into the subpopulation. G is selected at random without replacement, unless there are less found non-dominated solutions than there are subpopulations. In this case, the random selection process will restart after all global solutions have been assigned at least once. The individuals in the subpopulations are re-evaluated based on the new global solution, and the resulting "worst" solution in each subpopulation ($\arg\max_{\text{criteria}}$) is replaced by G, completing the "Share" step (lines 23–25), as illustrated in Figure 3.2. This is one iteration of FEA. The algorithm repeats until a stopping criterion is met.

To show that this framework works for distinct subpopulations, we use the CCNSGA approach as proposed in [100] as an example. The authors note two main adjustments to NSGA to create the co-operative approach: 1) integration of NSGA into CCEA and 2) the use of an elitist strategy to pass individuals to the next generation. The first adjustment

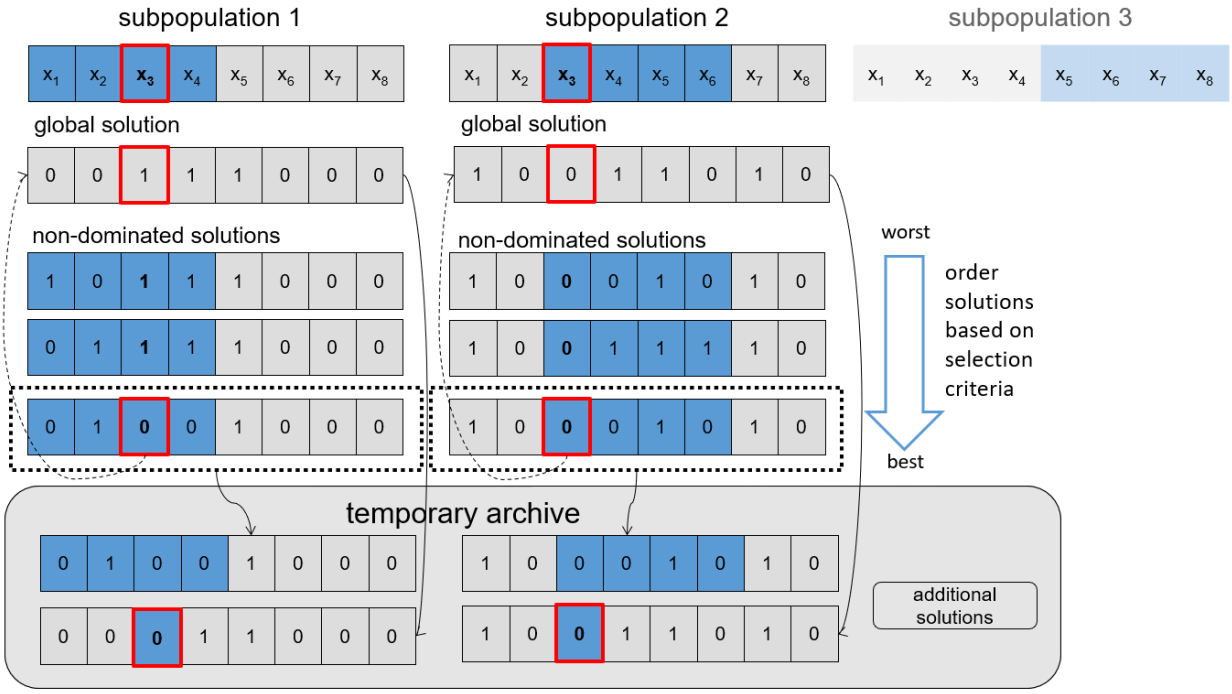


Figure 3.1: The modified compete step as performed on variable x_3 in the Multi-Objective Factored Evolutionary Algorithm. The dotted outline shows the selected solution from the subpopulation's non-dominated solution set. Both the entire solution and the solution where only x_3 is replaced are included in the temporary archive.

uses NSGA's crowding distance to make a selection from the non-dominated solutions found by each subpopulation to regulate size. The compete step of MOFEA does exactly that; however, we generalized this approach by allowing for any selection criterion to be used depending on the MOEA used to optimize the subpopulation. For the elitist strategy, the authors aim to carry over non-duplicate non-dominated solutions to the next iteration. In MOFEA, this is accomplished through the share step, where the unique non-dominated individuals are inserted into the subpopulations for the next iteration.

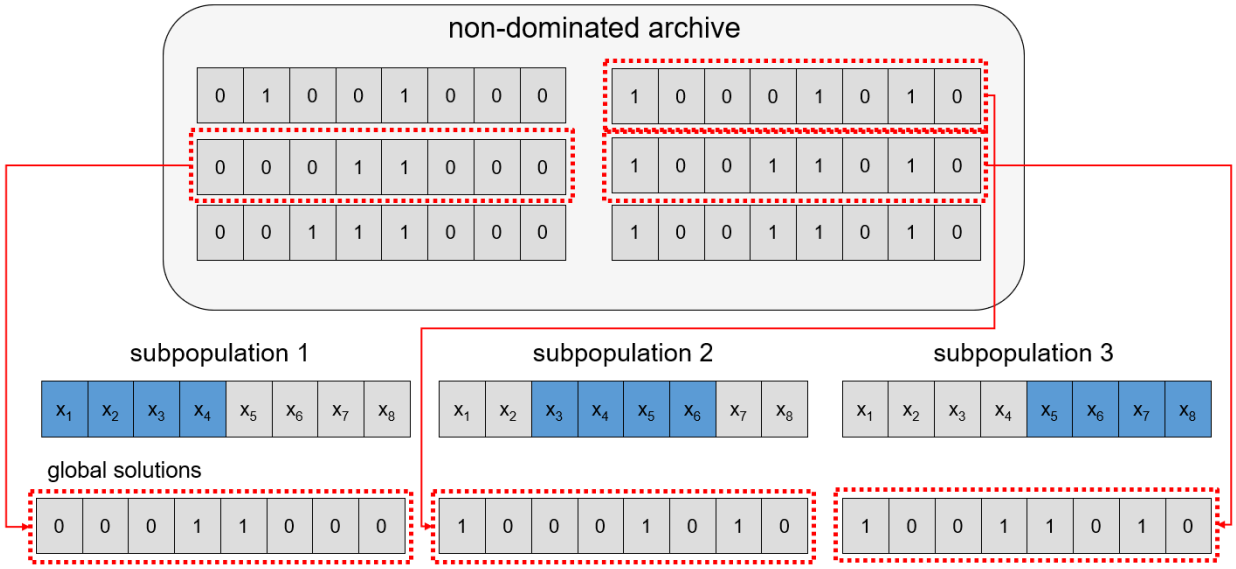


Figure 3.2: The modified share step of MOFEA. For each subpopulation, a random global solution is selected (without replacement) from the current iteration's set of found non-dominated solutions.

3.3 Multi-Objective Knapsack Problem Experiments

A popular benchmark problem that relates to many real world applications is the Multi-Objective 0-1 Knapsack (MO-KS) problem [186]. This problem was proposed as a benchmark for multi-objective combinatorial optimization (MOCO). It makes for a good benchmark since it can be adapted in terms of number of objectives, constraints, and variables. However, increasing the number of knapsacks does not necessarily relate to a real-world application [74]. To this end, a different multi-objective knapsack problem exists that looks at a single knapsack but minimizes the difference in resources (e.g. weight) in the knapsack, creating a balanced knapsack [51]. Our preliminary results look at the three and five objective versions of each of these multi-objective knapsack problems.

3.3.1 Experimental Approach

We applied the Non-Dominated Sorting Genetic Algorithm-II (NSGA2) to the proposed MOFEA framework [36]. NSGA2 is a popular approach that has proven to work well on problems with up to three objectives but starts declining in performance when the objectives increase further. The same authors therefore proposed NSGA3, which uses the aforementioned reference directions to increase performance [34]. Since we randomly initialized our problem instances, we have no prior knowledge of the problem. Furthermore, Carvalho and Britto found that the chosen reference points can positively or negatively impact the results found by NSGA3, indicating that an automatically initialized set of reference directions may not be a desirable approach [19]. Because of this, and in order to more clearly show the benefit of the MOFEA framework, we used NSGA2 instead of NSGA3 in our experiments.

3.3.1.1 Multi-Objective Knapsack Problem We adjusted the classic Multi Objective Knapsack (MOKS) problem [67] by changing the objectives of a single knapsack instead of defining multiple knapsacks. This adjustment was based on the problem defined by Fortin *et al.* [51]. The MOKS problem tries to maximize value, minimize weight, and minimize the difference in weight of the items in a single knapsack, while including constraints placed on the volume and weight of the knapsack using the same method as the multi-knapsack problem. An additional objective to balance the weights is defined as follows:

$$\min \sum_{j,k=1,j\neq k}^{D} |b_j x_j - b_k x_k|.$$

Fortin *et al.* call this type of multi-objective knapsack a balanced knapsack. To add two more objectives, we extended this problem to include minimization of the overall volume v_j of the items in the knapsack and minimizing the difference in volume. In other words, instead of

optimizing multiple knapsacks simultaneously, we are trying to find the optimal combination of items being added to a single knapsack but with multiple competing objectives in regards to the items. An example of a real-world problem that could benefit from such a balancing approach is the loading of a cargo plane; when there is a large discrepancy in item weights, it makes it more difficult to balance the cargo hold.

In our experiments, we used a 1000 item knapsack to test the algorithms' performance at a larger scale. For the balanced knapsack, we looked at the three base objectives (value, weight, and volume) and the extended five objective version (balanced weights and volume). The original MOKS uses fully randomized initialization of the values and weights for all knapsacks and constraints [186]. Therefore, we applied the same approach and initialized the values, weights, and volume randomly as follows: $a_i = [0.1, 100], b_i = [0.1, 5]$, and $v_i = [0.1, 10]$. For the classic knapsack problem, we initialized different sets of values and weights based on the number of objectives and constraints, using the same values as above. For this problem, we considered three and five objectives with a single weight constraint c_k.

3.3.1.2 Hyperparameter Tuning We performed a grid search to tune NSGA2 using the following parameter values:

- Algorithm runs: $50, 100, 200, 500$
- Population size: $250, 500, 750, 1000$
- Mutation rate: $0.10, 0.15, 0.20, 0.25$
- Crossover rate: $0.85, 0.90, 0.95, 0.98$

Based on this grid search, we found that running NSGA2 100 times with a population of 500 was the best combination. For the GA operators, we used tournament selection with $k = 5$ to select the parents, a mutation rate of 0.2 using bitflip mutation and a crossover rate of 0.95

with single-point crossover. The found hyperparameters were used in the FEA and CCEA implementations of NSGA2 as well. Furthermore, the FEA and CCEA implementations were run for 20 iterations, with two different sizes of subpopulations: 100 and 200, and a 20% overlap for F-NSGA2, i.e., 20 variables and 40 variables overlap for each subpopulation of size 100 and 200 respectively.

3.3.1.3 Evaluation Metrics As discussed in Chapter 2 Section 2.2.3, the hypervolume indicator (HV) is one of the most commonly used evaluation metrics in MOO [9]. Its popularity is partially because the only information needed to calculate the HV of a Pareto Front approximation is a single reference point. This is in contrast to measures such as Generational Distance, which requires the true Pareto Front to be known. Since we do not know the true Pareto Front for our problems, the HV is a natural choice to gain insight in the size of the covered objective space [183].

To assess the diversity of the Pareto Front approximations, we used the spread indicator S [69]. Lastly, to compare two Pareto fronts generated by different algorithms directly, we calculated the coverage C of the fronts, denoted as $\mathbf{X}'$ and $\mathbf{X}''$ [184]. We further adjusted this metric to find relative coverage of the non-dominated sets as compared to the total non-dominated set. The total non-dominated set, or union front $\mathbf{X}^*$, is created by combining the results from the set of algorithms g as

$$\mathbf{X}^* = \text{nondom}\left(\bigcup_{i=1}^{g} \mathbf{X}'_i\right).$$

$\mathbf{X}^*$ can then be used to calculate what percentage of each base non-dominated set is included in $\mathbf{X}^*$: $C(\mathbf{X}', \mathbf{X}^*)$. To calculate what percent of $\mathbf{X}^*$ consists of solutions from $\mathbf{X}'_i$, we adjust the coverage calculation as follows, creating Adjusted Coverage (AC):

$$AC(\mathbf{X}', \mathbf{X}^*) = \frac{|\{\mathbf{x}' \in \mathbf{X}' : \exists \mathbf{x}^* \in \mathbf{X}^* : \mathbf{x}^* \preceq \mathbf{x}'\}|}{|\mathbf{X}^*|}$$

		single knapsack		multi knapsack	
	pop.	**3 obj.**	**5 obj.**	**3 obj.**	**5 obj.**
NSGAII	**500**	12.18	12.01	20.99	34.16
CC-NSGAII	**100**	12.24	10.87	17.30	28.76
	200	9.42	10.40	15.99	28.31
F-NSGAII	**100**	12.28	11.92	23.09	39.63
	200	12.22	11.97	22.72	38.56

Table 3.1: Hypervolume results. Underlined results indicate statistically significant results.

3.3.2 Results

We ran each algorithm ten times on each of the problem sets and averaged the ten runs of the HV and spread indicator results, as well as the size of the non-dominated population (Tables 3.1, 3.2 and 3.3). Furthermore, we performed an ANOVA test with $\alpha = 5\%$, followed by a paired T-test with $p = 0.05$ to test statistical significance of the results. CC-NSGA2 results were not found to be significantly different when comparing variable group sizes of 100 and 200, with the exception of the three objective single knapsack problem. The opposite is true for F-NSGA2 results, which were found to be significantly different from the other algorithms' results. Lastly, fewer statistically significant differences were found between the different HV results, whereas the spread indicator results were largely statistically significant.

To examine the coverage between different solution sets, we randomly pick a single representative run of each algorithm to perform the coverage calculation. To ensure the results are not biased, we perform this process 10 times and average the coverage comparisons for the final results. We do this to avoid the combinatorial explosion that would result from averaging all combinations of the runs. We present two different coverage results: standard coverage and adjusted coverage. The adjusted coverage (Table 3.4) represents the percentage of non-dominated solutions each algorithm contributed to the combined non-

		single knapsack		multi knapsack	
	pop.	**3 obj.**	**5 obj.**	**3 obj.**	**5 obj.**
NSGAII	**500**	38.03	38.24	29.55	32.09
CC-NSGAII	**100**	13.39	8.15	5.89	7.09
	200	5.77	8.58	4.34	6.76
F-NSGAII	**100**	31.34	24.18	15.34	16.91
	200	33.32	30.80	10.39	10.76

Table 3.2: Spread indicator results. Underlined results indicate statistically significant results.

		single knapsack		multi knapsack	
	pop.	**3 obj.**	**5 obj.**	**3 obj.**	**5 obj.**
NSGA2	**500**	166	519	16	38
CC-NSGA2	**100**	482	521	21	61
	200	156	476	28	53
F-NSGA2	**100**	642	1334	8	14
	200	698	1383	5	9

Table 3.3: Size of the non-dominated solution sets.

dominated solution set. Tables 3.5–3.8 show a direct comparison between the algorithms' non-dominated solution sets, where the row algorithm's solution set covers $x\%$ of the column algorithm's solution set.

Finally, we visualized the three-objective versions of the balanced knapsack and the multi knapsack problems in Figure 3.3. Each of these figures shows the final non-dominated population of a randomly selected run of each of the algorithms, where the x-axis for the single balanced knapsack and the x, y, and z-axes for the multi-knapsack show negative

		single knapsack		multi knapsack	
	pop.	**3 obj.**	**5 obj.**	**3 obj.**	**5 obj.**
NSGA2	**500**	0.64%	14.28%	0.00%	0.00%
CC-NSGA2	**100**	42.69%	16.90%	0.00%	0.00%
	200	0.00%	11.70%	0.00%	0.00%
F-NSGA2	**100**	12.20%	32.85%	100.00%	98.67%
	200	44.47%	24.26%	0.00%	1.13%

Table 3.4: Adjusted coverage results.

		NSGA2	**CC-NSGA2**		**F-NSGA2**	
	pop.	**500**	**100**	**200**	**100**	**200**
NSGA2	**500**	*N/A*	83.50%	97.21%	28.14%	10.5%
CC-NSGA2	**100**	100.00%	*N/A*	100.00%	97.85%	98.24%
	200	25.56%	24.92%	*N/A*	31.55%	5.03%
F-NSGA2	**100**	85.50%	81.47%	96.12%	*N/A*	35.86%
	200	95.76%	78.27%	99.47%	87.02%	*N/A*

Table 3.5: Single balanced knapsack 3 objectives coverage results

values due to the transformation of the knapsack profit maximization to a minimization problem.

3.3.3 Discussion

F-NSGA2 improved the hypervolume results for three of the four problems: the three-objective balanced knapsack problem and the three- and five-objective multi-knapsack problems. No statistically significant differences were found for the five-objective single knapsack problem. When looking at the spread indicator results, however, we did find

		NSGA2	**CC-NSGA2**		**F-NSGA2**	
	pop.	**500**	**100**	**200**	**100**	**200**
NSGA2	**500**	*N/A*	99.25%	99.51%	96.45%	76.19%
CC-NSGA2	**100**	76.24%	*N/A*	99.02%	95.76%	96.90%
	200	64.72%	96.14%	*N/A*	73.48%	90.41%
F-NSGA2	**100**	65.06%	98.82%	94.69%	*N/A*	64.42%
	200	56.22%	100.00%	100.00%	73.29%	*N/A*

Table 3.6: Single balanced knapsack 5 objectives coverage results.

		NSGA2	**CC-NSGA2**		**F-NSGA2**	
	pop.	**500**	**100**	**200**	**100**	**200**
NSGA2	**500**	*N/A*	100.00%	100.00%	0.00%	0.00%
CC-NSGA2	**100**	0.00%	*N/A*	59.57%	0.00%	0.00%
	200	0.00%	100.00%	*N/A*	0.00%	0.00%
F-NSGA2	**100**	100.00%	100.00%	100.00%	*N/A*	100.00%
	200	100.00%	100.00%	100.00%	0.00%	*N/A*

Table 3.7: Multi knapsack 3 objectives coverage results.

statistically significant differences for all four problems, but single-population NSGA2 has a higher spread indicator for each of them. However, when we look at our coverage results, F-NSGA2 contributed a larger percentage to each problem's combined Pareto front. Based on these results, it appears that the spread indicator may have little influence on the quality of the solution set. It is important to note that this is only based on an observed lack of correlation between coverage and spread. These results are by no means conclusive, but warrant further investigation.

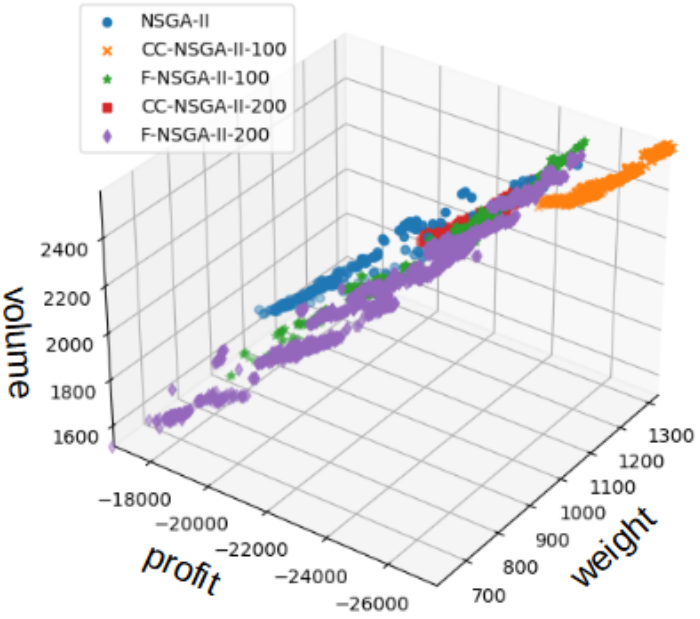
A visual inspection of Figure 3.3a shows that both instances of F-NSGA2 and the 100-

		NSGA2	**CC-NSGA2**		**F-NSGA2**	
	pop.	**500**	**100**	**200**	**100**	**200**
NSGA2	**500**	*N/A*	100.00%	100.00%	0.00%	1.87%
CC-NSGA2	**100**	9.40%	*N/A*	80%	0.00%	0.00%
	200	0.00%	80.00%	*N/A*	0.00%	0.00%
F-NSGA2	**100**	100.00%	100.00%	100.00%	*N/A*	100.00%
	200	100.00%	100.00%	100.00%	2.50%	*N/A*

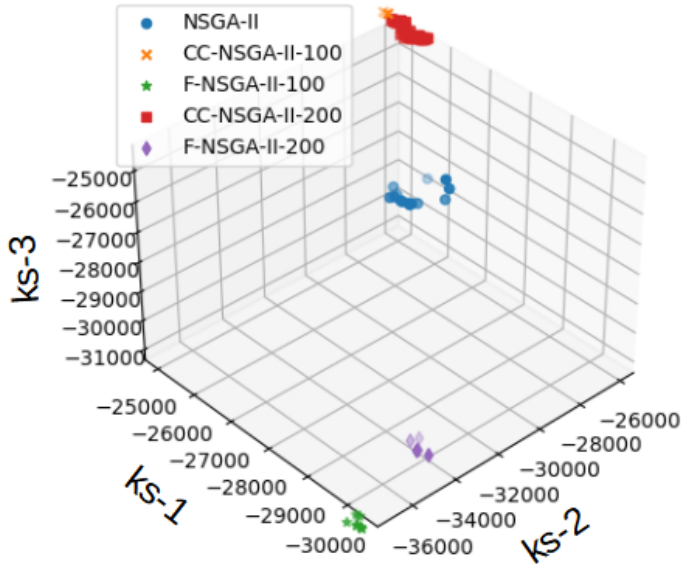
Table 3.8: Multi knapsack 5 objectives coverage results.

population instance of CC-NSGA2 cover more of the space than regular NSGA2. Taking into consideration that for this problem instance CC-NSGA2-100 and F-NSGA2-200 have the highest contribution to the total non-dominated solution set, the visual representation makes sense. CC-NSGA2-100 found solutions in a different part of the space than the other algorithms, but its solutions are not as widely spread across the solutions space, whereas F-NSGA2 has a significantly larger spread than CC-NSGA (Tables 3.1 and 3.2), potentially accounting for its large contributions to the total non-dominated solution set. However, NSGA2 has the largest spread indicator while contributing less than 1% of the non-dominated solutions to the total front (Table 3.4).

Figure 3.3b shows a different story. With the exception of the two CC-NSGA2 instances, each of the discovered non-dominated solutions is in a different part of the objective space. When looking at Tables 3.4 and 3.7, the non-dominated solutions discovered by F-NSGA2 with subpopulation size 100 covers all other algorithms' non-dominated solutions. The F-NSGA2-100 solutions are represented by the green star shaped cluster at the very bottom corner of the image. Since each objective was to be minimized, it makes sense that these solutions, which have converged at the lowest values of the three knapsacks, are dominating the other solutions. This is especially interesting given that the average number of non-



(a) Single balanced knapsack with three objectives.



(b) Classic multi knapsack problem with three objectives.

Figure 3.3: Visual representation of the non-dominated population found by each of the algorithms for the three objective versions of the two types of knapsack problems.

dominated solutions found by F-NSGA2 for this problem is smaller than the other algorithms (Table 3.3).

When considering the five-objective problems, F-NSGA2 with subpopulation size 100 once again contributed the largest percentage to the total Pareto front for both problems. Interestingly, NSGA2's results improved for the five-objective single knapsack, contrary to the general trend found in the literature, where NSGA2's performance is often found to deteriorate as objectives increase. When looking at the direct coverage comparison for this problem (Table 3.6), NSGA2 covers larger percentages of the solutions found by the four other algorithms; however, it is only contributing 14.28% to the total front. The larger contribution to the total Pareto front by F-NSGA2 could be explained by the larger number of non-dominated solutions found by the algorithm as compared to NSGA2 (Table 3.3).

The last problem we evaluated was the five-objective multi knapsack. The F-NSGA2 with a subpopulation size of 100 was the main contributor to the total Pareto front, and when looking at the pairwise comparison, both F-NSGA2 solution sets cover the three other algorithms' solution sets. Another interesting result is that CC-NSGA2 performed poorly on both multi-knapsack problems, in that its results are not only 100% covered by F-NSGA2, but by single population NSGA2 as well. Overall, both instances of F-NSGA2 did well on all four benchmark problems, indicating that using overlapping subpopulations is beneficial for multi-objective optimization. The use of a smaller subpopulation size in F-NSGA2 seems especially beneficial for finding non-dominated solutions.

3.4 Concluding Remarks

We developed and presented a cooperative co-evolutionary framework for solving multi-objective combinatorial optimization problems that divides the population in subpopulations, which can be disjoint or overlapping, and allows usage of any population-based multi-objective algorithm as the base algorithm. We applied our approach to two different

implementations of the multi-objective knapsack problem: the multi-knapsack problem, where the number of objectives equals the number of knapsacks [186], and the balanced single knapsack problem [51]. We compared overlapping and distinct subpopulations of different sizes as applied to NSGA2, as well the single population version of NSGA2.

MOFEA using NSGA2 performed well on both types of knapsack problems, where the discovered approximate Pareto front covers a high percentage of those found by the other algorithms. The HV results were similar across the different algorithms, where F-NSGA2 did hold a small edge over the other results, which indicates that there is room for improvement. We hope to gain more insight into the effects of using overlapping subpopulations by using SPEA2 and MOEA/D as the base-algorithms for MOFEA in addition to NSGA2. Furthermore, this research only considered static, pre-defined factor architectures. However, the decomposition strategy could influence the results, which is why performing an in-depth study of different decomposition techniques could provide better insight into how the factor architecture used in MOFEA affects MOO problems. The next chapter does just that: it explores the effect of overlapping subpopulations on both single- and multi-objective continuous optimization problems using different variable grouping strategies.

CHAPTER FOUR

INFLUENCE OF VARIABLE GROUPING ON LARGE-SCALE OPTIMIZATION

Now that we have established that the MOFEA framework can help improve results in MOO, we wanted to further investigate the effects of using overlapping subpopulations on different problems and different algorithms. However, before diving into MOO-specific research, we look at the effects of overlap on single-objective Large Scale Optimization (LSO) using different grouping strategies. This was a logical first step, since many MOO problems are of a large-scaled nature, and this research allows us to examine the effect of different variable decomposition techniques without the added difficulty of multiple objectives.

4.1 Problem Decomposition

Before providing related work in the area of variable grouping, we give more background information on different grouping strategies. There are three main approaches to variable grouping in CCEA: static grouping, random grouping and grouping based on variable interaction. Ma *et al.* [97] provide a comprehensive overview of different decomposition strategies for CCEA. They note that dynamic group assignments based on variable interaction learning improves CCEA results compared to random groupings. Five different groups of interaction learning based decomposition are defined by Ma *et al.*: perturbation (DG), statistical model (MEE), distribution model (FEA), approximate model, and linkage adaptation. The latter two are not considered in this work since approximate modeling is used for problems where evaluating the objective function is too expensive and needs to be approximated [126], and linkage adaptation methods influence operators of EA's directly. For example, Schaffer and Morishima [127] add a punctuation flag to the chromosome to indicate the crossover point, thus effectively grouping each chromosome.

4.1.1 Static and Random Grouping

An important aspect of cooperative co-evolutionary algorithms is the way the problem is decomposed into subgroups or factors. CCEA started with static grouping, initially creating n groups of one dimension for a problem with n variables. This was expanded to creating m groups with s variables, where the variables are split up sequentially [150]. However, such static groupings mean that if non-consecutive variables are interacting, they will not belong to the same group. A first approach to solve this problem is to perform random grouping of the variables [169, 170]. This is done by randomly selecting s population members to belong to m groups, which can be a one-time static assignment, or dynamically altered each generation. The number of groups m, and by consequence the number of members s in a group, can be a fixed parameter or dynamically altered. If m is fixed: $s = n/m$, where n is the total population size and the s variables are randomly chosen to belong to each group [170]. When dynamically choosing the number of groups, the s parameter is randomly set each co-operative co-evolutionary iteration [169]. A range of values for s needs to be set so the grouping is feasible given the population size.

4.1.2 Variable Interaction

Random grouping does not guarantee that interacting variables will be grouped together. This is where studying variable interaction could provide a solution. Chen *et al.* [20] explored a Variable Interaction Learning (VIL) approach to problem decomposition [21] in an attempt to determine if correctly identifying variable interaction improves performance. VIL uses a bottom-up approach to finding variable interaction, merging interacting variables into groups based on random permutations of the variables. They concluded that when the problem decomposition identified at least 10% of the total number of interactions, CCEA benefits from the decomposition strategy.

Omidvar *et al.* [108] also showed that finding an underlying structure of interaction to

create the factors improves CCEA's performance when compared to random grouping. In order to decompose variables automatically into groups for CCEA, Omidvar *et al.* introduced Differential Grouping (DG) [107]. DG is based on a process of identifying partial separability, allowing variables that are directly interacting to belong to the same group, while minimizing interdependence between groups (Algorithm 4.1). Interaction is determined by measuring how much the function changes between pairs of points. Δ_1 is measured by changing the value of variable x_i and evaluating the function at both points. Then, variable x_j is altered to a different value, where $j \neq i$, and compute Δ_2 is computed by re-evaluating the function at the points from Δ_1. If $|\Delta_1 - \Delta_2| > \epsilon$, the variables are said to interact, and the corresponding variable x_j is removed from the set of remaining decision variables and added to the group of interacting variables for x_i. The ϵ parameter plays an important role in determining interactions: a smaller ϵ will detect weaker interactions. The authors prove that x_i and x_j are not independent if the difference between Δ_1 and Δ_2 is large enough, based on the specified parameter ϵ.

4.2 Related Work and Motivation

Several extensions have been proposed to the DG algorithm. In 2017, Omidvar, *et al.* [109] extended DG to set the ϵ threshold automatically, creating DG2. This extended version also reduced the number of fitness evaluations necessary to find the groups of interacting variables. A second approach to adjust the ϵ threshold parameter automatically was introduced by Sun *et al.* in recursive DG [141]. Recursive DG looked at a pair of sets of variables and divided each set recursively to increase efficiency. Yang *et al.* [167] exploited gathered information about the interaction between variable groups to create efficient recursive DG. DG has also been adapted to become a graph-based approach to better suit LSO problems by Ling *et al.* [92]. They constructed a graph where each decision variable is a vertex, and weighted edges are defined based on pairwise interaction calculated

Algorithm 4.1 Differential Grouping

Input: function f, lower bounds lb, upper bounds ub, number of dimensions n, threshold ϵ
Initialize: Variable indeces $dims \leftarrow \{1, 2, \ldots, n\}$, set of variables with no interactions $sepvars \leftarrow \{\}$, groups of interacting variables $groups_o \leftarrow \{\}$

1: **for all** $i \in dims$ **do**
2: $grp \leftarrow \{i\}$
3: **for all** $j \in dims, j \neq i$ **do**
4: $p_1 \leftarrow lb \times ones(1, n)$
5: $p_2 \leftarrow p_1$
6: $p_2(i) \leftarrow ub$
7: $\Delta_1 \leftarrow f(p_1) - f(p_2)$
8: $p_1(j) \leftarrow 0$
9: $p_2(j) \leftarrow 0$
10: $\Delta_2 \leftarrow f(p_1) - f(p_2)$
11: **if** $|\Delta_1 - \Delta_2| > \epsilon$ **then**
12: $grp \leftarrow grp \cup j$
13: $dims \setminus j$ // Removes seen variables from variable indeces.
14: **if** $|grp| = 1$ **then**
15: $sepvars \leftarrow sepvars \cup grp$
16: **else**
17: $groups_o \leftarrow groups_o \cup \{grp\}$
return $groups_o \cup \{sepvars\}$

using DG. After graph construction, connected components are found using a depth-first search algorithm. The authors found that their graph-based approach improved solution quality for CCEA's on LSO.

Another commonly used method for identifying variable interaction is called Maximum Entropic Epistasis (MEE) [140]. The MEE approach identified direct and indirect variable interaction based on Mutual Information (MI) [40]. Sun *et al.* compared MEE and DG in their abilities to determine variable interaction. They concluded that MEE finds a more accurate decomposition. However, MEE was not applied to decompose a problem into subgroups, nor was it applied to decomposition for CCEA. Rather, their approach was used with the Separability Prototype for Automatic Memes (SPAM) framework [18], which used

variable interaction to select the appropriate operators to guide the search. The original SPAM framework and the adjusted version using MEE (SPAM-MEE) were compared to covariance matrix adaptation—evolutionary strategy (CMA-ES) [60]. The results showed that SPAM-MEE outperforms SPAM consistently and performed similarly to CMA-ES.

Chen *et al.* looked at the influence of variable interaction groups on CCEA for single-objective optimization [20]. Their study created variable groups through variable interaction and used three different versions of the found decomposition: groups representing the found interaction structure, groups only keeping part of the interaction structure, or random groups without variable interaction. The results indicated that learning the correct variable interaction structure improved results; however, the results generally improved more if the problem decomposition only partially adhered to the found interaction structure. Given that an exact match was found to be unnecessary, we believe an overlapping structure could have similar benefits, since the overlap would account for unidentified interactions between groups. In other words, using an overlapping structure that connects all variables may mitigate the need for variable interaction learning.

4.3 Decomposition Methods

We present our new decomposition approaches for FEA architectures: Overlapping Differential Grouping (ODG) and a tree based decomposition (Tree), where both methods can produce overlapping variable decompositions. ODG is an extension of DG [108] that continues to consider variables that have been marked as interacting instead of removing them (as in DG) allowing for overlapping groups. Tree based decomposition considers variables as nodes in a tree, and creates factors based on adjacent nodes in the tree.

4.3.1 Overlapping Differential Grouping

All of the versions of DG discussed in Section 4.2 are intended to identify disjoint factors in large-scale optimization. Here we present an alternative approach that adjusts DG to allow factors to overlap: Overlapping Differential Grouping (ODG). To find overlapping factors, instead of removing a variable once it has been found to interact, that variable remains in the set of decision variables. This allows a variable to be marked as interacting with multiple other variables: which leads to overlapping factors. This means that the interaction step of DG will be performed on all decision variables; however, for each such variable, only subsequent variables are compared, i.e., we add the condition $j > i$ to line 3 in Algorithm 4.1. This is because interactions with all prior variables have already been considered, and do not need to be considered again. Additionally, ODG removes line 13 from Algorithm 4.1 since we wish to evaluate interaction for each variable pair to create overlapping groups. Specifically, the pseudocode line corresponds to

$$dims \setminus j$$

which removes members of that group from future consideration.

4.3.2 Tree Based Grouping

In the original FEA work by Strasser, factor graphs were constructed to connect variables through a function called the "factor potential" [139]. The factor potential defines the strength of relationship between states for a variable. Variable groups are then created based on the connection of the factor potential to the related variables. In the continuous case, we could think of the factor potential as the strength of the variable interactions, connecting variables if they have a strong enough interaction. However, we believe that calculating variable interaction to build interaction graphs may not be necessary as long as

all variables are connected to each other either directly or indirectly, and overlap of variables allows for such an indirect connection of variables. Inspired by this idea, we created tree-based grouping (Tree).

We consider each variable as a vertex in a graph, and connect them in a tree T. Factors are then constructed to be adjacent nodes in the tree. For purposes of our experiments, we use a random tree, but we note that any tree could be used. We believe that, as long as the factor architecture is connected, the origin of the underlying tree (whether random or interaction-based) is not important. The goal is to enable communication between the factors so that the interacting effects propagate through the tree structure during optimization.

The algorithm is described more formally in Algorithm 4.2. This algorithm iterates over each node in the tree and creates a factor consisting of the variable i and the variables connected to it in the tree $T.neighbors(i)$. This method constructs a simple tree-based architecture that has a connected factor architecture. In a connected factor architecture, any factor must be able to connect to any other factor through a sequence of overlapping factors. More specifically, we can envision a factor architecture to consist of a graph where each vertex in the graph corresponds to a factor, and an edge is created between two factors whenever the intersection of the variable sets in these factors is non-empty. The resulting factor decomposition is said to be connected if the resulting graph is connected. An illustration of such a graph and the corresponding factors is shown in Figure 4.1 We note that tree-based grouping results in a connected factor decomposition where this property does not necessarily hold for ODG. We would also like to point out that there exists a Tree for which the resulting architecture would have a group that contains all variables; namely, if there is a "central" node that connects to all other nodes directly. If this were to occur we could 1) remove the variable group containing all variables, leaving groups of variable pairs where each pair contains the central node, or 2) generate a different Tree to create new groups.

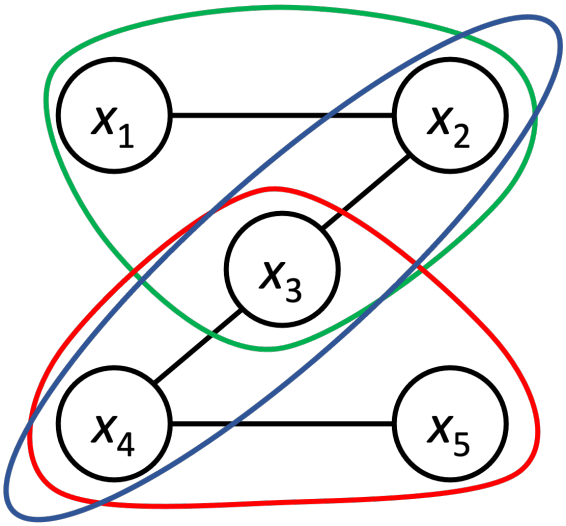


Figure 4.1: Sample tree decomposition for function F20 with five variables.

Algorithm 4.2 Tree Based Grouping

Input: tree T, number of dimensions n
Initialize: Variable indeces $dims \leftarrow \{1, 2, \ldots, n\}$, variable groups $groups_o \leftarrow \{\}$

1: **for all** $i \in dims$ **do**
2: $\quad grp(i) \leftarrow \{i\} \cup \{T.neighbors(i)\}$
3: $\quad groups_o \leftarrow groups_o \cup \{grp(i)\}$
return $groups_o$

4.4 Experimental Approach

The purpose of this research is to compare the influence of the factor decomposition on large-scale optimization. Thus, we hypothesize that the overlapping factor decomposition will provide significant benefit to optimization quality specifically on problems with a non-separable component. This is because the overlap inherently takes into account and manages variable interactions during the compete and share steps. We further expect that these interactions can be handled through sufficient iterations of the FEA algorithm to produce good results. This belief is motivated by prior work in a distributed setting relating the optimization process to a distributed consensus problem and showing that relaxed consensus

objectives can still lead to effective performance [14].

For our experimental set-up, we followed the guidelines in [143], performing 25 trials on 1000 dimensions. We held the number of function evaluations for most of our experiments to 3×10^6. Several different algorithms were compared: CCEA, FEA, and PSO; furthermore, we used PSO as the base algorithm for both FEA and CCEA. For CCEA decomposition, we used DG [108] and the original n 1-dimensional subproblem decomposition for PSO – CPSO-S [150]. To decompose FEA, we compared the proposed DG-extension (ODG) as well as the tree-based decomposition (Tree) using a random tree (Algorithm 4.2). In addition, we used another tree-based method, Tree2, where we merged the smallest pairs of factors from the Tree method iteratively until the total number of factors is 500. This helped determine the effect of factor decomposition and number of factors. Since we used PSO as the base algorithm for both FEA and CCEA methods, we also compared our results to single-population PSO.

For the canonical PSO experiments we used a population size of 1000 and used 3000 iterations in order to generate the same number of function evaluations. For PSO hyperparameters we set $c_1 = c_2 = 1.49445$ and the inertia weight $\omega = 0.729$, following work in [178]. We also decided to use the *gBest* topology.

Five CEC' 2010 LSO benchmark functions [143] were used to test our methods, since these were used in early experiments with DG [107, 108]. There are 20 functions split into five different categories: 1) separable, 2) single-group shifted and m-rotated, 3) $D/2m$ group shifted and m-rotated, 4) D/m group shifted and m-rotated, and 5) non-separable functions. D corresponds to the number of dimensions, and m determines how many variables are in a single group. These benchmark functions are all scalable, meaning the number of dimensions can be chosen. Fully separable and non-separable functions are defined as having one-dimensional non-interacting sub-vectors and having every pair of decision variables interact respectively. Li, *et al.* [89] define partially additively separable functions as having the

following general form, where $\mathbf{x}_i$ are mutually exclusive decision vectors of f_i and m is the number of independent factors:

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}_i) \ .$$

The parameter m determines the rotation matrix to adjust the degree of separability for the partially non-separable functions; and "shifted" refers to the use of a shift vector to shift the global optimum. For our experiments, we ran the algorithms on one benchmark function from each of the five categories for 1000 dimensions and set $m = 50$; each function has a global minimum value of 0. The specific functions studied are:

1. F3: Separable—Shifted Ackley
2. F5: Single-group m-nonseparable—Shifted m-rotated non-separable Rastrigin
3. F11: $D/2m$-group m-nonseparable—Shifted m-rotated Ackley
4. F17: D/m-group m-nonseparable—Shifted m-dimensional Schwefel's 1.2
5. F20: Fully nonseparable—Shifted Rosenbrock.

The definitions of these five functions can be found in Appendix B.

To examine the effect of the factor architecture in more detail, we created a manual grouping for function F20, because F20 comes from the fully non-separable group of functions. We note from the problem definition of fully non-separable problems that each variable i interacts with $i-1$ and $i+1$. So we manually crafted a factor architecture covering these interactions in order to examine the benefit of a factor architecture that matches the interactions within the problem.

For consistency, hyperparameters were held constant across functions and methods. We set 15 PSO iterations, and in earlier experiments, we did not see an influence of performance on PSO population size, so we set it to the smallest value: 10 particles per subswarm for all

Function		DG	ODG	Tree	Tree2
F3	Number of Factors	1000	1000	1000	500
	Average Factor Size	1	1	2.99	5.99
F5	Number of Factors	951	1000	1000	500
	Average Factor Size	1.05	2.23	2.99	5.99
F11	Number of Factors	513	1000	1000	500
	Average Factor Size	1.95	13.21	2.99	5.98
F17	Number of Factors	40	1000	1000	500
	Average Factor Size	25	24.25	2.99	5.99
F20	Number of Factors	266	1000	1000	500
	Average Factor Size	3.76	43.42	2.99	5.99

Table 4.1: Summary of groupings made by each algorithm.

FEA and CCEA trials. In DG and ODG, we set ϵ to 10^{-3} for consistency across the different methods and functions. We did not tune them individually because we sought to test the influence of the overlap rather than find the best performing configuration.

4.5 Results

Because of the generality of the FEA framework, we applied the framework to all of the architectures. We present metrics characterizing the different factor architectures generated for our experiments in Table 4.1. It is interesting to note that as the non-separability of the function increases, the factor size of ODG also increases. This is because ODG detects the variable interactions and groups them together; the more variable interactions, the larger the interaction groups.

We ran experiments as outlined in Section 4.4. Table 4.2 shows the mean value of the objective function found during optimization, averaged over 25 trials, and the standard

Function		DG	CPSO-S	ODG	Tree	Tree2	PSO	Manual
F3	Mean	**5.79E-05**	**5.93E-05**	**5.92E-05**	1.37E+00	1.79E+01	2.16E+01	
	std	8.23E-06	9.08E-06	9.52E-06	4.24E-01	4.91E+00	8.43E-03	
F5	Mean	8.05E+09	**5.95E+08**	1.22E+09	**5.53E+08**	**5.56E+08**	3.63E+10	
	std	2.58E+09	1.30E+08	4.15E+08	1.12E+08	1.27E+08	2.31E+09	
F11	Mean	2.08E+02	**2.00E+02**	2.04E+02	2.20E+02	2.22E+02	2.37E+02	
	std	4.52E-01	1.07E-02	3.67E-01	1.46E+00	2.48E-01	5.86E-02	
F17	Mean	1.77E+06	2.58E+06	1.57E+06	1.75E+06	**1.37E+06**	3.29E+07	
	std	1.07E+05	1.74E+05	7.79E+04	1.42E+05	1.04E+05	1.91E+06	
F20	Mean	6.10E+09	**7.36E+01**	4.58E+06	1.18E+04	1.41E+06	6.82E+12	4.60E+03
	std	4.74E+09	2.31E+01	7.81E+06	2.62E+03	6.91E+05	1.71E+11	2.17E+03

Table 4.2: Comparison of different optimization methods on CEC 2010 benchmark functions. Bold values indicate best results that were significantly better (Wilcoxon Rank-Sum p-value < 0.05)

deviation of the returned values for each of the functions. Entries highlighted in bold are statistically significantly better than the other algorithms, tested using Wilcoxon's Rank-Sum Test at a confidence level of 95%. Note that when multiple entries are bolded for a function, the corresponding methods were found to be statistically the same as one another. For example, for the fully separable function F3, the three methods—DG, CPSO-S, ODG—were found to be statistically equivalent but significantly better than Tree, Tree2, and PSO.

Observing the results in Table 4.2, we see that the overlapping methods typically outperform DG on the non-separable functions. This indicates that the overlapping decomposition provides significant benefit. They typically also have lower standard deviations, indicating more consistent and stable performance when compared to DG. On every function with a non-separable component, FEA methods perform competitively.

We plotted the convergence curves for the first trial of each method. These results can

be seen in Figure 4.2. Based on these results, we make a number of interesting observations. First, we see rapid convergence for the three best algorithms on F3, which makes sense given the fact F3 is fully separable. Thus we would not expect overlap to provide any benefit. However, as soon as variable interaction is introduced (as shown in the remaining four plots), DG suffers. We found that the type of factor architecture does, in fact, have an effect on performance, given the fact the "best" architecture varies across the functions studied.

Finally, we see that, on functions F17 and F20, none of the methods converge well within the limited number of function evaluations. Based on this, we ran a second set of experiments and allowed the functions to run for double the number of function evaluations (6×10^6). We ran 10 trials of each method and present the results in Table 4.3. If we also consider Figure 4.3, we see that most of the algorithms seem to have converged on F20, but it appears that they have not yet converged on F17, so there could be more benefit to continuing evaluation. The results improved significantly for all except DG on F17, and Single and Tree performance improved significantly on F20; however, the relative performance of each method did not change on F20. Specifically, if we rank the performance after 6 million fitness evaluations, we see that the rank is the same as when we stop at threemillion fitness evaluations. For the F20 trials using the manual grouping we used the standard 3×10^6 function evaluations, and we found an average over 25 trials of $4.597E+03$. This improved upon the next best score produced by FEA by over a factor of two. It is worth noting that CPSO-S performed best on F20. In particular, CPSO-S performed over two orders of magnitude better than the next best algorithm (Tree). Potential reasons are discussed in the next section.

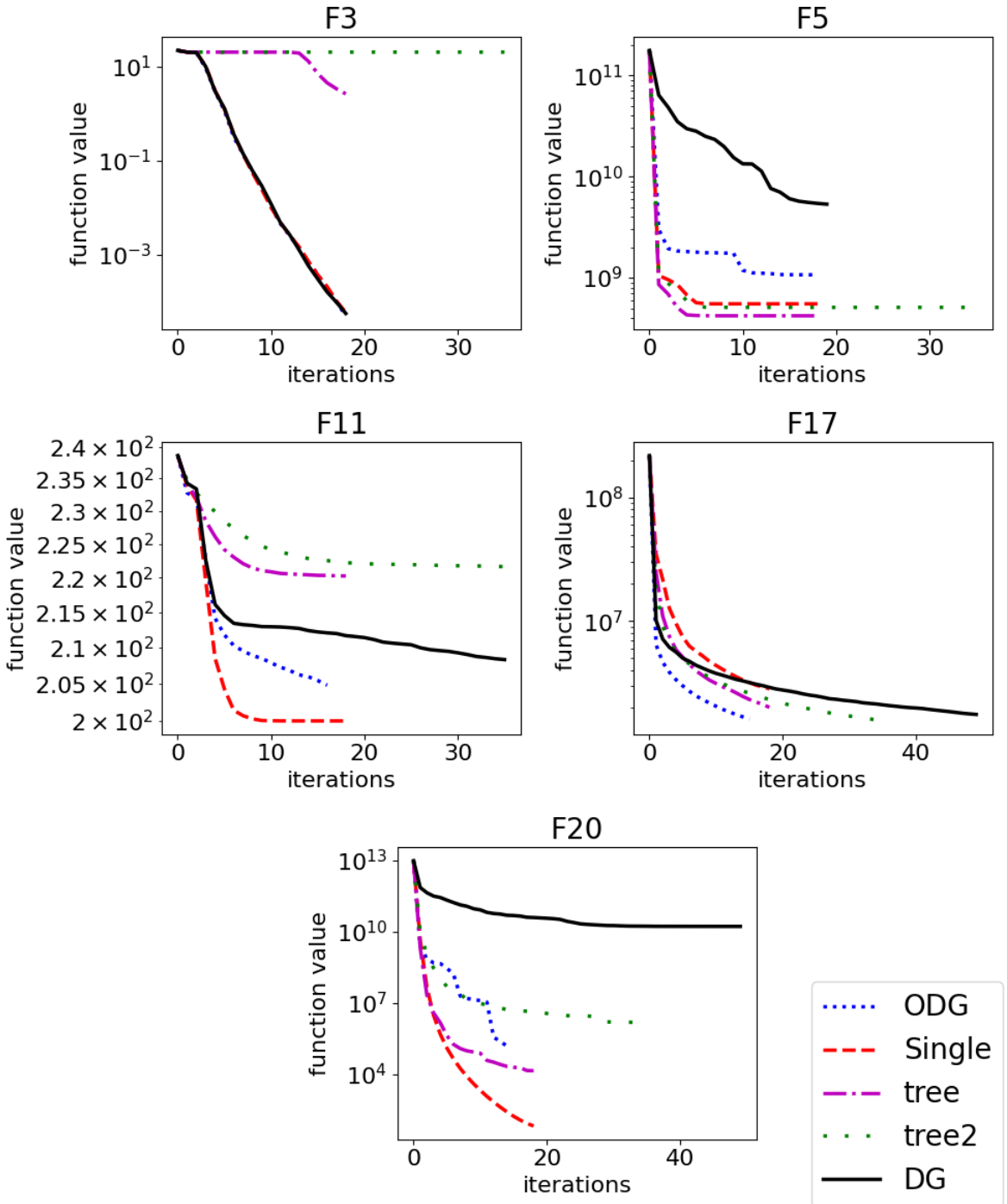


Figure 4.2: Convergence plots for first trial of each method. Learning terminates when max function evaluations (3×10^6) are reached

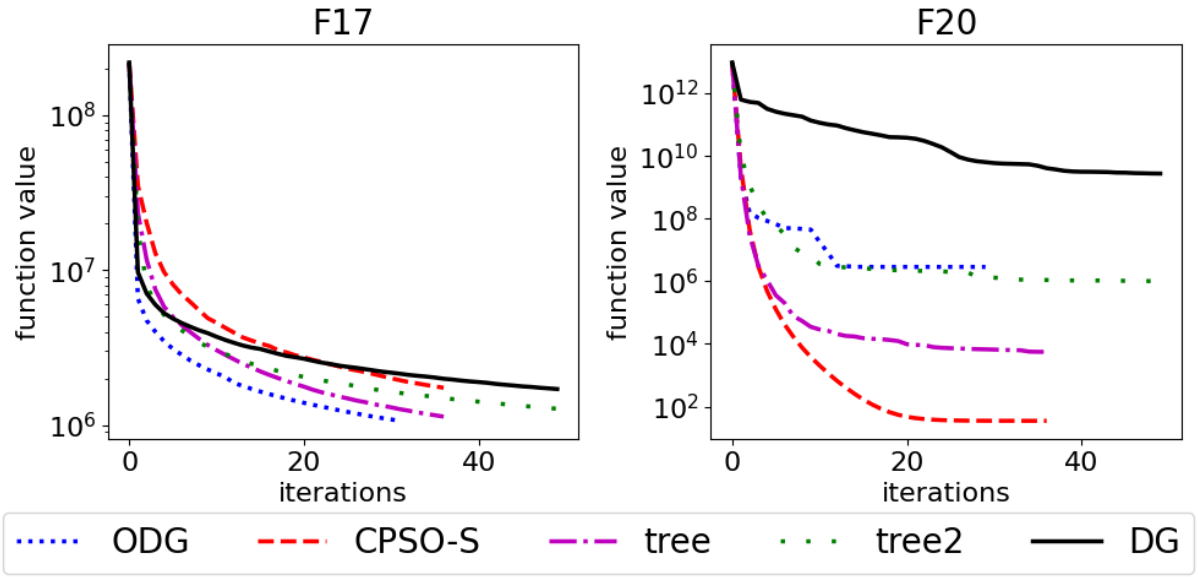


Figure 4.3: Convergence plots for F17 and F20 with 6×10^6 function evaluations

Function		DG	CPSO-S	ODG	Tree	Tree2	Manual
F17	Mean	1.68E+06	1.56E+06	**9.82E+05**	**1.02E+06**	**1.07E+06**	
	std	4.28E+04	1.37E+05	6.42E+04	9.30E+04	1.12E+05	
F20	Mean	4.25E+09	**6.24E+01**	4.26E+06	6.21E+03	9.94E+05	4.43E+03
	std	2.83E+09	7.45E+01	8.29E+06	8.12E+02	4.35E+05	1.38E+03

Table 4.3: Comparison of different optimization methods on F17 and F20 with double the number of function evaluations. Bold values indicate best results that were significantly better (Wilcoxon Rank-Sum p-value < 0.05)

4.6 Discussion

Our results confirm that proper problem decomposition is important for large scale optimization. Canonical PSO performed the worst on every function, typically by several orders of magnitude. This indicates that problem decomposition improves optimization performance on these large-scale optimization problems. Even so, the basic decomposition

performed by DG was not shown to be particularly effective on several of the functions studied.

In general, the introduction of overlapping factors in the problem decomposition helped with optimization on non-separable problems. The poor performance of the Tree algorithms on F3, which was completely separable, was expected since these methods always produce overlapping factors. Since F3 is fully separable, the overlap does not provide any benefit, and instead increases the difficultly of solving the sub-problems by increasing their factors' dimensionality. We also note that ODG performed very well on this function in that it was able to recognize the fully separable nature, thus reducing to DG.

On the other hand, the benefit of the overlap is prominent in F20. The FEA methods show improvement by many orders of magnitude over that of CCEA using DG. F20 is a fully non-separable problem indicating that overlap may be beneficial. If we look at F17, we also see a similar result where the overlapping methods perform significantly better than the non-overlapping methods, further supporting the benefit of overlap on non-separable problems.

It is worth noting that, as expected, the factor architecture seems to be important at determining optimization success. This observation is prominent in the improvement on F20. By addressing the underlying variable interactions present in the function with a manually defined factor architecture, the results were improved by a factor of two. Despite the improvements shown by choosing the correct factor architecture, it is interesting that Tree and Tree2 both produce architectures that lead to high quality results despite not addressing any underlying properties of the function. That implies that having overlapping factors is a great benefit to optimization regardless of architecture, but improvement can be extended further with the "right" factor architecture.

We also note that using variable interaction as the basis for problem decomposition may not be beneficial in creating an appropriate factor architecture. ODG considers these

interactions and is capable of determining separability, but it does not always outperform the random architectures Tree and Tree2. It is possible that ODG does not capture these interactions fully, or there may be other properties of the problem decomposition, such as whether all variables are connected through overlap, that play a larger role.

A consistent and unexpected result is the performance of CPSO-S on all functions. CPSO-S was the simplest version of CCEA with PSO and was not expected to perform well on non-separable problems because it does not consider any form of variable interaction. However, CPSO-S was competitive across all functions. In particular, it achieved the lowest scores on F11 and F20. We believe this could be explained by considering the hitchhiking phenomenon, which occurs when a solution on a whole improves the fitness score, but single variables' values are deteriorating in the process [15]. Using single variable groups could bypass this problem since each variable is now optimized separately. Additionally, the FEA implementation of CPSO only changes the global solution if the value improves the fitness score, leading to a Pareto improved solution. Note that this result can also be a byproduct of the challenges associated with using fitness evaluations as the means for making results comparable, as pointed out by Engelbrecht [44]. In particular, across the various factor architectures, not all fitness evaluations are created equal.

4.7 Concluding Remarks

Overall, we found that applying decomposition strategies is beneficial for many different problems in different areas of large-scale and multi-objective optimization, where different strategies appear to be better suited for certain problems. For LSO, we extended Differential Grouping to create overlapping groups and created a tree based decomposition approach. The different PSO implementations were tested using five representative functions from the set of CEC'2010 benchmark functions using the proposed guidelines [143]. Results showed that overlap can be beneficial for optimization of non-separable problems, or problems with a

non-separable component. Furthermore, the LSO results indicated that variable interaction learning may not be necessary when creating a connected architecture. Having confirmed the benefit of overlapping groups on LSO, we now move on to exploring the effects of variable grouping on MOO.

CHAPTER FIVE

INFLUENCE OF VARIABLE GROUPING ON MULTI-OBJECTIVE OPTIMIZATION

In the previous chapter we discovered that finding variable interaction may not be necessary as long as a connected factor architecture is created through overlap. In this chapter we further investigate the influence of variable grouping by looking at how overlap influences multi- and many-objective optimization. For this research, we looked at three different, commonly used MOEA's to gain insight into the MOFEA framework: the Non-Dominated Sorting Genetic Algorithm II (NSGA2) [36], the Strength Pareto Evolutionary Algorithm (SPEA2) [185], and the Multi-Objective Evolutionary Algorithm with Decomposition (MOEA/D) [175]. We also used three different grouping strategies to create disjoint and overlapping groups: linear [149], random [170], and differential grouping [108].

5.1 Related Work and Motivation

The related work section of Chapter 3 presented research using the idea of subpopulations in MOO. In this section, we focus on research that investigates the influence of variable decomposition on MOO more directly through different variable interaction strategies.

Two separate surveys on LSMOO were published in 2021 [63, 146]. These surveys categorized three different ways to solve LSMOO problems: decomposition, reduction, and search strategy adaptation. In decomposition-based approaches, there are two common ways to decompose a problem: by creating different variable subpopulations in the manner of Co-operative Co-Evolutionary Algorithms (CCEA) [17, 117] or through decomposition of the problem itself [98, 175]. The latter form often uses Decision Variable Analysis (DVA) to help create the appropriate weight vectors for decomposition [96, 177]. According to

Tian *et al.*, the second approach, which aims to reduce the variable space through problem transformation or dimensionality reduction, resulted in a higher likelihood of getting stuck in local optima [146]. The third category looks at adapting the search operators of MOEA's; this is a useful practice but need not be kept separate from CCEA's. When decomposing the variable space into subgroups to be optimized separately, we can apply any MOEA to the subpopulation to perform optimization. Therefore, reference-direction-based or genetic-operator-adjusting approaches can still be applied when creating subpopulations for CCEA.

Similar to FEA, the idea of combining competitive and cooperative methods was used by Goh and Tan [54] who combined the power of the two types of co-evolutionary algorithms to create a new competitive-cooperative co-evolutionary algorithm (COEA). In their algorithm, each subpopulation competed to represent a specific factor. The resulting factors then cooperated to find a better overall solution. This particular process enabled the algorithm to find interdependencies among the different subpopulations, where similar subpopulations represent similar factors. The competitive pressure also enabled a structure to emerge, which obviated the need for an algorithm such as DG. COEA was used for multi-objective optimization (MOO), and more specifically dynamic MOO. Their results showed that while their approach did improve optimization for dynamic environments, there were no significant improvements with static MOO.

Xu *et al.* also looked at improving dynamic MOO through co-evolutionary algorithms [163]. In their proposed CCEA adaptation, decision variables were partitioned into two subcomponents according to environmental sensitivities of each variable. In other words, since not all variables are equally affected by changes in the environment, the authors proposed grouping variables based on their level of influence: weak or strong. If a variable was strongly influenced by the environment, more resources were assigned to optimizing those variables. The results showed that the proposed environmental decomposition is beneficial for dynamic MOO problems.

Cao *et al.* adapted graph-based DG (gDG) [91] to be applied to multi-objective LSO problems to create Multi-Objective gDG (MOgDG)[17]. MOgDG first performed what the authors called property analysis, which identified whether a variable is diversity- or convergence-related based on the effect it has on the solution. DG was then used to construct a correlation matrix, which is called the variable interaction stage. Lastly, a graph was created and variables in the same connected subcomponent, determined through breadth-first or depth-first search, were put in the same group. The authors found that using their graph-based decomposition in combination with the CCEA version of NSGA2 and MOEA/D improved results on the DTLZ and WFG benchmark functions, where CC-MOEA/D had the overall best performance.

Zheng *et al.* also used a graph-based decomposition strategy to address the problem of water distribution network (WDN) design [179]. Their approach was problem specific: WDN's commonly have trees, blocks, and bridges in their network that correlate directly to specific parts of the WDN. Based on these naturally occurring structures, the network was decomposed into subnetworks which were then optimized separately using a multi-objective DE approach. They found that their approach significantly increased computational efficiency for WDN.

The above works indicate that, thus far, research on creating variable groupings for LSMOO has focused on novel strategies applied to specific algorithms. To the best of our knowledge, no direct comparison of different variable grouping approaches has been performed. We aim to address this knowledge gap through analysis of different decomposition strategies. Furthermore, most research using variable grouping focuses on two- and three-objective problems, whereas we scale up to five and ten objectives.

Lastly, in previous work by Li *et al.*, DTLZ was found to have different variable interactions along the different objectives [85]. These different variable interaction groups create overlap in the variable space; however, no work has been performed to take this overlap

into account directly when solving these benchmark functions. We aim to address this issue using the MOFEA framework.

5.2 Decomposition Methods

For this analysis, we compared the three most commonly used approaches: linear grouping, random grouping, and differential grouping, each with distinct and overlapping structures. In this section, we explain the specific grouping implementations and the adjustments to create overlapping groups.

5.2.1 Linear and Random Grouping

Linear grouping divides variables into groups based on their position in the variable space [149]. Say we have 100 variables ($x_0 \dots x_{99}$) and we want each variable group g_i to consist of ten variables: $g_0 = \{x_0 \dots x_9\}$, $g_1 = \{x_{10} \dots x_{19}\}$, ..., assuming some "cannonical" ordering of the variables. To extend this to the case of overlapping groups, a parameter offset size is provided. For example, if the offset equals 5, the groups become: $g_0 = \{x_0 \dots x_9\}$, $g_1 = \{x_5 \dots x_{14}\}$, $g_2 = \{x_{10} \dots x_{19}\}$, etc.

When creating random groupings, a pre-defined number of variable groups is created, and variables are spread evenly across the groups through random selection [170]. We extend the random grouping approach to create overlapping groups by iterating over the groups in the order they were created and adding variables from consecutive groups to a new group. Suppose we have the same parameters as above, but random grouping created the following groups: $g_0 = \{\mathbf{x_0}, \mathbf{x_3}, x_{12}, \mathbf{x_{54}}, \mathbf{x_{66}}, x_{70}, x_{71}, x_{79}, \mathbf{x_{92}}, x_{93}\}$, $g_1 = \{x_5, \underline{x_{22}}, x_{31}, x_{33}, x_{40}, x_{45}, \underline{x_{73}, x_{81}, x_{85}, x_{88}}\}$, ... The overlapping variables will be chosen randomly from consecutive groups to form a new group. If we set the overlap size to be the same as the group size, this would result in five variables being chosen from g_0 and five from g_1. For example, $g_{10} = \{\mathbf{x_0}, \mathbf{x_3}, \underline{x_{22}}, \mathbf{x_{54}}, \mathbf{x_{66}}, \underline{x_{73}, x_{81}, x_{85}, x_{88}}, \mathbf{x_{92}}\}$. The next overlapping

group would then be created by choosing variables from groups g_1 and g_2, and so on. This results in a connected architecture as defined previously.

5.2.2 Differential Grouping

In our experiments, we applied DG to create both distinct and overlapping groups for MOO. To create variable groups through differential grouping, we used two different methods:

1. the MOgDG approach as proposed by Cao *et al.* [17],

2. an approach where we collapsed the groups created along each objective, i.e., we applied differential grouping to each objective separately and then combined the resulting groups (Figure 5.1).

The collapsing groups approach works as follows. Figure 5.1 shows a problem with three objectives and ten variables. In this example, we randomly chose variables to belong to interaction groups to mimic applying DG along different objectives. The resulting groups are indicated using different shades of grey. For Obj 1 this resulted in the following groups: $\{\{x_1, x_2, x_5, x_9\}, \{x_3, x_7, x_8\}, \{x_4, x_6, x_{10}\}\}$. The solid lines show the group boundaries for the relevant objective, and the dotted lines indicate the group boundaries for the other objectives projected onto the relevant objective. For example, in Figure 5.1, objectives one and two group variables x_1 and x_2 in the same group; however, for the third objective, x_1 and x_2 are in different groups. Based on the third objective's variable groups, we "extended" the boundary to the other two objectives to create collapsed groups, this now splits up variables x_1 and x_2 for the first two objectives as well. For the overlapping groups, we applied DG along each objective separately and kept the variable groups created along the objectives.

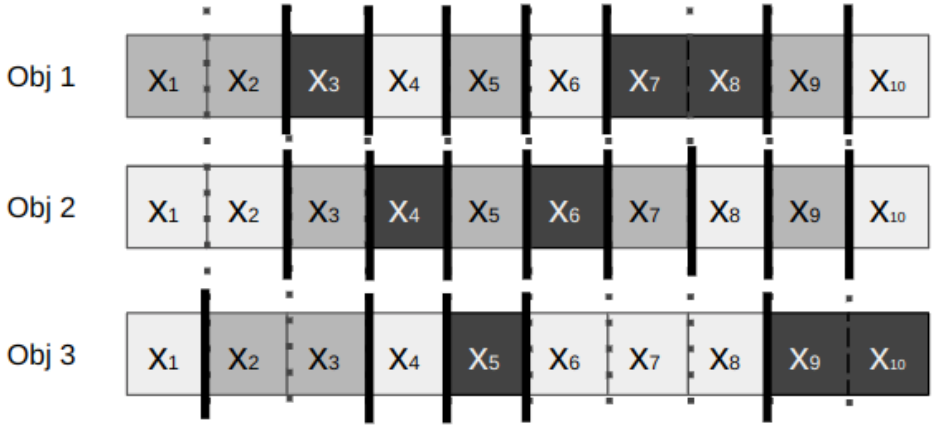


Figure 5.1: Example collapsing groups after applying differential grouping along three objectives for ten variables.

5.3 Experimental Approach

We applied NSGA2, SPEA2, and MOEA/D with different factor architectures to the DTLZ benchmark suite with M objectives. This test-suite was chosen because of its ability to scale both the number of variables and the number of objectives to the desired sizes, as well as their availability in the pymoo library, ensuring correct implementation of the functions [6]. Table 5.1 shows the four characteristics of the DTLZ functions as defined in [65]. Modality is Unimodal (U) or Multimodal (M). Separability is separable (S) or unknown (?). A function is biased (Y) when there is a large discrepancy between distribution of solutions in the variable and objective spaces. Geometry is linear, concave, unknown (?) or disconnected. In our experiments we set the number of objectives (M) to equal three, five, and ten for each of the DTLZ problems. We evaluated the results using three metrics: hypervolume (HV), spread (S), and adjusted coverage (AC). For each algorithm, the average HV and S were calculated across ten runs for each problem. The Wilcoxon rank-sum test with $\alpha = 5\%$ was performed to assess statistical significance. The HV and S results are presented for all

	Modality	**Separability**	**Bias**	**Geometry**
D1	M	S	N	linear
D2	U	S	N	concave
D3	M	S	N	concave
D4	U	S	Y	concave
D5	U	?	N	?
D6	U	?	Y	?
D7	U&M	?	N	disconnected

Table 5.1: Characteristics of the DTLZ benchmark suite [65].

grouping strategies per algorithm, and are discussed next.

5.4 Results

For clarity, the coverage calculations and results are presented separately for each of the different decompositions: single population, disjoint, and overlapping decompositions. In our experiments, we selected a single run at random out of the set of runs for each algorithm to combine into the union front $\mathbf{X}^*$ and calculate AC, as explain in Chapter 3. We repeated this step $k = 10$ times and averaged the results to get the final AC for each algorithm on a single problem. In each of these sections, we also explain the experimental set-up in more detail for the different algorithm implementations.

5.4.1 Single Population Experiments

We compared the three different single-population multi-objective evolutionary algorithms: MOEA/D, SPEA2, and NSGA2. The purpose of these experiments is to provide a baseline for comparison with the CCEA-based methods. We ran each algorithm with a

population of 500 individuals for 200 generations, which we approximate as $100,000$ fitness evaluations per algorithm run, and we repeated this 10 times. For each of the algorithms, the choice of the hyperparameters and mutation and crossover strategies were based on the original literature [36, 175, 185]. Since the main focus of the research is the influence of variable grouping on the algorithms, we are able to evaluate this influence regardless of the used hyperparameters. However, this means we cannot draw conclusions regarding specific algorithm performance; we can only make claims on the influence of variable grouping. With regards to MOEA/D, the authors note that the decomposition strategy may impact the results significantly. To this end, we considered both Chebyshev decomposition and PBI decomposition with penalty factor $\theta = 5$ as suggested by the authors (as explained in Chapter 2 Section 2.2.4.3) [175]. It is important to note the hyperparameter settings could introduce bias into the results, since not all algorithms were tuned for these benchmarks. For example, in the original MOEA/D paper, the algorithm was only applied to DTLZ1 and DTLZ2, and in the original SPEA2 paper, the algorithm was not applied to the DTLZ benchmark suite, only to the ZDT benchmarks [184]. Thus we acknowledge that we cannot draw general performance-based conclusions from these experiments with untuned hyperparameters.

AC results of the single-population experiments are shown in Table 5.2. Note that NSGA2 consistently contributed the largest percentage of non-dominated solutions to the union front on the different benchmark functions. There was one exception: three-objective DTLZ7, where MOEA/D with PBI had the largest percentage of non-dominated solutions. Interestingly, this was also the only case in which using PBI decomposition had a clear improvement in terms of *AC*. This is in contrast to the idea that more sophisticated scalarization should improve results [175]; however, we only used a single penalty value of $\theta = 5$, which is likely to impact results. Based on these results, we opted to use the Chebyshev decomposition in subsequent experiments for all problems except DTLZ7. A second notable result is that SPEA2 always contributed the fewest non-dominated solutions,

Problem	M	MOEA/D		SPEA2	NSGA2
		PBI	CH		
DTLZ1	3	0.20±0.010	0.19±0.011	0.11±0.010	0.50±0.007
	5	0.21±0.003	0.20±0.010	0.16±0.003	0.43±0.010
	10	0.21±0.003	0.21±0.003	0.16±0.003	0.43±0.010
DTLZ2	3	0.10±0.034	0.20±0.104	0.17±0.094	0.53±0.124
	5	0.20±0.024	0.26±0.042	0.09±0.022	0.45±0.051
	10	0.20±0.021	0.22±0.016	0.13±0.014	0.45±0.027
DTLZ3	3	0.07±0.045	0.47±0.344	0.03±0.026	0.43±0.280
	5	0.20±0.009	0.22±0.010	0.15±0.004	0.43±0.015
	10	0.20±0.008	0.20±0.007	0.15±0.006	0.45±0.021
DTLZ4	3	0.01±0.020	0.03±0.041	0.06±0.014	0.90±0.045
	5	0.13±0.041	0.17±0.058	0.00±0.001	0.71±0.040
	10	0.16±0.017	0.17±0.004	0.13±0.004	0.54±0.011
DTLZ5	3	0.08±0.020	0.31±0.060	0.04±0.054	0.57±0.040
	5	0.11±0.011	0.16±0.006	0.11±0.012	0.62±0.008
	10	0.14±0.013	0.15±0.005	0.11±0.005	0.59±0.012
DTLZ6	3	0.22±0.005	0.23±0.007	0.14±0.014	0.41±0.020
	5	0.21±0.062	0.21±0.062	0.15±0.048	0.43±0.172
	10	0.25±0.102	0.25±0.103	0.19±0.079	0.30±0.284
DTLZ7	3	0.41±0.032	0.22±0.034	0.00±0.000	0.37±0.021
	5	0.20±0.009	0.20±0.008	0.11±0.028	0.49±0.018
	10	0.17±0.004	0.17±0.004	0.13±0.003	0.53±0.010

Table 5.2: Average adjusted coverage: single-population.

but its contribution did often improve as the number of objectives increased. This makes sense, since an increase in objectives often means there are more non-dominated solutions to be found. These results confirm a known difficulty of many-objective optimization and emphasizes the importance of performing more research into reducing the number of non-dominated solutions making it into temporary and final solution sets.

5.4.2 Disjoint Variable Grouping Experiments

Next, we examined the effects of disjoint groupings using two different variable grouping strategies: linear and random. We applied these strategies to all three base algorithms.

We based the number of generations for CCMOEA on the group size of the method used. Both linear and random grouping use a static group size; to determine this size we ran preliminary experiments with different group sizes (50, 100, 200, and 250) and found that ten groups of 100 variables had the most promising results. Based on the number of groups, we determined the number of generations to run CCMOEA to be 20, using a population size of 500 for each subpopulation, to approximate the same $100,000$ function evaluations as used for the single population algorithms.

When applying the two different DG methods (MOgDG and collapsing DG) to create disjoint groups, one of two scenarios occurred: 1) all variables ended up in their own group, or 2) all variables were contained in a single group of all variables. This confirmed the results found by Cao *et al.*. We tried applying single-variable grouping optimization, but this resulted in a high computational overhead and large memory usage, making it infeasible to run these experiments in a reasonable timeframe, reducing the approach's usefulness for real-world applications. Therefore, we only present results for linear and random grouping used with co-operative co-evolution since the only usable DG results corresponded to the single-population case (already reported in Section 5.4.1).

We compared the different co-operative co-evolutionary approaches to the single

NSGA2				
Problem	M	Single	Linear	Random
DTLZ1	3	0.00±0.000	0.49±0.016	0.51±0.016
	5	0.00±0.000	0.00±0.000	0.00±0.000
	10	0.00±0.000	0.42±0.054	0.58±0.054
DTLZ2	3	0.86±0.031	0.07±0.016	0.07±0.026
	5	0.69±0.034	0.15±0.016	0.16±0.019
	10	0.60±0.011	0.19±0.010	0.21±0.006
DTLZ3	3	0.00±0.000	0.53±0.019	0.47±0.019
	5	0.00±0.004	0.50±0.014	0.49±0.015
	10	0.51±0.225	0.22±0.094	0.27±0.132
DTLZ4	3	0.99±0.010	0.00±0.002	0.01±0.012
	5	0.94±0.022	0.00±0.000	0.06±0.022
	10	0.89±0.002	0.00±0.000	0.10±0.002
DTLZ5	3	0.93±0.028	0.04±0.017	0.03±0.011
	5	0.14±0.139	0.14±0.156	0.72±0.122
	10	NEM	NEM	NEM
DTLZ6	3	0.00±0.000	0.34±0.019	0.66±0.019
	5	0.00±0.010	0.43±0.096	0.56±0.101
	10	NEM	NEM	NEM
DTLZ7	3	0.00±0.000	0.43±0.014	0.57±0.014
	5	0.00±0.000	0.48±0.023	0.52±0.023
	10	0.00±0.000	0.42±0.012	0.58±0.012

Table 5.3: Average adjusted coverage: CC-NSGA2.

SPEA2				
Problem	M	Single	Linear	Random
DTLZ1	3	0.00±0.000	0.49±0.016	0.51±0.016
	5	0.00±0.000	0.00±0.000	0.00±0.000
	10	0.00±0.000	0.42±0.054	0.58±0.054
DTLZ2	3	0.86±0.031	0.07±0.016	0.07±0.026
	5	0.69±0.034	0.15±0.016	0.16±0.019
	10	0.60±0.011	0.19±0.010	0.21±0.006
DTLZ3	3	0.00±0.000	0.53±0.019	0.47±0.019
	5	0.00±0.004	0.50±0.014	0.49±0.015
	10	0.51±0.225	0.22±0.094	0.27±0.132
DTLZ4	3	0.99±0.010	0.00±0.002	0.01±0.012
	5	0.94±0.022	0.00±0.000	0.06±0.022
	10	0.89±0.002	0.00±0.000	0.10±0.002
DTLZ5	3	0.70±0.097	0.14±0.066	0.17±0.051
	5	0.01±0.004	0.00±0.002	0.99±0.005
	10	NEM	NEM	NEM
DTLZ6	3	0.00±0.000	0.39±0.103	0.61±0.103
	5	0.00±0.000	0.03±0.054	0.97±0.054
	10	NEM	NEM	NEM
DTLZ7	3	0.00±0.000	0.43±0.014	0.57±0.014
	5	0.00±0.000	0.48±0.023	0.52±0.023
	10	0.00±0.000	0.42±0.012	0.58±0.012

Table 5.4: Average adjusted coverage: CC-SPEA.

MOEA/D				
Problem	M	Single	Linear	Random
DTLZ1	3	0.00±0.000	0.49±0.016	0.51±0.016
	5	0.00±0.000	0.00±0.000	0.00±0.000
	10	0.00±0.000	0.42±0.054	0.58±0.054
DTLZ2	3	0.86±0.031	0.07±0.016	0.07±0.026
	5	0.69±0.034	0.15±0.016	0.16±0.019
	10	0.60±0.011	0.19±0.010	0.21±0.006
DTLZ3	3	0.00±0.000	0.53±0.019	0.47±0.019
	5	0.00±0.004	0.50±0.014	0.49±0.015
	10	0.51±0.225	0.22±0.094	0.27±0.132
DTLZ4	3	0.99±0.010	0.00±0.002	0.01±0.012
	5	0.94±0.022	0.00±0.000	0.06±0.022
	10	0.89±0.002	0.00±0.000	0.10±0.002
DTLZ5	3	0.97±0.017	0.02±0.015	0.01±0.006
	5	0.01±0.002	0.18±0.127	0.81±0.125
	10	NEM	NEM	NEM
DTLZ6	3	0.00±0.000	0.39±0.060	0.61±0.060
	5	0.01±0.003	0.33±0.186	0.66±0.183
	10	NEM	NEM	NEM
DTLZ7	3	0.00±0.000	0.43±0.014	0.57±0.014
	5	0.00±0.000	0.48±0.023	0.52±0.023
	10	0.00±0.000	0.42±0.012	0.58±0.012

Table 5.5: Average adjusted coverage: CC-MOEA/D.

population implementation for each algorithm separately (Tables 5.3–5.5). This enabled us to check the influence of applying different (disjoint) variable grouping strategies to the base algorithms. For DTLZ5 and DTLZ6 with ten objectives, over $15,000$ non-dominated solutions were found for each algorithm run; due to computational resource limitations, this means we were unable to calculate adjusted coverage. This is indicated in the tables by NEM (Not Enough Memory). For each of the algorithms, using the co-operative approach resulted in improved results for DTLZ1, DTLZ3, DTLZ6, and DTLZ7. When looking at the characteristics of the functions in Table 2.2, we can see that the three multi-modal functions are the ones for which including a grouping strategy improved the results. The fourth function, DTLZ6, is unimodal, but it is the only function that has unknown Pareto optima with bias. In the MOEA/D results for DTLZ1 with three objectives, the single population algorithm had good coverage but with a large standard deviation, indicating that there was at least one outlier.

5.4.3 Overlapping Variable Grouping Experiments

We apply the same strategy to determine the number of generations for FEA as we did for CCMOEA. For linear and random grouping with overlap, since we have a fixed number of 19 groups, we reduced the population size to 250 to reduce the number of function evaluations. However, DG dynamically assigns groups, which means we had to adjust the number of generations and the population size based on the number of groups created by DG. This resulted in a small number of generations and smaller population sizes for MOFEA with DG as the grouping strategy to assure a similar number of fitness evaluations would be performed for each run of the algorithm.

Table 5.6 shows the number of groups found for each problem's different objectives as well as the group sizes. The large number of subpopulation pairs that contain the same variables, in combination with the small factor and overlap sizes, could explain why

MOgDG and the collapsing of groups would result in fully separable or fully non-separable groupings when not maintaining the objective split. For example, for DTLZ1, there are three factors that contain 900 or more variables, and all other generated factors only contain 2–10 variables. However, there are 155 pairs that contain at least one overlapping variable. This indicates that most factors overlap, which would result in most if not all variables appearing to interact with each other, leading to a fully non-separable architecture when not allowing for overlap. The only problem that we found to be fully non-separable when applying DG is DTLZ7; all variables appeared to be interacting for each of the objectives. Because this effectively reduced the grouping to a single population, we did not run DG experiments on DTLZ7. When running our experiments for the ten-objective problems, we found that running FEA with the factor architecture generated using DG was too slow to run successfully for all problems but DTLZ1. In Tables 5.7–5.9, these results are indicated as not available (N/A), not to be confused with NEM that indicates we were unable to calculate the adjusted coverage due to memory limits.

Adding in overlap improved results compared to the single population for the problems where using a distinct grouping approach was beneficial. For the problems where single population NSGA2 remained the most effective approach, the inclusion of overlap also helped, but with no statistically significant difference. But, whereas the distinct groupings made little to no contribution to the total front, the FEA approach contributed new solutions to the front. This indicates that FEA may be exploring different parts of the variable and objective space, thus improving exploration as desired. Furthermore, we found that using differential grouping to create the overlapping factor architecture achieved the largest improvement in results for several problems, but this came with a trade-off in computational cost since DG created between 18 and 800 factors with a large amount of overlap (Table 5.6). Since we did not perform any parallelization or multi-processing, this resulted in runtimes

		M		
Problem	Characteristics	3	5	10
DTLZ1	# factors	37	202	152
	Avg factor size	54	20	59
	# pop. pairs w/ overlap	155	1455	2128
	Avg overlap size	6.5	4	17
DTLZ2	# factors	24	286	294
	Avg factor size	83	10	27
	# pop. pairs w/ overlap	84	627	2104
	Avg overlap size	12	5	13
DTLZ3	# factors	18	34	57
	Avg factor size	111	118	67
	# pop. pairs w/ overlap	46	217	2368
	Avg overlap size	22	28	15
DTLZ4	# factors	20	32	72
	Avg factor size	100	125	125
	# pop. pairs w/ overlap	74	151	767
	Avg overlap size	14	40	47
DTLZ5	# factors	26	182	283
	Avg factor size	77	22	25
	# pop. pairs w/ overlap	98	1391	1749
	Avg overlap size	10	4	12
DTLZ6	# factors	34	151	781
	Avg factor size	59	4	12
	# pop. pairs w/ overlap	143	1506	18488
	Avg overlap size	7	4	2
DTLZ7	# factors	1	1	1
	Avg factor size	1000	1000	1000

Table 5.6: Grouping summary with three, five, and ten objectives after applying DG to each objective.

NSGA2					
Problem	M	Single	Lin. Ov.	Rand. Ov.	DG
DTLZ1	3	0.33±0.431	0.13±0.096	0.20±0.139	0.35±0.231
	5	0.00±0.000	0.39±0.179	0.18±0.262	0.43±0.192
	10	0.00±0.000	0.16±0.034	0.00±0.000	0.84±0.034
DTLZ2	3	0.81±0.069	0.03±0.016	0.07±0.056	0.09±0.043
	5	0.76±0.252	0.07±0.091	0.02±0.025	0.15±0.142
	10	0.72±0.041	0.18±0.032	0.10±0.063	N/A
DTLZ3	3	0.00±0.000	0.46±0.042	0.01±0.028	0.53±0.056
	5	0.00±0.000	0.81±0.051	0.12±0.036	0.07±0.033
	10	0.11±0.201	0.78±0.174	0.11±0.046	N/A
DTLZ4	3	0.83±0.124	0.09±0.091	0.00±0.008	0.08±0.053
	5	0.32±0.156	0.11±0.063	0.02±0.020	0.55±0.098
	10	0.42±0.326	0.08±0.059	0.50±0.309	N/A
DTLZ5	3	0.87±0.071	0.01±0.016	0.06±0.085	0.07±0.023
	5	0.11±0.039	0.00±0.003	0.17±0.260	0.73±0.241
	10	0.06±0.018	0.76±0.141	0.18±0.141	N/A
DTLZ6	3	0.00±0.000	0.42±0.059	0.00±0.000	0.58±0.059
	5	0.01±0.022	0.87±0.102	0.06±0.052	0.06±0.051
	10	0.05±0.031	0.81±0.103	0.14±0.086	N/A
DTLZ7	3	0.00±0.000	1.00±0.000	0.00±0.000	N/A
	5	0.00±0.000	1.00±0.000	0.00±0.000	N/A
	10	0.00±0.000	1.00±0.000	0.00±0.000	N/A

Table 5.7: Average adjusted coverage: F-NSGA2.

SPEA2					
Problem	M	Single	Lin. Ov.	Rand. Ov.	DG
DTLZ1	3	0.00±0.000	0.22±0.047	0.15±0.048	0.62±0.047
	5	0.00±0.001	0.16±0.076	0.06±0.079	0.79±0.116
	10	0.00±0.001	0.27±0.050	0.00±0.000	0.73±0.050
DTLZ2	3	0.23±0.248	0.14±0.061	0.23±0.139	0.40±0.203
	5	0.49±0.010	0.07±0.011	0.07±0.030	0.37±0.029
	10	0.58±0.090	0.20±0.047	0.22±0.102	N/A
DTLZ3	3	0.00±0.000	0.72±0.043	0.07±0.033	0.21±0.044
	5	0.00±0.003	0.43±0.086	0.25±0.114	0.32±0.057
	10	0.00±0.015	0.42±0.091	0.57±0.100	N/A
DTLZ4	3	0.37±0.161	0.13±0.098	0.15±0.102	0.36±0.077
	5	0.15±0.071	0.09±0.017	0.27±0.062	0.49±0.030
	10	0.26±0.206	0.21±0.088	0.53±0.134	N/A
DTLZ5	3	0.23±0.128	0.13±0.076	0.29±0.204	0.36±0.165
	5	0.02±0.014	0.20±0.405	0.00±0.003	0.77±0.393
	10	0.02±0.013	0.91±0.206	0.07±0.203	N/A
DTLZ6	3	0.00±0.000	1.00±0.000	0.00±0.000	0.00±0.000
	5	0.00±0.000	0.00±0.000	0.22±0.204	0.78±0.204
	10	0.00±0.000	0.83±0.166	0.17±0.166	N/A
DTLZ7	3	0.00±0.000	1.00±0.000	0.00±0.000	N/A
	5	0.00±0.000	1.00±0.000	0.00±0.000	N/A
	10	0.00±0.000	1.00±0.000	0.00±0.000	N/A

Table 5.8: Average adjusted coverage: F-SPEA2.

MOEA/D					
Problem	M	Single	Lin. Ov.	Rand. Ov.	DG
DTLZ1	3	0.03±0.045	0.35±0.119	0.30±0.122	0.33±0.083
	5	0.04±0.056	0.22±0.096	0.18±0.184	0.55±0.250
	10	0.04±0.056	0.22±0.113	0.25±0.159	0.48±0.266
DTLZ2	3	0.08±0.102	0.15±0.147	0.25±0.163	0.52±0.192
	5	0.42±0.088	0.09±0.046	0.26±0.066	0.23±0.044
	10	0.49±0.047	0.14±0.032	0.37±0.060	N/A
DTLZ3	3	0.02±0.037	0.48±0.074	0.01±0.019	0.50±0.078
	5	0.24±0.126	0.64±0.118	0.04±0.025	0.08±0.037
	10	0.36±0.158	0.38±0.178	0.26±0.136	N/A
DTLZ4	3	0.03±0.066	0.11±0.107	0.45±0.091	0.41±0.086
	5	0.15±0.142	0.11±0.038	0.30±0.064	0.44±0.094
	10	0.20±0.208	0.06±0.031	0.74±0.188	N/A
DTLZ5	3	0.05±0.133	0.03±0.047	0.46±0.121	0.47±0.123
	5	N.E.M.	N.E.M.	N.E.M.	N.E.M.
	10	N.E.M.	N.E.M.	N.E.M.	N/A
DTLZ6	3	0.00±0.000	0.34±0.044	0.00±0.000	0.66±0.044
	5	N.E.M.	N.E.M.	N.E.M.	N.E.M.
	10	N.E.M.	N.E.M.	N.E.M.	N/A
DTLZ7	3	0.00±0.000	1.00±0.009	0.00±0.009	N/A
	5	0.00±0.000	0.94±0.060	0.06±0.060	N/A
	10	0.00±0.000	0.97±0.031	0.03±0.031	N/A

Table 5.9: Average adjusted coverage: F-MOEA/D.

of twelve hours or more for a single experiment for three objectives. [1]

5.5 Discussion

Table 5.11 shows the average AC across the three different algorithms for the decomposition methods that improved over their single population versions. We only show results for DTLZ1, DTLZ3, DTLZ4, and DTLZ7, since, for DTLZ2, no decomposition approach covered the majority of the union front when compared to their single population implementation, and we did not have enough computational resources to calculated the AC for DTLZ5 and DTLZ6. The coverage results differed greatly across the problems, and there was considerable variance for several solution sets. A possible explanation for this phenomenon is the use of function evaluations as the stopping criterion. Due to the stochastic aspect of population-based algorithms, the number of generations it takes for an algorithm to converge can vary widely. Since we chose a relatively small number of function evaluations (to enable running a large number of experiments), it is likely that the algorithms did not converge at the same rate across different iterations, resulting in non-trivial levels of variance in the final non-dominated archives. The coverage results also indicate that different algorithms explore different parts of the non-dominated space, as shown by the fact that a single algorithm's non-dominated solution set rarely contributed to the majority of the union front. The only exceptions to this were DTLZ2 and 3-objective DTLZ4, where single population NSGA2 performed well.

We present the hypervolume results in Tables 5.12–5.14 and spread in Tables 5.15–5.17. When considering HV, we see that all approaches performed well on DTLZ1, most likely due to the linear geometry of the Pareto front, making it one of the easier problems to solve. However, the spread results for DTLZ1 are very low for each algorithm. That said,

[1] We cannot compare runtimes accurately since, among other reasons, we were using different machines to perform different sets of experiments [80].

		DTLZ1		
Algorithhm	Grouping	3	5	10
NSGA2	Rand	0.08±0.014	0.02±0.011	0.02±0.010
	DG	0.26±0.065	0.06±0.007	0.09±0.036
SPEA2	Lin	0.21±0.011	0.14±0.009	0.05±0.007
	DG	0.25±0.045	0.41±0.014	0.40±0.018
MOEA/D	Lin. Ov.	0.20±0.054		
	DG		0.37±0.016	0.44±0.032
		DTLZ3		
Algorithhm	Grouping	3	5	10
NSGA2	Lin	0.13±0.015	0.12±0.008	
	Rand		0.10±0.007	
	Lin. Ov.		0.11±0.016	0.18±0.025
	DG	0.14±0.040		
SPEA2	Lin	0.20±0.023	0.15±0.005	0.26±0.008
	Rand	0.21±0.034	0.15±0.005	0.26±0.007
	Lin. Ov.	0.18±0.029	0.13±0.007	
MOEA/D	Rand		0.12±0.009	0.21±0.013
	Lin. Ov.		0.11±0.011	0.10±0.038
	DG	0.14±0.013		

Table 5.10: Average adjusted coverage: DTLZ1 and DTLZ3.

single population algorithms had a low HV for DTLZ3, DTLZ6, and DTLZ7, but these were the problems for which spread was higher. This seems to indicate that when more diverse solutions were found, they were located in regions of the objective space that did not add to

		DTLZ4		
Algorithm	Grouping	3	5	10
NSGA2	Single	0.91±0.027	0.28±0.063	0.46±0.071
	Rand. Ov.			0.25±0.121
	DG		0.35±0.042	
SPEA2	Rand. Ov.			0.12±0.026
	DG	0.07±0.034	0.12±0.044	
MOEA/D	Rand. Ov.	0.02±0.013		0.16±0.025
	DG		0.25±0.043	
		DTLZ7		
Algorithm	Grouping	3	5	10
NSGA	Rand	0.21±0.010	0.19±0.008	0.23±0.015
	Lin. Ov.	0.20±0.010	0.17±0.010	0.09±0.028
SPEA	Rand	0.05±0.018	0.18±0.014	0.26±0.018
	Lin. Ov.	0.16±0.007	0.14±0.013	0.21±0.017
MOEA/D	Rand	0.19±0.005	0.18±0.008	0.18±0.031
	Lin. Ov.	0.20±0.010	0.14±0.032	0.04±0.017

Table 5.11: Average adjusted coverage: DTLZ4 and DTLZ7.

an improved Pareto front approximation.

Single population NSGA2 outperformed the other algorithms significantly on DTLZ2. DTLZ2 is a unimodal, concave problem that is considered to be separable. Given this, it makes sense that a single population approach would perform well, since creating separate subproblems could introduce more complexity than is necessary, resulting in slower convergence. When multi-modality was introduced in DTLZ3, we found that the results

NSGA2							
Problem	M	Single	Linear	Lin. Ov.	Random	Rand. Ov.	DG
DTLZ1	3	0.989±0.018	0.996±0.002	0.995±0.005	0.995±0.003	0.999±0.001	0.999±0.001
	5	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000
	10	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	N/A
DTLZ2	3	0.999±0.000	0.173±0.015	0.999±0.001	0.140±0.030	0.997±0.003	0.999±0.000
	5	0.992±0.007	0.227±0.035	0.991±0.005	0.191±0.036	0.977±0.018	0.996±0.003
	10	0.986±0.004	0.494±0.072	0.984±0.007	0.471±0.031	0.926±0.016	N/A
DTLZ3	3	0.410±0.016	0.988±0.003	0.989±0.002	0.990±0.001	0.818±0.045	0.991±0.001
	5	0.339±0.039	0.999±0.000	0.999±0.000	0.999±0.000	0.735±0.041	0.702±0.054
	10	0.270±0.037	0.999±0.000	0.999±0.000	0.999±0.000	0.660±0.057	N/A
DTLZ4	3	0.999±0.002	0.147±0.018	0.998±0.003	0.220±0.010	0.999±0.001	0.999±0.000
	5	0.991±0.000	0.000±0.000	0.994±0.005	0.381±0.043	0.999±0.000	0.999±0.000
	10	0.989±0.000	0.000±0.000	0.991±0.008	0.457±0.068	0.999±0.000	N/A
DTLZ5	3	0.999±0.000	0.159±0.019	0.995±0.003	0.172±0.017	0.994±0.003	0.999±0.001
	5	0.946±0.013	0.216±0.043	0.665±0.073	0.491±0.277	0.878±0.022	0.984±0.007
	10	0.931±0.014	0.540±0.071	0.862±0.018	0.617±0.045	0.859±0.017	N/A
DTLZ6	3	0.172±0.021	0.999±0.000	0.999±0.000	0.999±0.000	0.319±0.025	0.999±0.000
	5	0.104±0.025	0.999±0.000	0.999±0.000	0.999±0.000	0.273±0.017	0.315±0.088
	10	0.096±0.016	0.999±0.000	0.999±0.000	0.999±0.000	0.221±0.070	N/A
DTLZ7	3	0.177±0.002	0.721±0.003	0.725±0.006	0.726±0.004	0.606±0.010	N/A
	5	0.049±0.016	0.726±0.005	0.726±0.004	0.726±0.003	0.462±0.019	N/A
	10	0.002±0.001	0.711±0.002	0.710±0.003	0.719±0.002	0.125±0.031	N/A

Table 5.12: NSGA2 Hypervolume (HV) results.

were much less straightforward. Grouping strategies improved HV, but not necessarily spread. But when considering AC, the different versions of SPEA2 contributed the most non-dominated solutions for all three instances of DTLZ3. This makes it difficult to draw any

SPEA2							
Problem	M	Single	Linear	Lin. Ov.	Random	Rand. Ov.	DG
DTLZ1	3	0.996±0.008	0.998±0.001	0.999±0.001	0.998±0.001	0.999±0.000	0.999±0.000
	5	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000
	10	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	N/A
DTLZ2	3	0.981±0.003	0.277±0.017	0.996±0.001	0.295±0.021	0.997±0.001	0.999±0.000
	5	0.978±0.001	0.398±0.037	0.960±0.018	0.460±0.019	0.983±0.009	0.999 0
	10	0.977±0.004	0.541±0.042	0.912±0.022	0.683±0.027	0.964±0.012	N/A
DTLZ3	3	0.261±0.025	0.903±0.008	0.916±0.012	0.949±0.013	0.612±0.025	0.620±0.020
	5	0.236±0.006	0.912±0.007	0.928±0.009	0.981±0.004	0.621±0.020	0.716±0.014
	10	0.192±0.026	0.946±0.011	0.973±0.011	0.994±0.003	0.546±0.017	N/A
DTLZ4	3	0.973±0.002	0.184±0.003	0.990±0.014	0.266±0.072	0.999±0.000	0.999±0.000
	5	0.968±0.005	0.000±0.000	0.992±0.000	0.481±0.042	0.999±0.002	0.999±0.000
	10	0.939±0.013	0.000±0.000	0.975±0.012	0.534±0.045	0.999±0.001	N/A
DTLZ5	3	0.983±0.005	0.255±0.010	0.979±0.020	0.292±0.009	0.998±0.001	0.998±0.001
	5	0.972±0.009	0.373±0.023	0.559±0.177	0.525±0.162	0.980±0.006	0.963±0.008
	10	0.974±0.006	0.473±0.048	0.632±0.039	0.843±0.074	0.944±0.021	N/A
DTLZ6	3	0.193±0.014	0.999±0.000	0.999±0.000	0.999±0.000	0.365±0.018	0.400±0.018
	5	0.201±0.015	0.998±0.000	0.998±0.000	0.999±0.000	0.377±0.028	0.377±0.037
	10	0.160±0.019	0.998±0.000	0.998±0.000	0.999±0.000	0.352±0.069	N/A
DTLZ7	3	0.033±0.009	0.735±0.004	0.732±0.001	0.732±0.003	0.271±0.086	N/A
	5	0.011±0.006	0.734±0.002	0.733±0.001	0.734±0.003	0.077±0.024	N/A
	10	0.001±0.002	0.728±0.002	0.726±0.002	0.731±0.001	0.014±0.007	N/A

Table 5.13: SPEA2 Hypervolume (HV) results.

conclusions on which decomposition method would be best to solve problems with a multi-modal, concave Pareto front; however, it appears that SPEA2 might be the appropriate base algorithm to use.

MOEA/D							
Problem	M	Single	Linear	Lin. Ov.	Random	Rand. Ov.	DG
DTLZ1	3	0.999±0.000	0.999±0.001	0.999±0.000	0.999±0.001	0.999±0.000	0.999±0.000
	5	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000
	10	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	N/A
DTLZ2	3	0.987±0.001	0.305±0.021	0.999±0.001	0.319±0.016	0.999±0.001	0.999±0.000
	5	0.993±0.003	0.279±0.062	0.997±0.003	0.542±0.018	0.999±0.001	0.998±0.000
	10	0.986±0.007	0.413±0.083	0.999±0.000	0.464±0.105	0.999±0.001	N/A
DTLZ3	3	0.538±0.011	0.967±0.018	0.989±0.002	0.975±0.012	0.935±0.015	0.990±0.001
	5	0.496±0.051	0.999±0.000	0.999±0.000	0.999±0.001	0.929±0.011	0.887±0.020
	10	0.444±0.053	0.999±0.001	0.999±0.000	0.998±0.003	0.844±0.019	N/A
DTLZ4	3	0.978±0.031	0.137±0.069	0.991±0.008	0.273±0.074	0.999±0.000	0.993±0.012
	5	0.996±0.004	0.281±0.068	0.999±0.001	0.499±0.032	0.999±0.000	0.999±0.000
	10	0.978±0.030	0.002±0.002	0.992±0.014	0.422±0.132	0.999±0.000	N/A
DTLZ5	3	0.980±0.002	0.261±0.024	0.997±0.001	0.294±0.013	0.999±0.000	0.999±0.000
	5	0.933±0.031	0.219±0.026	0.643±0.081	0.312±0.296	0.989±0.009	0.985±0.010
	10	0.922±0.032	0.312±0.043	0.775±0.061	0.317±0.087	0.953±0.015	N/A
DTLZ6	3	0.122±0.046	0.999±0.000	0.999±0.000	0.999±0.000	0.741±0.011	0.999±0.000
	5	0.112±0.038	0.999±0.000	0.999±0.000	0.999±0.000	0.317±0.027	0.306±0.118
	10	0.150±0.071	0.999±0.000	0.999±0.000	0.999±0.000	0.229±0.028	N/A
DTLZ7	3	0.179±0.017	0.745±0.004	0.746±0.002	0.745±0.001	0.547±0.043	N/A
	5	0.053±0.013	0.730±0.002	0.736±0.008	0.742±0.001	0.183±0.134	N/A
	10	0.023±0.006	0.713±0.005	0.711±0.002	0.737±0.001	0.006±0.008	N/A

Table 5.14: MOEA/D Hypervolume (HV) results.

Linear and random decomposition approaches resulted in low HV for DTLZ2, DTLZ4, and DTLZ5 (which are all unimodal) for each objective, although they did not improve as the number of objectives increased. Random grouping had significantly better performance

NSGA2							
Problem	M	Single	Linear	Lin. Ov.	Random	Rand. Ov.	DG
DTLZ1	3	0.081±0.047	0.025±0.002	0.024±0.005	0.023±0.003	0.028±0.001	0.026±0.001
	5	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	10	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	N/A
DTLZ2	3	0.157±0.156	1.620±0.038	0.222±0.053	1.717±0.009	0.137±0.124	0.047±0.024
	5	0.006±0.004	1.689±0.029	0.838±0.170	1.714±0.010	0.257±0.451	1.674±0.007
	10	0.000±0.000	1.580±0.083	1.190±0.070	1.715±0.005	0.001±0.002	N/A
DTLZ3	3	1.172±0.083	0.416±0.005	0.440±0.012	0.421±0.000	1.086±0.111	0.554±0.052
	5	0.284±0.150	0.387±0.011	0.395±0.008	0.399±0.002	0.870±0.188	1.092±0.134
	10	0.000±0.000	0.417±0.015	0.417±0.018	0.425±0.012	0.180±0.119	N/A
DTLZ4	3	0.104±0.121	1.344±0.059	0.156±0.044	1.714±0.012	0.155±0.030	0.049±0.017
	5	0.004±0.000	0.000±0.000	0.151±0.050	1.723±0.003	0.390±0.030	0.034±0.034
	10	0.004±0.000	0.000±0.000	0.101±0.030	1.492±0.137	0.485±0.029	N/A
DTLZ5	3	0.105±0.104	1.651±0.020	0.283±0.076	1.670±0.017	0.016±0.013	0.036±0.015
	5	0.011±0.007	1.677±0.037	1.546±0.055	1.615±0.153	0.155±0.112	1.438±0.162
	10	0.000±0.000	1.622±0.071	1.569±0.086	1.396±0.615	0.013±0.023	N/A
DTLZ6	3	0.836±0.116	0.002±0.000	0.002±0.000	0.002±0.000	1.148±0.156	0.040±0.047
	5	0.106±0.041	0.017±0.002	0.145±0.082	0.003±0.001	0.891±0.105	1.051±0.252
	10	0.000±0.000	0.007±0.004	0.131±0.088	0.011±0.009	0.133±0.147	N/A
DTLZ7	3	1.066±0.124	0.766±0.270	0.875±0.018	0.579±0.223	0.687±0.375	N/A
	5	0.548±0.220	0.920±0.285	0.947±0.139	0.752±0.249	1.372±0.150	N/A
	10	0.023±0.006	0.713±0.005	0.711±0.002	0.737±0.001	0.006±0.008	N/A

Table 5.15: NSGA2 Spread Indicator (SI) results.

on DTLZ7 for HV on all objectives and for spread on five- and ten-objective DTLZ7. Given that DTLZ7 is known to have a disconnected geometry, this makes sense since overlap could add connections between variables where there are none.

SPEA2							
Problem	M	Single	Linear	Lin. Ov.	Random	Rand. Ov.	DG
DTLZ1	3	0.038±0.020	0.027±0.001	0.027±0.001	0.026±0.001	0.028±0.000	0.028±0.000
	5	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	10	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	N/A
DTLZ2	3	0.026±0.007	1.299±0.070	0.130±0.030	1.404±0.098	0.083±0.065	0.047±0.012
	5	0.016±0.003	1.421±0.067	0.354±0.101	1.389±0.031	0.199±0.281	0.065 0.01
	10	0.001±0.001	1.285±0.064	0.755±0.192	1.464±0.101	0.110±0.082	N/A
DTLZ3	3	0.912±0.113	0.323±0.009	0.366±0.019	0.375±0.016	1.429±0.048	1.558±0.023
	5	0.615±0.061	0.277±0.010	0.326±0.013	0.358±0.010	1.162±0.113	1.351±0.120
	10	0.524±0.115	0.328±0.022	0.367±0.013	0.393±0.014	0.422±0.172	N/A
DTLZ4	3	0.010±0.002	1.255±0.094	0.204±0.068	1.498±0.147	0.098±0.041	0.046±0.025
	5	0.012±0.002	0.000±0.000	0.109±0.033	1.547±0.173	0.312±0.075	0.214±0.149
	10	0.012±0.001	0.000±0.000	0.102±0.073	1.405±0.144	0.324±0.036	N/A
DTLZ5	3	0.031±0.005	1.168±0.102	0.221±0.077	1.320±0.090	0.030±0.019	0.024±0.004
	5	0.011±0.004	1.241±0.075	1.122±0.239	1.493±0.085	0.174±0.060	0.296±0.207
	10	0.001±0.001	1.334±0.062	1.329±0.080	0.111±0.054	0.105±0.064	N/A
DTLZ6	3	0.892±0.128	0.002±0.001	0.001±0.000	0.002±0.000	1.484±0.070	1.511±0.077
	5	0.794±0.123	0.002±0.003	0.093±0.078	0.007±0.003	1.189±0.102	1.164±0.251
	10	0.491±0.142	0.000±0.000	0.022±0.043	0.005±0.003	0.317±0.098	N/A
DTLZ7	3	0.617±0.059	1.272±0.017	1.254±0.062	1.326±0.031	0.929±0.122	N/A
	5	0.714±0.068	1.470±0.121	1.487±0.079	1.561±0.017	1.238±0.161	N/A
	10	0.716±0.063	1.425±0.074	1.425±0.140	1.490±0.036	1.336±0.060	N/A

Table 5.16: SPEA2 Spread Indicator (SI) results.

Overall, overlapping methods resulted in a good HV for most functions, with DTLZ7 being the only exception. The overlapping techniques appeared to be the most beneficial for DTLZ4, DTLZ5, and DTLZ6, each with statistically significant results compared to the

MOEA/D							
Problem	M	Single	Linear	Lin. Ov.	Random	Rand. Ov.	DG
DTLZ1	3	0.118±0.002	0.028±0.001	0.028±0.000	0.028±0.001	0.029±0.000	0.025±0.000
	5	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	10	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	N/A
DTLZ2	3	0.539±0.009	1.719±0.018	0.326±0.063	1.689±0.044	0.210±0.083	0.163±0.073
	5	0.545±0.041	1.729±0.002	0.738±0.177	1.718±0.010	0.223±0.093	0.540±0.022
	10	0.053±0.046	1.572±0.070	0.584±0.110	1.219±0.087	0.068±0.058	N/A
DTLZ3	3	1.639±0.225	0.395±0.018	0.447±0.013	0.407±0.010	0.846±0.047	0.555±0.086
	5	1.506±0.106	0.393±0.004	0.410±0.016	0.380±0.023	1.133±0.053	1.380±0.077
	10	0.236±0.236	0.330±0.051	0.380±0.045	0.380±0.034	0.306±0.165	N/A
DTLZ4	3	0.456±0.092	1.127±0.563	0.161±0.094	1.563±0.177	0.180±0.027	0.091±0.050
	5	0.519±0.050	1.478±0.127	0.427±0.146	1.652±0.120	0.324±0.038	0.423±0.049
	10	0.488±0.084	0.300±0.557	0.179±0.152	1.562±0.166	0.463±0.039	N/A
DTLZ5	3	0.576±0.024	1.717±0.012	0.544±0.070	1.713±0.014	0.131±0.011	0.155±0.035
	5	0.254±0.148	1.496±0.119	1.512±0.122	1.621±0.112	0.484±0.043	0.434±0.089
	10	0.003±0.005	1.375±0.210	1.404±0.134	1.588±0.114	0.009±0.010	N/A
DTLZ6	3	1.723±0.003	0.002±0.000	0.002±0.000	0.002±0.000	1.360±0.079	0.002±0.000
	5	1.098±0.160	0.009±0.008	0.070±0.078	0.017±0.008	1.140±0.100	0.914±0.552
	10	0.054±0.083	0.001±0.000	0.001±0.000	0.001±0.000	0.180±0.213	N/A
DTLZ7	3	1.374±0.017	1.320±0.006	1.328±0.007	1.289±0.030	1.331±0.058	N/A
	5	1.571±0.070	1.477±0.108	1.547±0.104	1.260±0.310	1.027±0.394	N/A
	10	1.576±0.159	0.604±0.461	0.430±0.335	1.063±0.287	0.458±0.336	N/A

Table 5.17: MOEA/D Spread Indicator (SI) results.

other results. More specifically, for DTLZ4, random overlap significantly improved HV for each of the base-algorithms, as did DG for the three- and five-objective problems. In the case of DTLZ5, DG significantly improved results for three and five objectives for NSGA2

and MOEA/D. Additionally, random overlap improved results for SPEA2 and MOEA/D for all objectives. Lastly, linear overlap had significantly better results for all three algorithms for all objectives when applied to DTLZ6. This is interesting since DTLZ5 and DTLZ6 have unknown geometries, and DTLZ4 and DTLZ6 include bias (Chapter 2 Section 2.3.2). The results using overlapping subpopulations are consistent with results found by Pryor *et al.* [118]; we find that linear and random overlap (which create connected groups) often perform well even though they do not perform any type of variable interaction learning. Unfortunately, the coverage results for DTLZ5 and DTLZ6 were inconclusive due to the aforementioned memory shortage.

Last, we noticed some interesting time-related phenomena in the results that warrant further investigation. Specifically, we observed that as the number of non-dominated solutions grew (as in DTLZ5 and DTLZ6), all co-operative versions of MOEA/D slowed down more than the other algorithms. We believe this may be the result of how the elite population is updated by the algorithm, since the algorithm updates the elite population after every offspring y' is generated [175]. This seemed to result in larger sets of non-dominated solutions as compared to the other algorithms.

5.6 Concluding Remarks

We identified certain function characteristics that seem to benefit from specific variable grouping approaches. We summarize our findings per characteristic. When comparing multi- and uni-modal functions, we found that uni-modal problems with more straightforward geometries (DTLZ2 and DTLZ4) did not benefit much from variable grouping strategies. When we considered different geometries more specifically we found the following trends:

1. Disconnected geometry (DTLZ7): Random grouping had signficantly better performance.

2. Unknown geometries and/or bias (DTLZ4, DTLZ5, and DTLZ6): Overlapping groups seemed to be the most beneficial approach, especially in the five and ten objective cases.

Lastly, looking at the separability of functions, we confirm results by other authors:

1. Separable: Variable grouping improved results for some but not all separable functions, indicating that other function characteristics may be more important when dealing with fully separable problems.

2. Unknown separability (DTLZ5, DTLZ6, and DTLZ7): Overlapping groups generally improved results, where three objective DTLZ5 was the only exception.

We confirmed the results of Li *et al.* [85], who found that certain problems in the DTLZ test-suite are partially separable when applying DG to the separate objectives. To address this partial separability, we used overlapping subpopulations through the MOFEA framework. We showed that MOFEA is applicable to different algorithms, and offers improvements over single population algorithms in different situations. Furthermore, we found that DG had good results, but it was slower than the other grouping strategies. We believe this is due to the number of groups created as well as the number of overlapping variables, since more overlap means more iterations in the compete step of MOFEA. Overall, random and linear overlap improved the results as compared to single and disjoint populations in many cases, providing more support for the idea that a connected grouping may be more important than accurate variable interaction learning.

Finally, we empirically confirmed that as the number of objectives increases, so does the number of non-dominated solutions. For problems DTLZ5 and DTLZ6, the final non-dominated solution sets contained thousands of solutions for the five- and ten-objective versions of the problem. This raised the question how such large solution sets could be made

more manageable for a human end-user to make a decision on which solution to use; which is the focus of the next chapter.

CHAPTER SIX

SOLUTION SET REDUCTION

In the previous chapter, the five and ten objective results for the DTLZ5 and DTLZ6 benchmark problems provided a good example of the curse of dimensionality that occurs in MaOO. This led us to exploring ways to reduce the size of the final non-dominated solution set, which is the primary focus of this chapter. Inspired by the idea of overlapping subpopulations, we created the Objective Archive Management (OAM) strategy. We present OAM as a novel solution set reduction approach and offer a visualization strategy of the reduced solution sets in higher objective spaces.

6.1 Related Work and Motivation

To reduce the number of non-dominated solutions as the objective space increases, decomposition-based approaches such as MOEA/D [175] and NSGA3 [34] are used widely. However, these approaches rely on pre-defined reference vectors (using a weight vector) to guide the search. Such approaches come with their own set of issues, the most prominent being the decrease in diversity and the need to determine the appropriate weight vectors [70]. Several adjustments to these decomposition based methods have been proposed to address these issues [28, 46, 59, 93, 115, 172].

To better address the issue of diversity loss in MaOO, methods adjusting the selection criteria have been proposed. This is accomplished by changing the Pareto dominance relationship or creating a specialized fitness function, where the adjustments focus on achieving a good balance between diversity and convergence. This has been accomplished through methods such as α-dominance [66], dominance-ratio adjustment [125], objective reduction based on dominance relations [12], maximum-vector-angle-first principle [162], generalized

Pareto optimality [181], and adjusted distribution estimation [166]. Similarly, using a performance indicator to evaluate solutions can be an effective strategy. Hypervolume-based evolutionary algorithms are the most common approach [35, 102], but the hypervolume calculation has two serious drawbacks: its dependence on a reference point and the high computational cost [26].

Archive maintenance tactics offer a different kind of solution to the problems found in MaOO. In this approach, the focus lies on an external archive that maintains the set of found non-dominated solutions. Archive management strategies often use ideas from the aforementioned methods, for example, using the hypervolume indicator [78], reference-point based archive management [10], and two-archive based methods where one archive focuses on diversity (indicator-based) and the other on convergence (Pareto-based) [16, 156, 176].

Each of the aforementioned approaches offers ways to balance convergence and diversity as the objectives increase, but they do not address the issue of large non-dominated solution sets for the end-user to inspect. Most of the research focusing on helping the decision maker in their choice for a final solution focuses on dimensionality reduction to aid in visualization [26] or by incorporating preferences directly into the search processes [56, 86, 154]. However, dimensionality reduction comes at the cost of information loss in the objective space, and preferences are highly domain-specific.

There has been research in selecting a subset of solutions after the final non-dominated solution set has been generated, but most research in this area has focused on using the hypervolume metric to find the best solution subset [102, 131]. However, as previously mentioned, the hypervolume indicator comes with two major drawbacks [26]. A more promising approach was presented by Takagi *et al.*, where they perform environmental selection based on an MOEA's chosen selection procedure [142]. The authors used NSGA2 as an example for their method. Given a solution set $\mathcal{S}$ that needs to be reduced to a size k, the authors propose sorting $\mathcal{S}$ based on NSGA2's environmental selection procedure:

the crowding distance. In other words, the crowding distance is calculated for each of the solutions in $\mathcal{S}$, and the k best solutions according to the crowding distance are selected to create a new solution set. There are two downsides to this approach: it requires 1) a pre-defined solution set size and 2) a specific algorithm to be selected to determine the type of environmental selection to be applied. The latter means that expert knowledge is required to make an appropriate choice [160].

With our approach, we focus on managing the number of non-dominated solutions using a multi-archive approach to facilitate decision making for the end user. Our archive management strategy creates separate archives of non-dominated solutions for each objective, where each archive focuses on maintaining the best solutions for the relevant objective while introducing diversity. We update the objective archives throughout the generations, and after the final generation, we find the solutions that belong to multiple archives to create a small final solution set to present to the end user.

It is important to note that creating a diverse set of non-dominated solutions is not part of this research. We did not aim to improve the optimization process that tries to find an approximation to the optimal Pareto front. Our aim is to reduce these solution sets, as generated by any MOEA, for an end user that wishes to make as little decisions as possible in regards to *how* to reduce a large non-dominated solution set. Furthermore, our approach does not require reducing the number of objectives, defining a fixed size of the final solution set, or the need for expert knowledge (either to determine reference vectors or for algorithm selection). In other words, our algorithm can be used by people with limited to no knowledge on MaOO.

6.2 Objective Archive Management

Since we are organizing a group of non-dominated solutions S (the archive) into subgroups S_i for each objective, we call our algorithm "objective" archive management

(OAM). Key to the approach is the management of diversity of the solutions in the archive(s). Diversity is determined by creating a dissimilarity or distance matrix $\mathbf{M}_d$ for the solutions' variables ($\mathbf{M}_{d_{var}}$) and fitness scores ($\mathbf{M}_{d_{fit}}$) separately. We refer to the form as being diverse in the variable space and the latter as being diverse in the objective space.

Our approach is as follows. For each objective M_i, we sort S according to that objective. The first $k\%$ solutions of the sorted set are added to S_i. Then the second $k\%$ of the original solution set are selected, from which $\ell\%$ diversity solutions are chosen; half of which are diverse in the objective space, and half of which are diverse in the variable space. Selecting for diversity in this way ensures that the chosen diversity solutions are still good solutions for objective M_i. The objective archive solution selection process is shown in Figure 6.1 and the general pseudocode is given in Algorithm 6.1.

In our experiments we used the cosine similarity metric to measure diversity due to its useful qualities in high dimensional spaces. Specifically, cosine similarity distinguishes different solutions from a directional perspective, making it a good choice to diversify the search space [164]. We select the unique solutions with the highest dissimilarity score to be added to the objective archive (Algorithm 6.2). We look at both variable and objective diversity. Each solution $s \in \mathcal{S}$ consists of a set of variables $X \leftarrow \{x_0, \ldots, x_n\}$ and a set of fitness scores $F \leftarrow \{f_0, \ldots, f_M\}$, where n is the variable dimensionality, and M is the number of objectives. The function "cosine_distance" in Algorithm 6.2 refers to the pairwise analysis of all $s \in \mathcal{S}$ in the variable (Line 5) or objective space (Line 8), resulting in a symmetrical $|\mathcal{S}| \times |\mathcal{S}|$ distance matrix, from which we select the solutions with the largest distance. In other words, for both the variable values and the objective values, we select the solution pairs with the highest dissimilarity, check whether these solutions have been added to the archive (from being part of a different pair of dissimilar solutions), and if not, add the new solutions to the archive. By checking both objective and variable diversity, we aim to account for biased problems. Biased problems have functions with a large discrepancy

Algorithm 6.1 Objective Archive Management

Input: Number of objectives M, non-dominated archive $\mathcal{N}$, parameter best solutions k, parameter diverse solutions ℓ

$\mathcal{F} \leftarrow \{\}$
$k \leftarrow \lceil k \times |\mathcal{N}| \rceil$
for all $i = 0$ to M **do**
    $\mathcal{F}_i \leftarrow \{\}$
    $\mathcal{N}' \leftarrow \text{sort}(\mathcal{N}, i)$
    $\mathcal{F}_i \leftarrow \mathcal{F}_i \cup \mathcal{N}'[:k]$
    $\mathcal{N}'' \leftarrow \text{diversify\_archive}(\mathcal{N}'[k:2k], \ell)$ // Algorithm 6.2
    $\mathcal{F}_i \leftarrow \mathcal{F}_i \cup \mathcal{N}''$
return $\mathcal{F}$

between the distribution of solutions in the search space and the distribution of solutions in the objective space [64]. The collection of archives is referred to as the Objective Archive (OA).

We can now use the created OA to reduce the non-dominated solution set into a more manageable size. We do this by counting how many times each solution occurs in the M objective archives (Algorithm 6.3). The user can then choose how many archives a solution needs to belong to in order to be included in the final solution set. For example, if the user wants as little choice as possible, the overlap count of a single solution should be close to the number of objectives M. Note that if overlap is set equal to M, this means each solution is in the top $2k\%$ of the population for each objective.

The OAM approach can be applied in two different ways to any MOO or MaOO algorithm. First, it can be used as an external archive throughout an algorithm's optimization process to improve the convergence/diversity trade-off as desired. Second, it can be applied after an algorithm has terminated to reduce the solution set size. We refer

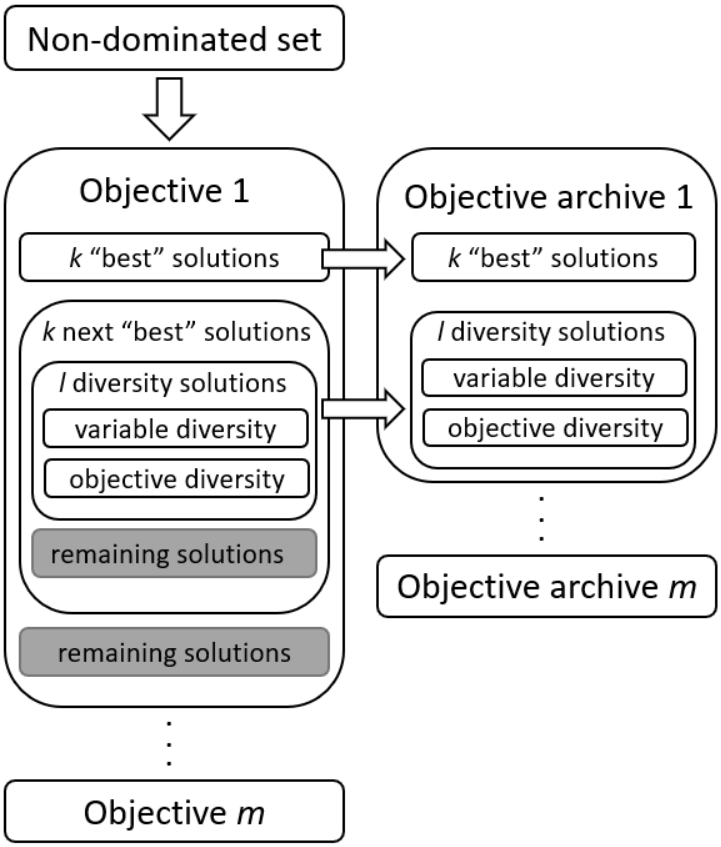


Figure 6.1: Visual representation of the OAM process to create the objective archives.

to the former as E-OAM (external OAM), and the latter as S-OAM (single OAM).

In E-OAM, the OA is updated continuously to include any newly found non-dominated solutions, serving as an external archive. The OA can be kept as a storage mechanism without injecting any solutions back into the algorithm, or it can be used to inject a reduced number of non-dominated solutions (by finding the overlapping solutions) into the next generation of the algorithm to help guide the search. In this paper, we only look at keeping and updating an external archive, without injection, to create a final, reduced solution set. When applying S-OAM, we generate the OA after the algorithm of choice has terminated. In other words, an OA is created based on the final non-dominated solution set. The overlapping solutions can then be found to reduce the number of solutions further to a more manageable size.

Algorithm 6.2 Diversify Archive

Input: Solution set $\mathcal{S}$, ℓ diverse solutions

$\ell \leftarrow \lceil \ell \times |\mathcal{S}| \rceil$

$S' \leftarrow \{\}$

$S_{var} \leftarrow \{X_0, \ldots, X_{|\mathcal{S}|}\}$

$S_{fit} \leftarrow \{F_0, \ldots, F_{|\mathcal{S}|}\}$

$\mathbf{M}_{d_{var}} \leftarrow \text{cosine\_distance}(S_{var})$

$S'_{var} \leftarrow \text{sort}(\mathbf{M}_{d_{var}})$

$S' \leftarrow S' \cup S'_{var}[: \ell/2]$

$\mathbf{M}_{d_{fit}} \leftarrow \text{cosine\_distance}(S_{fit})$

$S'_{fit} \leftarrow \text{sort}(\mathbf{M}_{d_{fit}})$

$S' \leftarrow S' \cup S'_{fit}[: \ell/2]$

return S'

6.3 Experimental Approach

In our studies, we applied NSGA2, MOEA/D, and SPEA2 to the DTLZ [37] and WFG [65] benchmark suites. We found that single-population NSGA2 performed well on MaOO problems (as compared to MOEA/D and SPEA2). As a result, we decided to use NSGA2 as our base algorithm; however, as previously stated, the OAM approach could be applied to any algorithm. We compared OAM-NSGA2 to the environmental selection approach in [142] and NSGA3 [34], implemented using the pymoo library [6]. NSGA3 is a popular reference vector based approach created for MaOO that has been shown to perform well and to produce small final non-dominated solution sets [162]. Each algorithm was run with a population of 1000, for 100 generations. Through these studies, we found that functions DTLZ5, DTLZ6, WFG3, and WFG7 produced large non-dominated solution sets. We used these four functions for our experiments, each with five and ten objectives and with

Algorithm 6.3 Find Overlapping Solutions

Input: Objective archive OA, minimum overlap count oc

Initialize: Dictionary count $\leftarrow \{\}$, reduced solution set $\mathcal{S} \leftarrow \{\}$

arch $\leftarrow$ flatten(OA)
for all $X \in$ arch **do**
    count[X] $\leftarrow 0$
for all $i = 0$ to M **do**
    for all $X \in OA_i$ **do**
        count[X] = count[X] + 1
for all $x, c \in$ count **do**
    if $c \geq oc$ **then**
        $\mathcal{S} \leftarrow \mathcal{S} \cup \{x\}$
return $\mathcal{S}$

100 decision variables [65]. We performed 30 independent iterations of the algorithms on each problem and report Hypervolume (HV) [157] and Spread (S) [3]. Furthermore we visualize the obtained reduced solution sets through radar charts to obtain a more intuitive understanding of the solutions being selected. The Wilcoxon rank-sum test with $\alpha = 0.05$ was performed to assess statistical significance for all results.

6.4 Results

Our results consisted of four different sets of experiments. 1) We started by evaluating the convergence-divergence trade-off in the OAM strategy by varying the k and l parameters. 2) We compared solution set reduction using OAM to environmental selection. 3) We compared our external-archive approach using NSGA2 to NSGA3. And lastly, 4) we applied

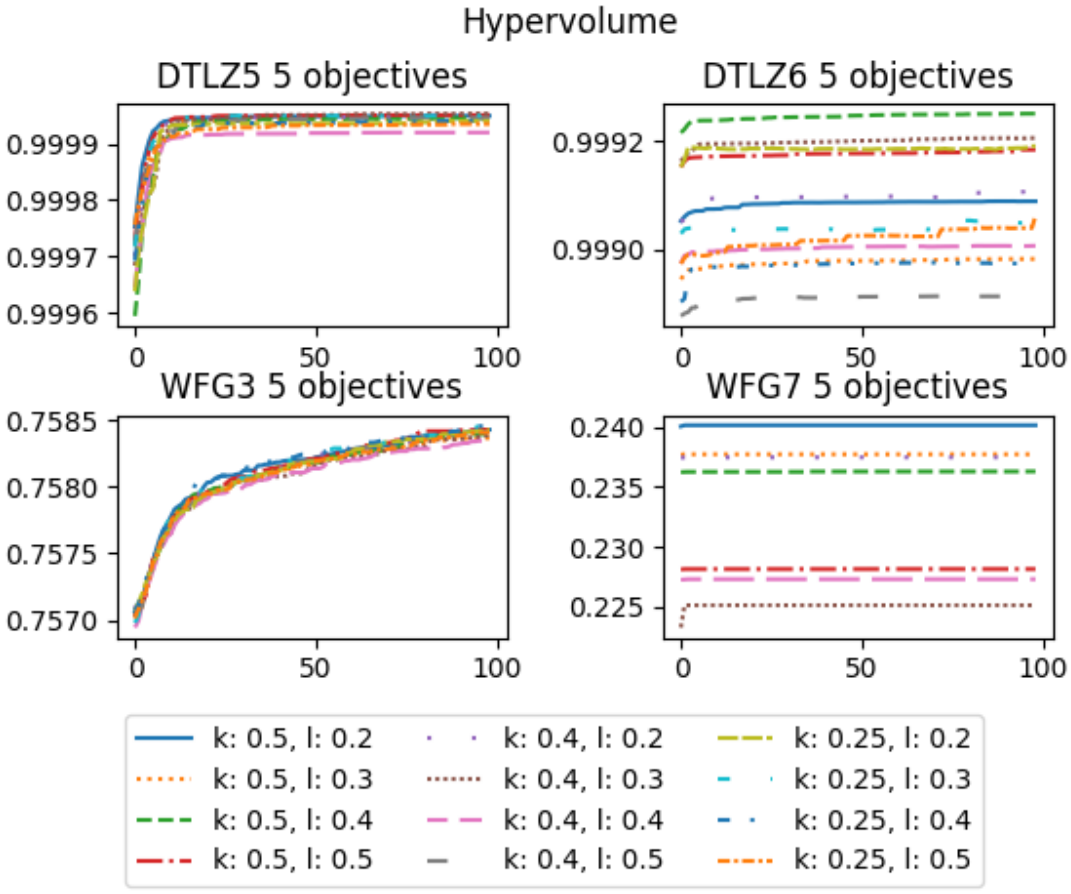


Figure 6.2: Five-objective HV results for OAM with different k and l parameter values.

the single-archive OAM approach to the solution sets found by NSGA3.

6.4.1 Convergence vs. Diversity

We examined the influence of the convergence and diversity parameters k and ℓ on the final OA quality. We ran experiments for all combinations of $k = \{0.25, 0.4, 0.5\}$ and $\ell = \{0.2, 0.3, 0.4, 0.5\}$. To obtain the results shown in Figures 6.2 – 6.5, we used E-OAM to maintain an external archive that was updated after each generation of NSGA2. At the end of each generation, the non-dominated solutions found by NSGA2 were added to each objective archive and re-evaluated for non-dominance before applying the OAM algorithm

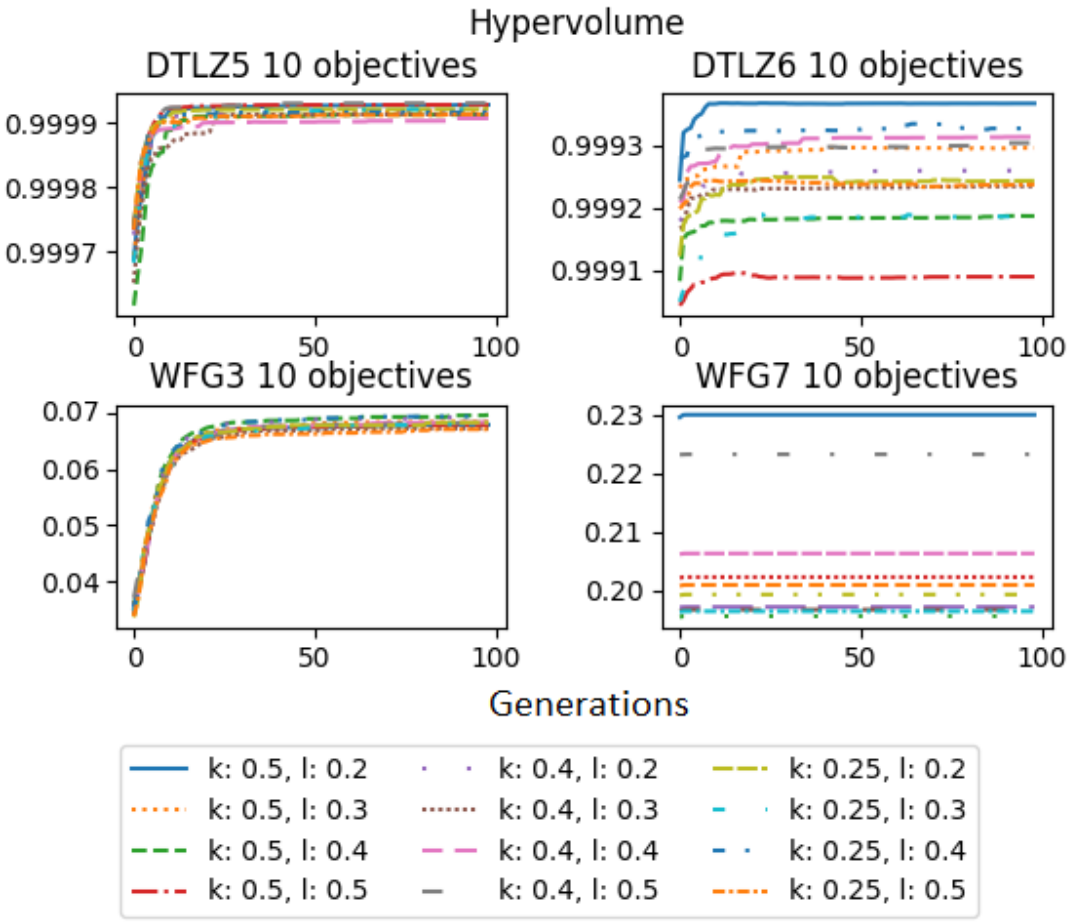


Figure 6.3: Ten-objective HV results for OAM with different k and l parameter values.

to obtain the updated archive. We present HV and S for the resulting external archive at each generation (without applying solution set reduction). We averaged the full runs of each algorithm to generate the final graphs.

Following statistical hypothesis testing, we found that there was no significant difference between the different parameter settings for HV and S on problems DTLZ5, DTLZ6, and WFG3. When considering the graphs in Figures 6.2 – 6.5, we see that there are only small differences between HV and S for those problems, and the convergence lines follow similar trends. The same does not hold true for WFG7, where we do find statistically

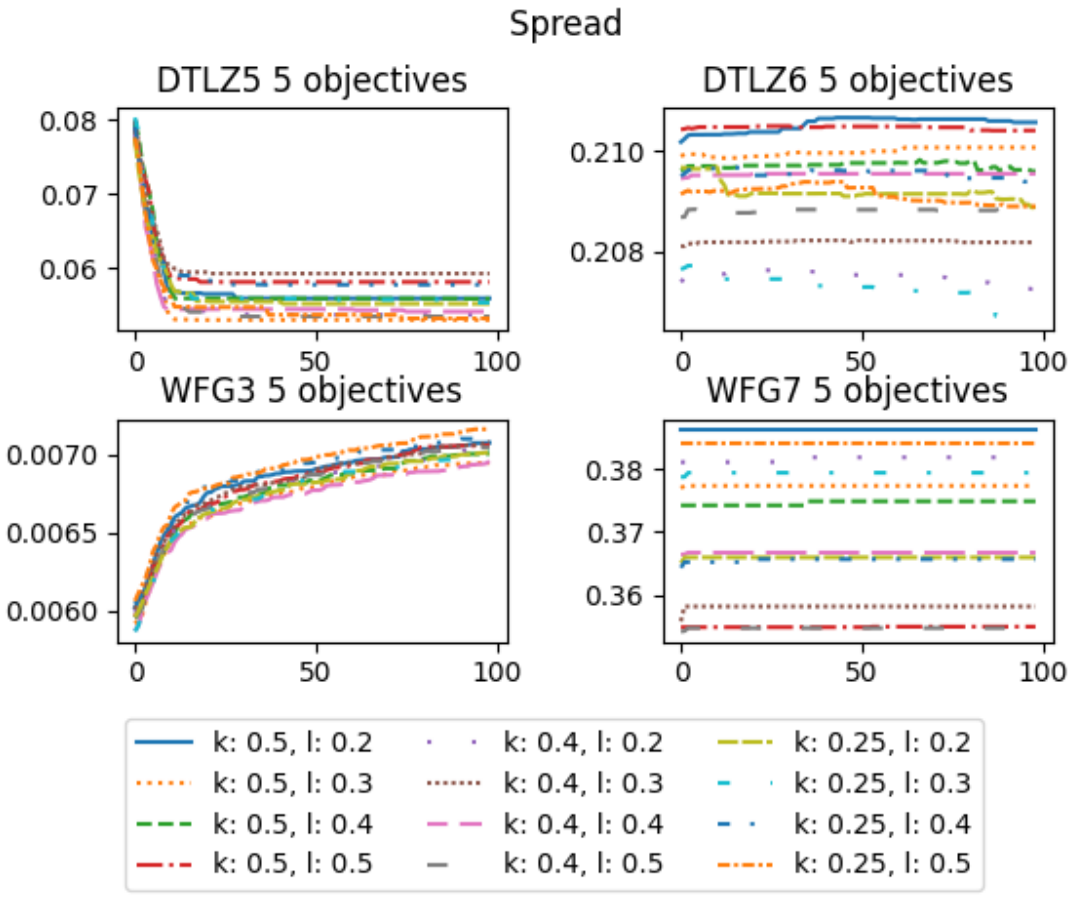


Figure 6.4: Five-objective S results for OAM with different k and l parameter values.

significant differences between the different parameter settings. However, there is little to no convergence for either HV or S for WFG7, regardless of the chosen k and ℓ parameters. This indicates the problem lies with the performance of NSGA2.Since the OA is only updated with solutions found by the underlying optimization algorithm, it is directly influenced by that algorithm's performance.

Table 6.1 shows the parameter combinations with the most promising results for each of the problems. These were chosen based on which parameter combination had a good balance between HV and S. But as mentioned previously, since the difference between the values is small and not significant for three out of four problems, the choice of k and ℓ may

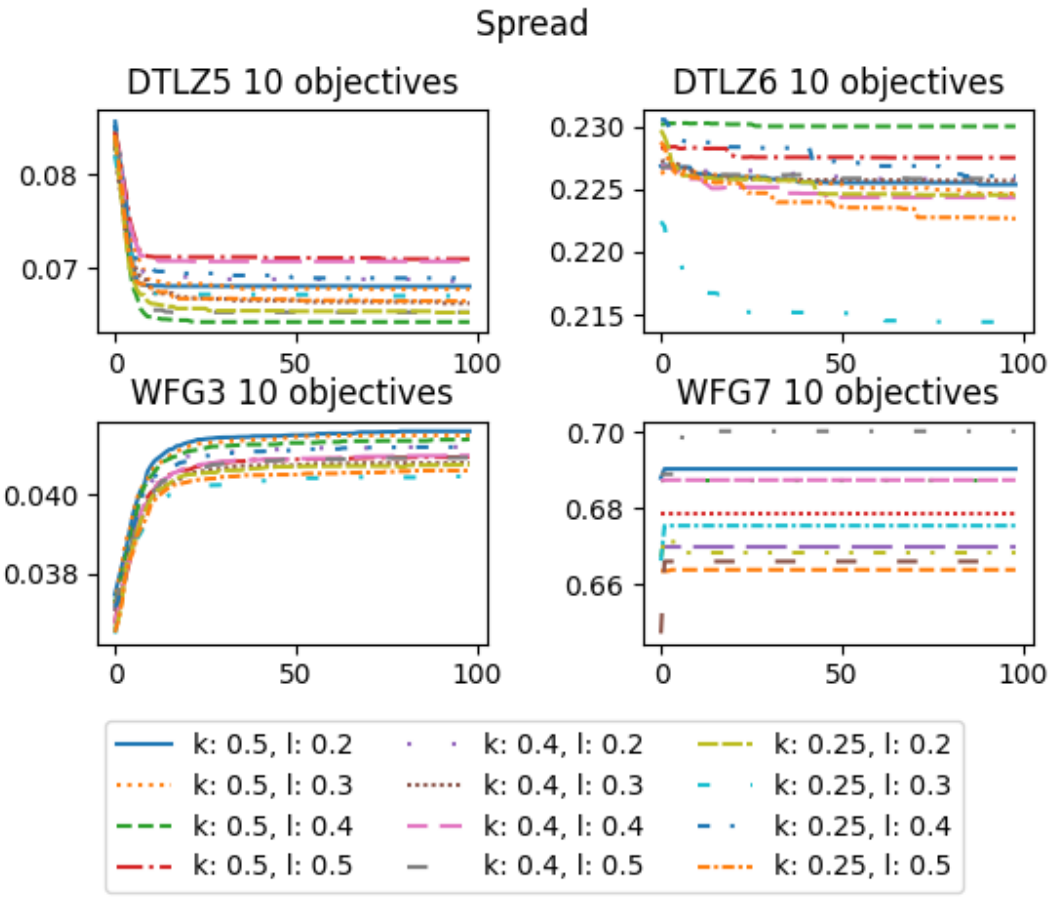


Figure 6.5: Ten-objective S results for OAM with different k and l parameter values.

only have a small influence when using an external archive. Generally, we can see that the "best" results were achieved when retaining a larger percentage of the found non-dominated solutions (i.e., a large k value).

6.4.2 Environmental Selection Results

In this section we considered the solution set reduction aspect of OAM compared to the Environmental Selection (ES) [142]. In these experiments we performed reduction using both OAM and ES on the same non-dominated solution set generated by NSGA2. We applied OAM in two different ways (see Section 6.2):

Problem	M	k	ℓ
DTLZ5	5	0.40	0.30
	10	0.40	0.50
DTLZ6	5	0.50	0.40
	10	0.50	0.20
WFG3	5	0.20	0.40
	10	0.50	0.40
WFG7	5	0.50	0.20
	10	0.50	0.40

Table 6.1: Chosen parameter combinations (k and l) for each problem.

1. E-OAM: We updated the Objective Archive after each generation of NSGA2 and obtained the final solution set by finding the overlapping solutions of the external archive.

2. S-OAM: We used the same parameters k and ℓ and generated an Objective Archive based on the final non-dominated solution set generated by NSGA2. We then found the overlapping solutions in the resulting archive.

We used the generated OA to find overlapping solutions to determine the reduced solution set. We looked at the number of solutions generated by both E-OAM and S-OAM with overlap equal to 60% and 80% of the number of objectives. The resulting solution set sizes, as well NSGA2's solution set size, are shown in Table 6.2. Since using 80% overlap results in empty solution sets for some of the problems, we decided to use 60% overlap to generate the final non-dominated solution sets.

We applied ES to both the solutions found in the complete OA generated by E-OAM before finding overlap and to the non-dominated solution set generated by NSGA2. We set

		NSGA2	E-OAM		S-OAM	
Problem	M		60%	80%	60%	80%
DTLZ5	5	931	46	0	144	35
	10	887	100	31	354	111
DTLZ6	5	654	140	3	381	129
	10	776	123	53	291	127
WFG3	5	643	36	0	49	49
	10	837	47	33	478	297
WFG7	5	995	61	2	326	75
	10	1000	474	119	285	251

Table 6.2: Average solution set size for NSGA2, E-OAM, and S-OAM with different overlap sizes (indicated by the percentages).

the number of the selected solutions to be the same as the number created by the OAM overlap. This means the number of solutions selected from the OA was the same as the number of solutions generated by the E-OAM overlap, and the number of solutions selected from NSGA2's solution set was the same as those from S-OAM. Therefore, we denote the two different ES-based selections as ES-E and ES-S respectively. The choice of parameters k and ℓ (Table 6.1) was based on the results from the previous section (Figures 6.2 – 6.5). We report HV and S for the original non-dominated solution set from NSGA2 as well as for the different implementations of OAM and ES. We compared the quality of the selected solutions through HV (Table 6.3), S (Table 6.4), and solution visualizations for the four reduced solution sets (Figures 6.6 – 6.9).

E-OAM not only reduced the solution set to a more manageable size but improved HV and S for most problems as compared to NSGA2's non-dominated solution set. Considering classic NSGA2 does not use any archive management strategy, this makes sense since we are

Problem	M	NSGA2	E-OAM	ES-E	S-OAM	ES-S
DTLZ5	5	0.988	**0.997**	0.985	0.987	0.985
	10	0.985	**0.998**	0.980	0.982	0.980
DTLZ6	5	0.912	**0.968**	0.879	0.912	0.880
	10	0.915	0.920	0.880	0.914	0.881
WFG3	5	0.757	0.758	0.756	0.757	0.756
	10	0.066	0.066	0.060	0.066	0.060
WFG7	5	0.106	**0.201**	0.093	0.105	0.093
	10	0.062	**0.190**	**0.144**	0.062	0.049

Table 6.3: Hypervolume for NSGA2, OAM, and ES. Bold indicates statistical significance with $\alpha = 0.05$.

Problem	M	NSGA2	E-OAM	ES-E	S-OAM	ES-S
DTLZ5	5	0.005	**0.0151**	0.006	0.003	0.001
	10	0.006	**0.034**	0.007	0.004	0.000
DTLZ6	5	0.058	**0.117**	0.019	0.058	0.014
	10	0.049	0.048	0.009	0.048	0.002
WFG3	5	**0.011**	0.006	0.007	0.006	0.000
	10	0.006	**0.270**	**0.190**	0.006	0.001
WFG7	5	0.047	**0.346**	**0.212**	0.046	0.001
	10	0.077	**0.623**	**0.450**	0.077	0.002

Table 6.4: Spread for NSGA2, OAM, and ES. Bold indicates statistical significance with $\alpha = 0.05$.

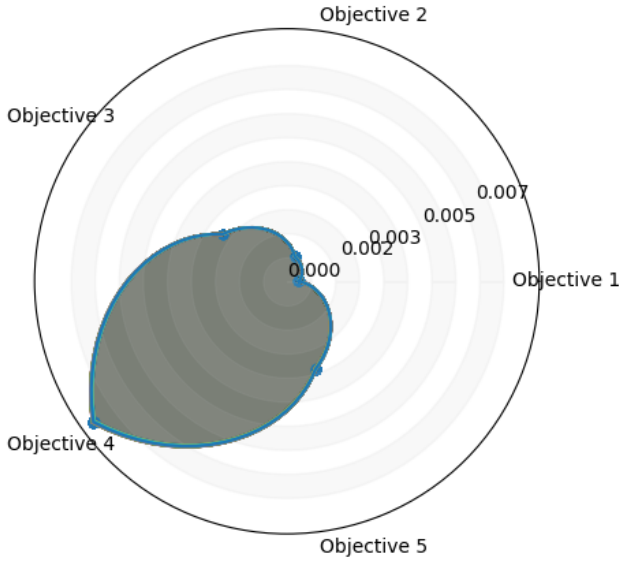
keeping track of all non-dominated solutions found when using the E-OAM approach. The interesting part is that the reduced solution sets still improved HV and S for most problems,

compared to NSGA2, which only has significantly better results for S on the five-objective WFG3.

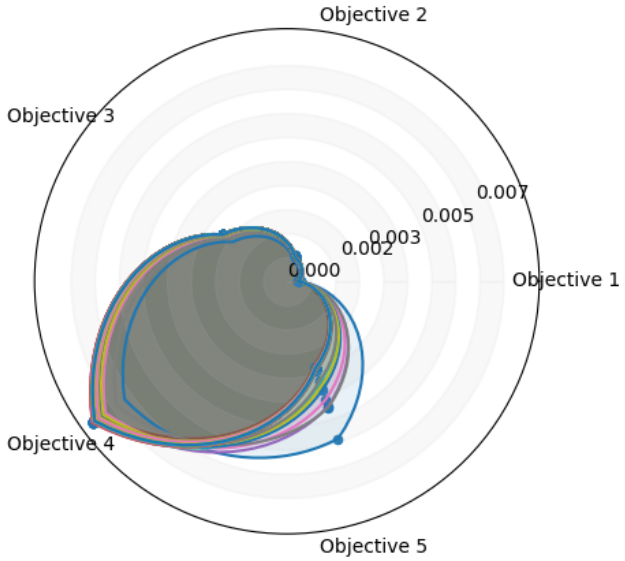
When comparing ES to OAM, we see that for both the single use and external archive approach, OAM has better performance than ES on all problems. In order to evaluate what this means for the final non-dominated solution set, we used radar charts to visualize the four different reduced solutions sets from a single, randomly selected, run of NSGA2 on five- and ten-objective DTLZ5 (Figures 6.6 – 6.9). We created plots for all four problems, but are only showing results for DTLZ5 in this chapter, the remainder of the plots can be found in Appendix C.

Each radar chart shows a single solution set, where the sets contain approximately 45 and 140 solutions for five-objective DTLZ5 and 100 to 200 solutions for ten-objective DTLZ5. Each solution within a solution set has a different line color. The center of the radar is the origin of the graph and is equal to 0, which would be the ideal score for the normalized objectives based on minimization. As we move toward the edge of the radar, the value increases, which indicates a worse objective score, as indicated by the numbers on the plot. Note that the range of numbers changes between plots, i.e., the maximum objective values vary across plots. The objectives are spaced evenly along the perimeter of the circle; the exact values for each objective are indicated using the blue dots, where a dot close to the center of the graph indicates a more desirable objective score.

In these plots, we can see that the solutions selected from NSGA2's final solution set have lower objective scores overall, but there is not much diversity in the selected solutions, especially when using ES. In the five-objective case, most solutions chosen from the single run have a higher value for objective four, whereas the external archive solutions also include solutions that have a lower value for objective four with a higher value for objective five. Similarly, for ten-objectives, ES-S, ES-E, and S-OAM resulted in solutions with worse values for objectives eight and nine, with little variety in the chosen solutions. E-OAM, on the other

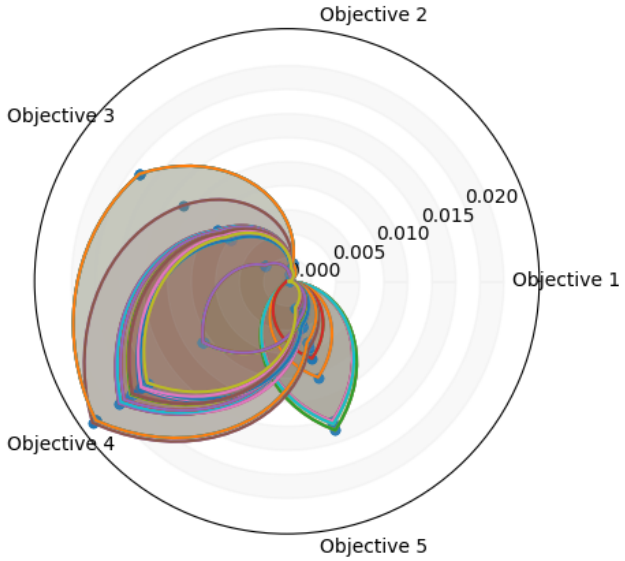


(a) NSGA2 ES-S

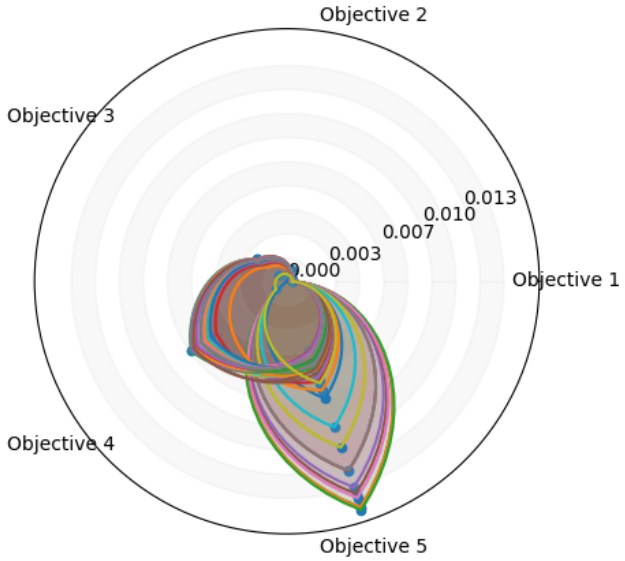


(b) NSGA2 S-OAM

Figure 6.6: DTLZ5 five objectives NSGA2 single run.

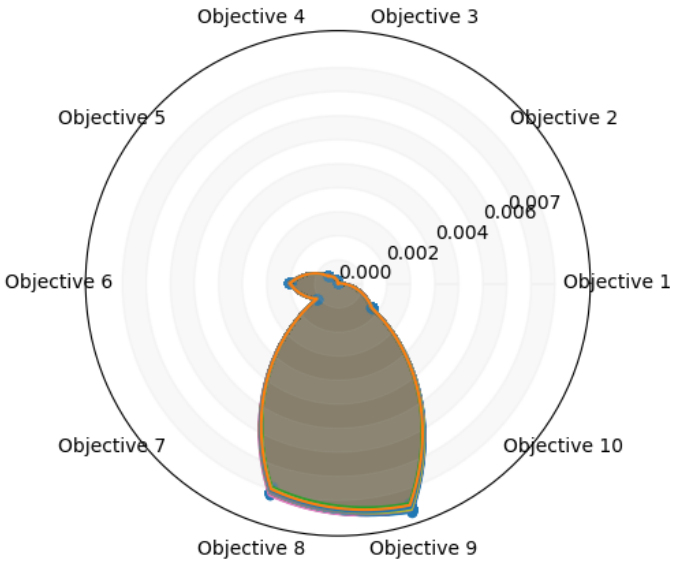


(a) NSGA2 ES-E

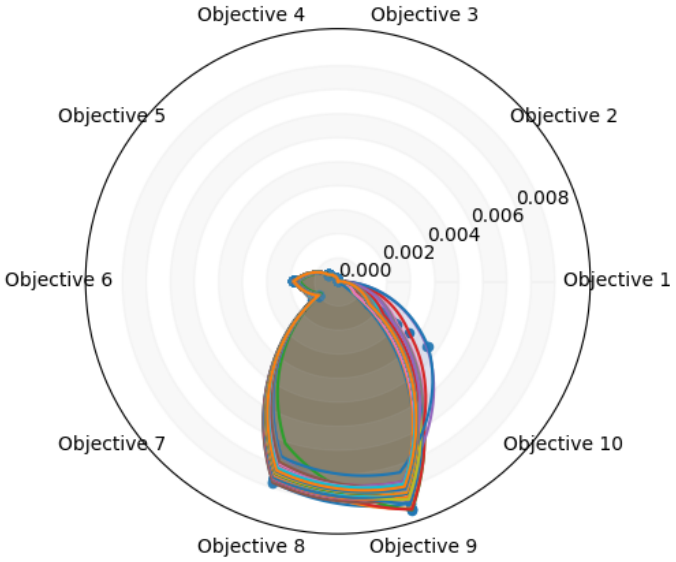


(b) NSGA2 E-OAM

Figure 6.7: DTLZ5 five objectives NSGA2 external archive.

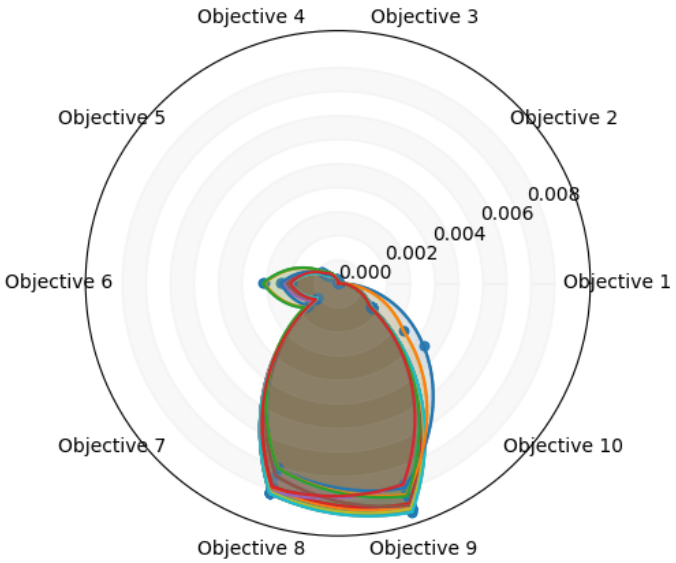


(a) NSGA2 ES-S

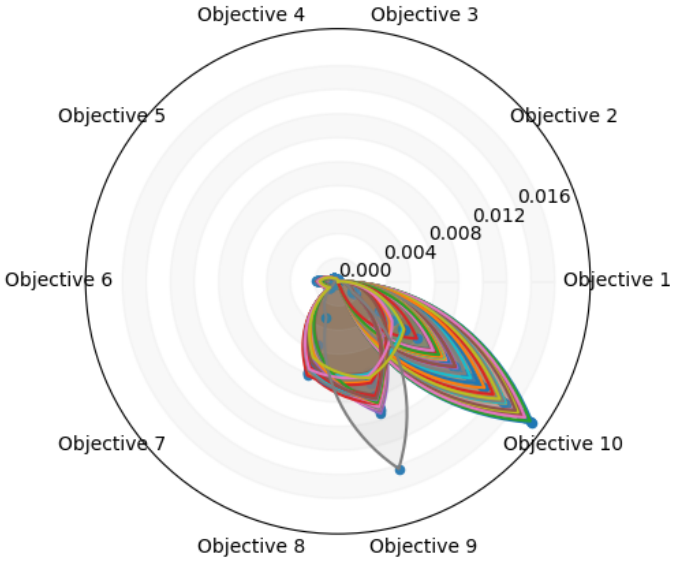


(b) NSGA2 S-OAM

Figure 6.8: DTLZ5 ten objectives NSGA2 single run.



(a) NSGA2 ES-E



(b) NSGA2 E-OAM

Figure 6.9: DTLZ5 ten objectives NSGA2 external archive.

hand, provided solutions that improve on objectives eight and nine at the cost of objective ten, delivering more diverse solutions to choose from. This opens up more choices if the end-user decides they care more about one objective or the other.

6.4.3 NSGA3 Results

Our final experiments considered two different aspects of the algorithm as compared to and applied to NSGA3:

1. We compared the final results from NSGA3 to the final archive found by E-OAM with NSGA2. In this way, we can evaluate how using an external archive updated with OAM compares to using reference vectors for reducing the number of solutions.

2. We applied S-OAM to the final results set found by NSGA3 to further demonstrate its applicability to reducing the size of any solution set.

Since NSGA3 relies on reference vectors to guide the algorithm through the search space, we ran experiments with different numbers of partitions using the Das-Dennis approach to generate different sized sets of reference vectors [29]. We tested three, four, six, and nine reference vectors (Table 6.5). The population size was kept at 1000. This means that for the ten objective problems, the number of reference vectors was larger than the population size, which could affect NSGA3's performance. Based on these results, we used four partitions to generate the reference vectors for all problems since it performs well on HV and S and creates smaller sized solution sets.

6.4.3.1 External Archive Solutions

This section evaluated using OAM as an external archive throughout the evolutionary process and compared the final non-dominated overlapping solution set created by OAM-NSGA2 to the non-dominated solutions found by NSGA3. In addition to HV and S (Table 6.7), we calculated adjusted coverage as presented in Chapter 3. Since we did not change the NSGA2 process, it came as no surprise that NSGA3 covered

		# Part.	3	4	6	9
DTLZ5	5	Size	12	26.5	45	131
		HV	0.999	0.999	0.999	0.999
		Spread	0.023	0.028	0.027	0.030
	10	Size	126	265	506	1236
		HV	0.999	0.999	0.999	0.995
		Spread	0.052	0.053	0.049	0.067
DTLZ6	5	Size	35	70	210	680
		HV	0.999	0.999	0.999	0.999
		Spread	0.141	0.143	0.143	0.149
	10	Size	220	714	1435	1649
		HV	0.999	0.999	0.998	0.997
		Spread	0.230	0.231	0.243	0.243
WFG3	5	Size	31	53	79	174
		HV	0.863	0.869	0.897	0.892
		Spread	0.457	0.542	0.468	0.533
	10	Size	42	75	173	653
		HV	0.129	0.188	0.270	0.136
		Spread	0.738	0.887	1.822	0.891
WFG7	5	Size	35	70	210	710
		HV	0.557	0.611	0.654	0.662
		Spread	1.859	1.86	1.862	1.871
	10	Size	220	715	1371	1622
		HV	0.410	0.466	0.413	0.368
		Spread	3.844	3.856	3.753	3.504

Table 6.5: NSGA3 partitioning results.

			NSGA3	E-OAM
DTLZ5	5	*AC*	60.80%	39.20%
		Size	25	38
	10	*AC*	77%	23%
		Size	266	90
DTLZ6	5	*AC*	85.50%	14.50%
		Size	70	130
	10	*AC*	83%	17%
		Size	714	160
WFG3	5	*AC*	100%	0%
		Size	80	34
	10	*AC*	99%	1%
		Size	188	48
WFG7	5	*AC*	100%	0%
		Size	70	64
	10	*AC*	100%	0%
		Size	715	400

Table 6.6: Adjusted coverage (AC) and solution set size for NSGA3 and E-OAM-NSGA2.

most or all of the non-dominated solutions found by NSGA2 (Table 6.6). Using OAM as an external archive alone has no influence on algorithm performance, so if NSGA2 does not perform well, this directly influenced the solutions kept in the OA. When considering the number of solutions found, E-OAM and NSGA3 were similar for the five-objective problem, but E-OAM consistently resulted in smaller solution sets for the ten-objective problems.

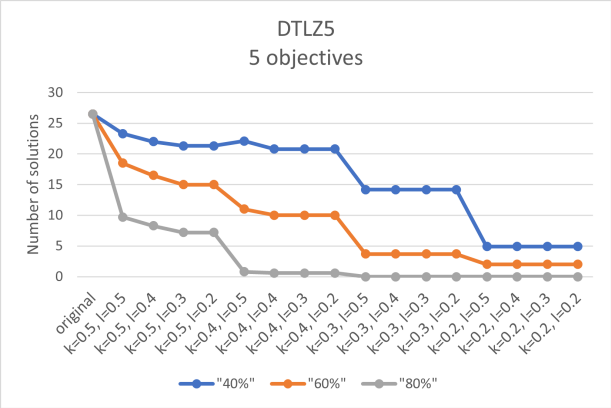
Interestingly, when considering HV, we found no significant difference for the DTLZ

		NSGA3	
		HV	S
DTLZ5	5	0.999	0.028
	10	0.999	0.053
DTLZ6	5	0.999	0.143
	10	0.999	0.231
WFG3	5	0.897	0.542
	10	0.270	1.822
WFG7	5	0.611	1.861
	10	0.466	3.856

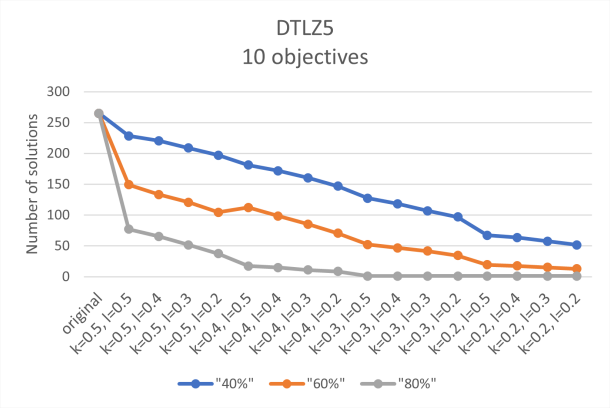
Table 6.7: Hypervolume and spread for NSGA3.

problems, but NSGA3 did significantly improve performance for both the five- and ten-objective WFG problems (as compared to the E-OAM NSGA2 results reported in Table 6.3). This was not a surprise, considering previous research comparing NSGA3 and NSGA2 in many-objective spaces unanimously show that NSGA3 improves on NSGA2 [26, 34]. However, given this information, it is interesting to note that E-OAM-NSGA2 performed reasonably well on the DTLZ problems. According to Huband et al., DTLZ5 and DTLZ6 are supposed to represent degenerate Pareto fronts [65], but this attribute no longer holds true when the number of objectives is of size four or more. This could explain why a more complex algorithm no longer adds much benefit.

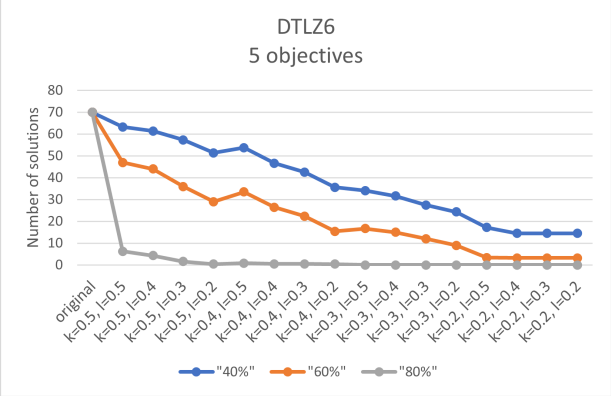
When comparing the spread (S) results, we found that NSGA3 had significantly better S for all problems. This makes sense when considering the use of reference vectors. The use of such pre-defined directions means the algorithm is spreading out more evenly across the search space, as compared to NSGA2, which uses no such guidance.



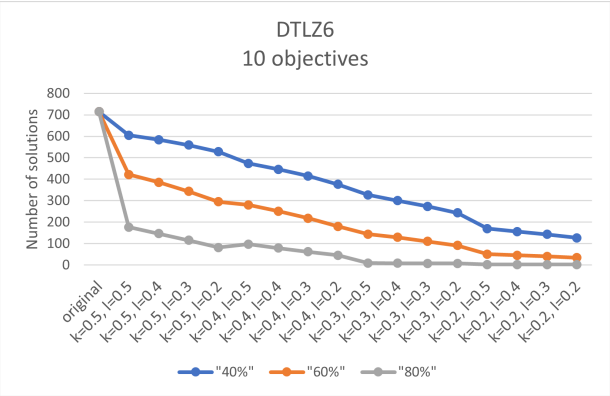
(a) DTLZ5 five objectives.



(b) DTLZ5 ten objectives.



(c) DTLZ6 five objectives.

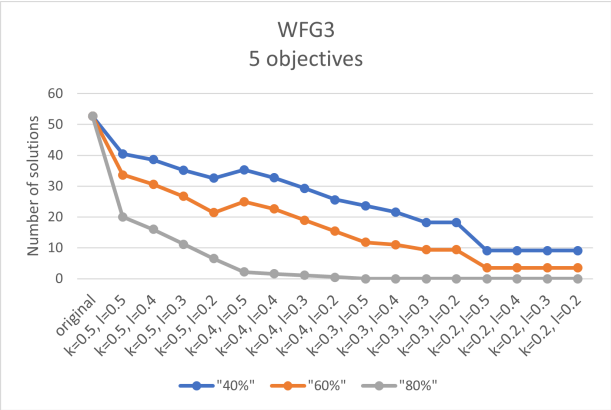


(d) DTLZ6 ten objectives.

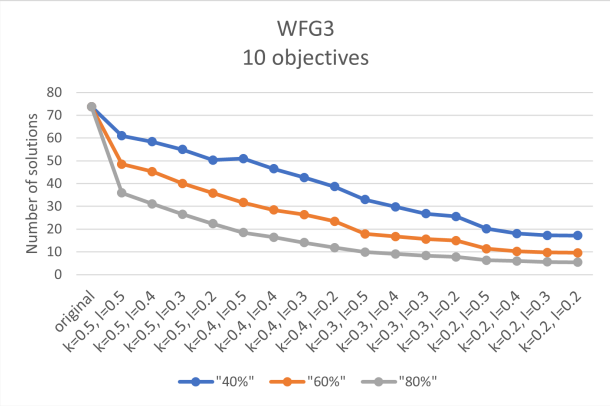
Figure 6.10: Reduction size of NSGA3 solution sets for the DTLZ problems using different k and l parameters with varying levels of overlap (40%, 60%, and 80%).

6.4.3.2 Direct Solution Set Reduction In addition to the comparative analysis, we applied S-OAM to the results found by NSGA3 to show its general applicability and to further investigate the type of results that are selected when applying S-OAM. We investigated the influence of the k and l parameters, as well as the amount of overlap, on the solution set size (Figures 6.10 and 6.11).

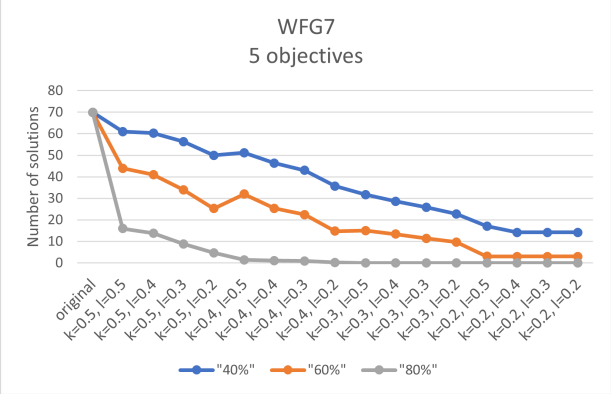
As expected, Figures 6.10 and 6.11 show that the number of chosen solutions gradually decreases as the parameters k and l get smaller for the ten objective problems. However, in the five objective case, we do see stagnation early on when looking at 80% overlap. As k



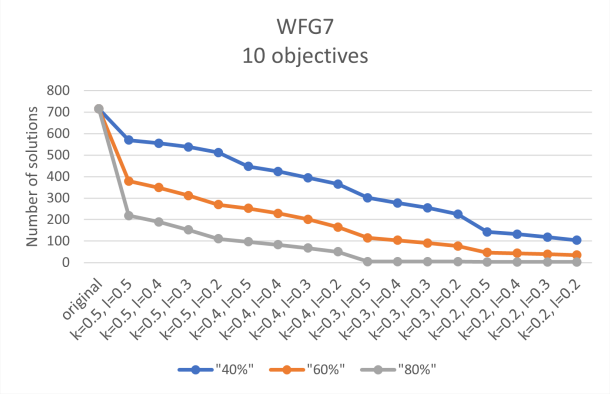
(a) WFG3 five objectives.



(b) WFG3 ten objectives.



(c) WFG7 five objectives.



(d) WFG7 ten objectives.

Figure 6.11: Reduction size of NSGA3 solution sets for the WFG problems using different k and l parameters with varying levels of overlap (40%, 60%, and 80%).

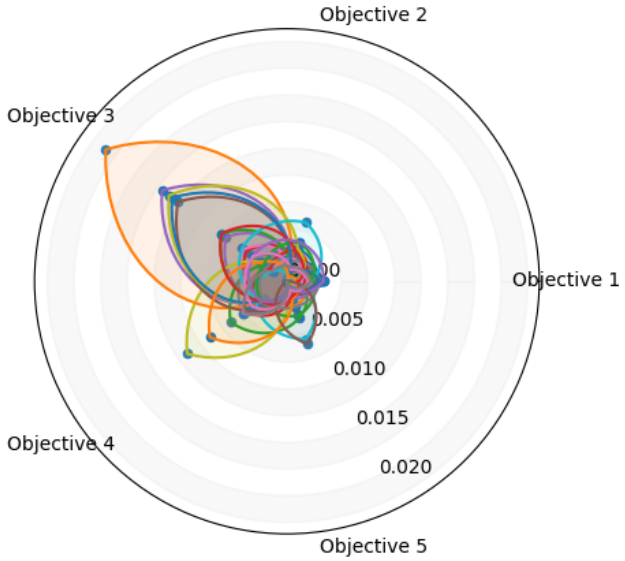
and l decrease, fewer solutions are being put in each objective archive. If we have a smaller solution set to start with, as is the case for the five objective problems, applying a large overlap parameter can lead to an empty solution set. For the ten objective instances, using 80% overlap still returns one to five solutions when both k and l are set to the smallest value of 0.2. For DTLZ5 with five objectives, we see large jumps in size occurring when the value for the k parameter decreases. We believe this is due to the small starting size of the archive (on average 26.5 solutions are generated by NSGA3).

We created radar plots to visualize the selected solutions (Figures 6.12 – 6.15). We

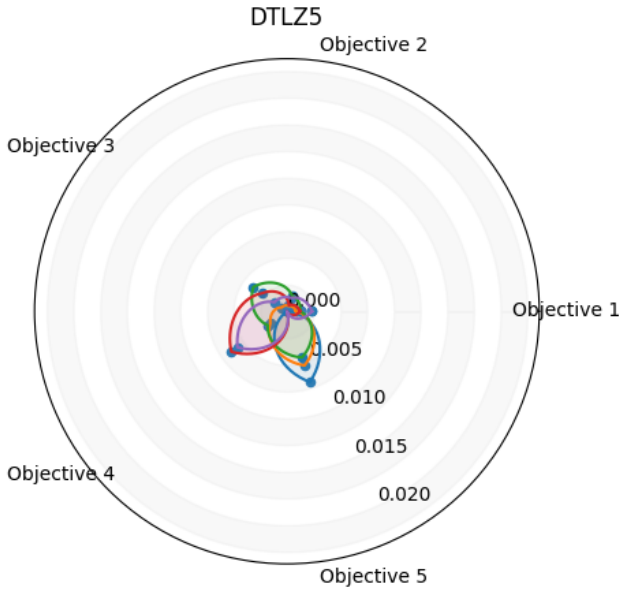
used 60% overlap and set parameters $k = 0.3$ and $l = 0.5$ for 5 objectives, selecting $15 - 20\%$ of the total solutions, and $k = 0.2$ and $l = 0.5$ for 10 objectives, selecting $7 - 15\%$ of the total solutions. Once again, we created plots for all four problems, but results are only shown for DTLZ5 and WFG3. The other plots can be found in Appendix C. We chose these problems to visualize because we used DTLZ5 as the example when comparing OAM to ES, and because WFG3 is one of the problems for which NSGA3 does significantly better in terms of HV.

When considering the solution sets for both DTLZ5 and WFG3, we see that the full solution set includes solutions with a high objective score for some of the objectives. For all problems, OAM selects solutions that maintain the diversity of the NSGA3 solution set but removes solutions that are very similar or that have worse objective scores without a strong positive impact on the other objectives. For example, when we look at WFG3, the starting solution set as found by NSGA3 explores solutions with a trade-off between the first two objectives in the five objective case (Figure 6.14) and the first six objective in the ten objective case (Figure 6.15). The selected OAM solutions still offer those trade-off options but removed the solutions with the worst fitness functions. This results in solutions closer to the center of the radar plot, which is more desirable.

Finally, when we consider the solution set generated by E-OAM for DTLZ5 (Figures 6.7 and 6.9) and compare it to the solutions found by NSGA3, we see that the E-OAM solutions appear to be a subset of the NSGA3 solutions. Given what we know in terms of HV and S for these results, this makes sense. E-OAM-NSGA2 generated less diverse solutions, but the found solutions are similar in quality to the solutions in NSGA3 that are closer to the center of the radar charts. This explains the similar HV scores for the two algorithms. In other words, even though NSGA3 resulted in more diversity, this is likely because NSGA3 was keeping solutions with a relatively large increase in one objective score to gain a small decrease in another objective score. When applying OAM to the NSGA3

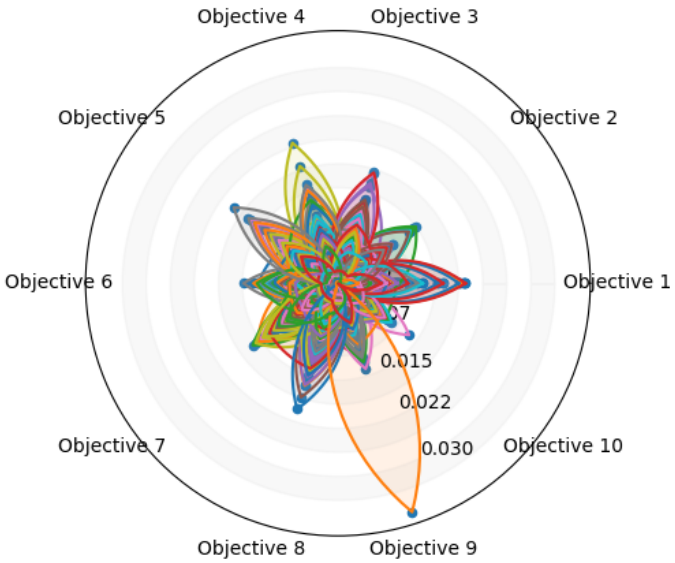


(a) NSGA3

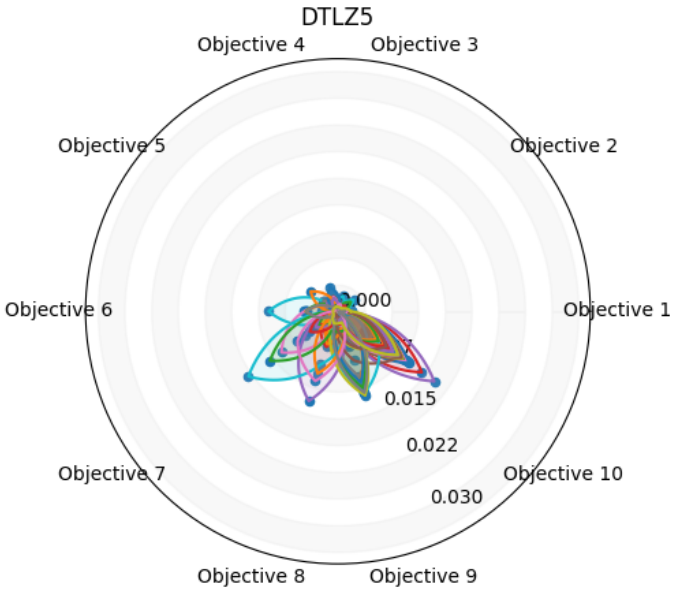


(b) NSGA3 OAM

Figure 6.12: DTLZ5 five objectives NSGA3

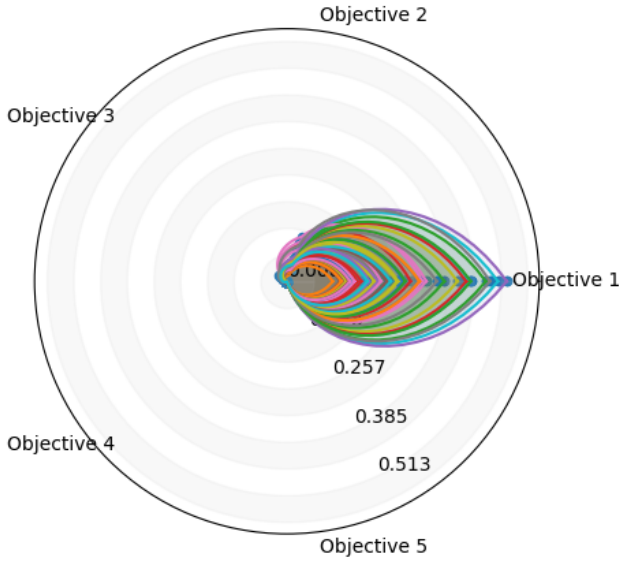


(a) NSGA3

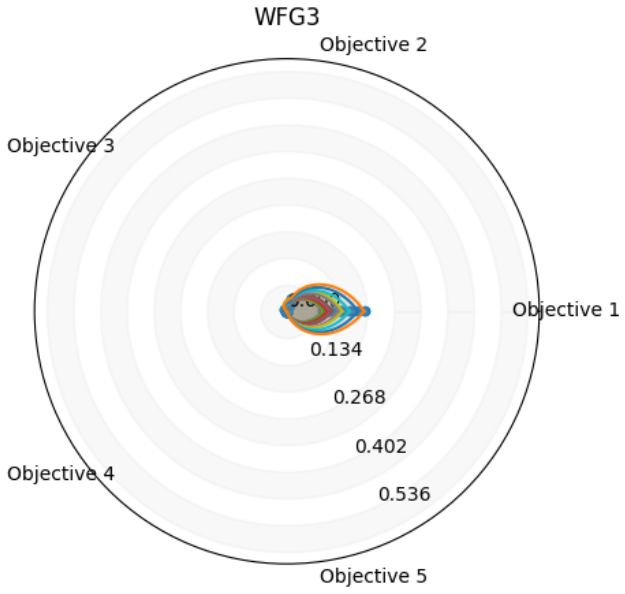


(b) NSGA3 OAM

Figure 6.13: DTLZ5 ten objectives NSGA3

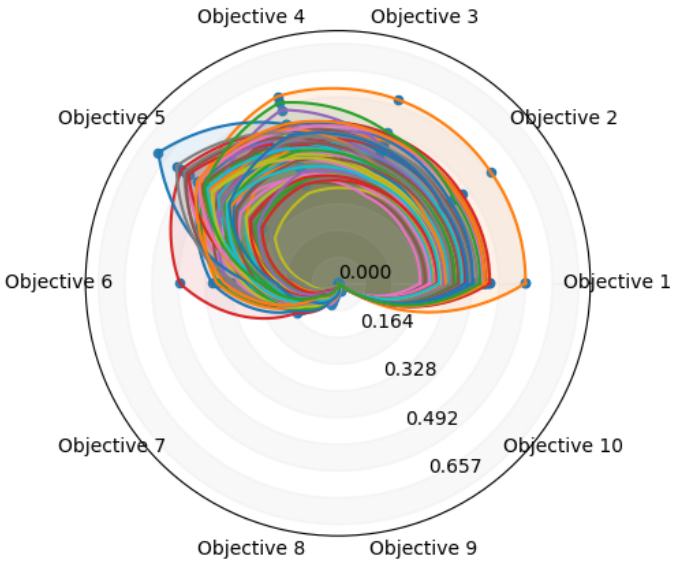


(a) NSGA3

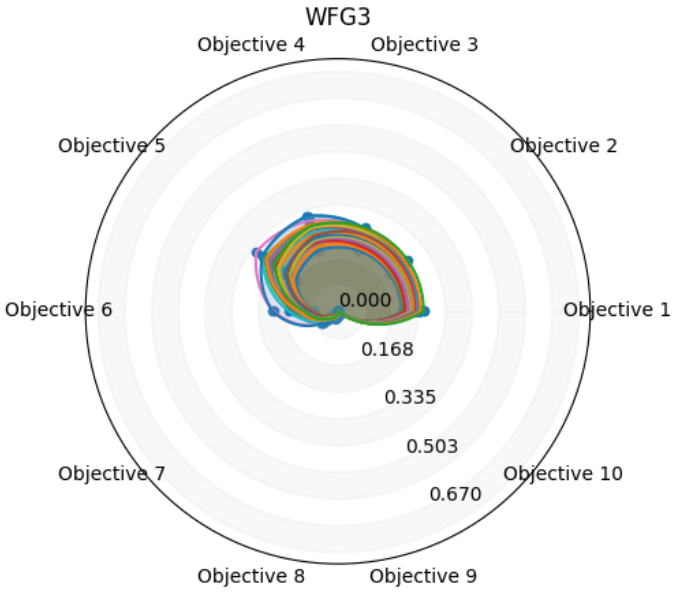


(b) NSGA3 OAM

Figure 6.14: WFG3 five objectives NSGA3



(a) NSGA3



(b) NSGA3 OAM

Figure 6.15: WFG3 ten objectives NSGA3

solution set, it removes many of these solutions while maintaining the increased diversity in the NSGA3 solution set. What this indicates is that applying OAM resulted in selecting solutions that perform well on all objectives given the original solution set. In other words, if the original algorithm generates a diverse set of solutions, OAM is able to reduce this solution set successfully to a smaller size while maintaining that diversity.

6.5 Discussion

In our experiments, we found that the OAM approach found more diverse solutions than Environmental Selection [142]. Furthermore, the external archive approach improved upon NSGA2's final non-dominated solution set, yielding similar quality results to NSGA3 on problems DTLZ5 and DTLZ6. However, NSGA3 still had better performance on the WFG3 and WFG7 problems. When we applied the OAM approach to the final solution set found by NSGA3, OAM selected diverse solutions but with better overall fitness across the selected solutions. In the five objective case, the solution sets were reduced from $60 - 100$ solutions to $5 - 20$ solutions. When reducing the 10-objective solution set, the number of solutions went from a range of $300-500$ to $15-25$. Through visual assessment, we found that the selected solutions struck a good balance between the different objectives as compared to the full solution set. Lastly, we would like to note that the current version of the algorithm allows the return of an empty solution set (as discussed in Section 6.4.3.2). This is a problem that needs to be addressed, for example, by including an archive weighting technique and returning the solutions with the highest weight if no solutions are being selected through overlap. Overall, we conclude that using OAM for solution set size reduction performs well regardless of which algorithm is used to create a non-dominated solution set.

6.6 Concluding Remarks

We introduced the Objective Archive Management (OAM) strategy to create a reduced final solution to many objective optimization problems. OAM creates an archive for each objective, effectively distributing the solutions into separate archives. Each of these objective archives maintains the top $k\%$ solutions for the relevant objective. To maintain diversity, the objective archives also included a diversity management technique that takes both variable and objective space diversity into account. Once the Objective Archive is created, we count how many times a solution occurs across the objective archives and add solutions that occur in a minimum number of archives to the final, reduced solution set. We found that our approach successfully reduced large solution sets to a more manageable size while maintaining desirable properties. Our approach has several benefits compared to existing approaches: it requires no pre-defined reference vectors, it can be applied to any algorithm or any solution set, the end-user does not need MOEA specific knowledge, and it is easy to adjust the diversity-convergence trade-off.

CHAPTER SEVEN

REAL WORLD APPLICATION - PRECISION AGRICULTURE

In addition to the benchmarks we investigated, we applied MOFEA to generate fertilizer prescription maps as part of the On-Farm Precision Experimentation and Data-Intensive Farm Management projects. These projects aim to assist farmers in their decision making process by analysing specific fields to improve production. This chapter will go over the general project structure and goals before showing how we used population-based algorithms and different multi-objective optimization approaches to find experimental and optimal prescription maps.

7.1 On-Farm Precision Experimentation

Precision Agriculture (PA) is an interdisciplinary field found at the intersection of agriculture and technology. It is a rapidly growing research area that uses advanced technologies to improve all aspects of agriculture. Commonly researched problems include smart irrigation [52], crop monitoring through wireless sensor networks [130], remote sensing for assessing crop condition and predicting yield response [99], and input optimization (e.g., fertilizer, herbicide, pesticide, or irrigation). The PA field gained traction in the 1980s with the advent of the Global Positioning System (GPS), enabling machines to apply treatments with requirements localized to each field [116]. This led to the creation and use of Variable Rate Technology (VRT): machines that are able to switch the amount of fertilizer being applied as they move across a field, known as Variable Rate Application (VRA). VRA decreases the amount of fertilizer applied to a specific field while increasing profitability, effectively also reducing environmental impact [79, 121]. VRA tries to determine the rate of fertilizer to apply to different parts of a field based on a variety of factors, such as

precipitation, elevation, and previous years' yield [101].

7.1.1 Data-Intensive Farm Management

The Data-Intensive Farm Management (DIFM) [13] initiative by the US Department of Agriculture (USDA) is used as a basis for our study. DIFM uses farm and field-specific data to optimize different aspects of farming to increase profit by gathering data from the farmers as well as publicly available data, such as precipitation and satellite imagery. Figure 7.1 shows the overall workflow of the DIFM project. "Field information" refers to the physical data we gather from the farms, which is put into a database. "Data organization & analysis" refers to the upkeep of the database containing farm and field data as well as any analysis that is performed directly on the stored data. We use the field-specific data to train a yield prediction model in order to find accurate yield response curves. The yield response can then be used to create optimal prescription maps ("Field profit maximization"). Currently, the prescription maps we create prescribe fertilizer or seeding rate, but the process could be used to create different types of prescription maps. After the farmers apply the prescribed fertilizer, we gather more data from the field from the subsequent harvest ("Yield, Protein, Net return") and add this data back into the database. To create the best possible prescription maps, we need as much field-specific data as possible. Only creating optimal maps could limit the application rates; therefore, to analyse how a field responds to different rates and gather more field data, we create experimental prescription maps. These experimental trials are created based on previous years' yield, and in some cases crop quality (e.g. protein), levels. Figure 7.2 shows an example of a prescription map as used by the DIFM project [13]. Experimental fertilizer rates are randomized and stratified across the field to create a spatial and temporal database on how different parts of the field react to different fertilizer rates over time ("Parameterization"). The gathered data is analysed in order to find improved yield response curves. This creates a continuous feedback system,

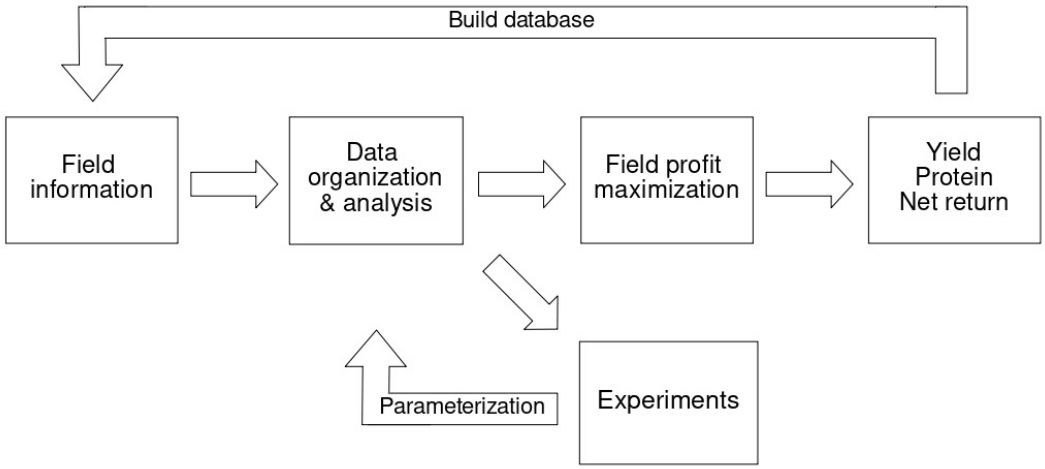


Figure 7.1: DIFM process for field profit maximization.

where the gathered data for each field keeps growing, thus hopefully creating more accurate models.

7.1.2 Fertilizer Prescription Maps

In order to evaluate the applicability of MOCO algorithms on a real world problem, we considered the creation of fertilizer prescription maps for field-specific optimization. The project workflow indicates that there are two different types of prescription maps: experimental and optimal. The experimental prescriptions are combinatorial: we are looking to find the optimal combination of a predefined set of input rates to apply to a field, and these fertilizer rates are discrete values. In our case, each field is divided into strips based on the width of the farming equipment, and each of these strips is divided into cells or plots based on a length specification. Figure 7.2 shows an example of what this looks like. The length of the plots varies from field to field based on farmer preference, but usually ranges from 200 feet to 400 feet. Each plot in a field is prescribed a value out of a set of input values, for example, a rate of 20, 40, 80, or 120 pounds of Nitrogen per acre. We then try to spread these different rates across the plots to gather as much useful data as

Figure 7.2: Example of a prescription map for experimental fertilizer application. Different colors represent different fertilizer rates.

possible from a field by measuring subsequent yield and crop quality. On the other hand, an optimized map prescribes inputs (e.g., fertilizer) that results in an optimized net return for the farmer. These prescribed values can be discrete if there is a set of values the prescription should use. For example, if the farmer only wishes to apply rates of Nitrogen of certain values (similar to the experimental prescriptions), this would result in a combinatorial optimization problem. However, if there are no such predefined values, the problem becomes a continuous optimization problem.

7.2 Related Work and Motivation

In recent years, MOO has been applied to four different aspects of PA: wireless sensor networks [165], pesticide application [173], irrigation [87], and, lastly, fertilizer application, which is what our research focuses on. Wireless sensor networks connect different measuring instruments through a wireless network. These instruments are placed throughout fields to

gather pertinent data such as soil moisture, temperature, salinity, etc. The gathered data can then be used to inform the farmer of crop quality and health [71]. MOO can be applied to optimize communication between the nodes in the network, for example, to improve energy efficiency while avoiding congestion [165]. Zhai *et al.* look to optimize pesticide application of crops to minimize cost and maximize expected benefit [173]. Irrigation aims to address the problem of water scarcity in agriculture by designing improved irrigation water allocation schemes [87]. Its multi-objective nature is defined by simultaneously trying to optimize water productivity, allocation equity, profit, economic benefit, blue water utilization, and leakage loss.

The question of sustainability in agriculture has existed since the start of precision agriculture as an area of study [8]. But addressing these sustainability issues in practice has proven more difficult. VRA involves technology that allows farmers to apply different input rates to different parts of the field to control their production and reduce cost more precisely [121]. For example, using VRA, farmers are able to apply less fertilizer overall than if they were to apply a uniform rate across an entire field, thus also improving their sustainability practice [8].

Several studies have shown that VRA can help with sustainability [31, 158]; however, this is not always the case. Some studies found that using VRA increases the cost for the farmer [144]. To the best of our knowledge, only two studies have applied MOO algorithms to VRA prescription maps to provide a set of potential prescriptions. Zheng *et al.* apply a multi-objective fireworks optimization algorithm (MOFOA) for variable-rate fertilization [180]. Their goal was to find the optimal fertilization for oil crops based on yield, energy consumption, and spatial effects. The authors compare their algorithms to NSGA2, Pareto-Frontier Differential Evolution [2], Differential Evolution-Multi-Objective for Constrained Optimization [57], and Non-dominated Sorting Particle Swarm Optimization [88]. The authors found that MOFOA finds a better Pareto front approximation than the

other algorithms. It is noted that MOFOA requires more tuning to get optimal results and more function evaluations are involved for each generation.

The second study applying MOO to fertilizer application used an economic optimization model to determine the fitness of fertilizer prescription maps and irrigation strategies for optimal crop yield in western Switzerland [83]. This approach by Lehman *et al.* included an economic model and different levels of price risks. The authors also integrated climate change into their model to determine whether it has a negative impact on farming profit. Their results found that climate change does increase income risk for farmers. While climate change was used as a variable that influences farmer profit, the authors do not consider how fertilizer application affects climate change. Furthermore, the authors mention their use of a GA, but it is unclear how these different objectives are included in the GA.

7.3 Trial Design

A first problem that needed to be addressed in the Data-Intensive Farm Management project was the creation of experimental trial designs for farmers to apply to their field to gather information. By trial design, we refer to laying down experimental fertilizer rates on a field in such a way that the different amounts of fertilizer are distributed evenly across the field while balancing a farmers needs with creating an appropriate scientific experimental design. Therefore, we need to take two different objectives into account. First, to maximize the gathering of useful information, we want to create experimental prescription maps that stratify fertilizer rates across yield bins, as yield is one of the primary concerns for farmers. Additionally, when creating prescriptions for winter wheat, there is a potential protein premium if the wheat produced has a minimum percentage of protein; to this end, we can also create protein bins and stratify the fertilizer across both yield and protein data. Second, the farmers we work with in Montana want to minimize applicator rate jumps between consecutive plots to reduce strain on their application equipment. When applying

different fertilizer rates along a field, VRT is required. However, it takes a lot of effort for the VRT machinery to apply a large amount of fertilizer on one plot and then switch to a small amount of fertilizer for the next plot. Such big rate changes or jumps have the potential to wear out the equipment more rapidly, thus increasing maintenance or repair cost to the farmer. Eventually, we realized another interesting objective to include would be minimization of overall fertilizer rate to reduce the impact on the environment. To create these experimental maps, we started with a vanilla genetic algorithm using a weighted-sum bi-objective function, maximizing stratification and minimizing jumps. When we added in the third objective, we used NSGA2 as well as CCEA and FEA with NSGA2 as the base algorithm to run experiments. We wanted to get potential results across the Pareto front and evaluate how using subpopulations influences the results.

7.3.1 Trial Design Objective Functions

The experimental prescription map problem consists of optimizing the following objective functions:

1. Minimize jumps in consecutive cells
2. Maximize stratification across the field
3. Minimize overall fertilizer rate

Stratification and jump minimization were the first two objectives to be used in the weighted-sum Genetic Algorithm approach [113]. Adding in the third objective of overall fertilizer minimization required a more sophisticated trade-off, which is why we explored the use of NSGA2 and two different decomposition approaches to evaluate the three objective trial design prescription maps.

For a set of predefined fertilizer rates, the stratification strategy tries to ensure that each fertilizer rate is represented equally across k pre-determined yield bins. We considered

two different methods for discretization into bins: 1) by looking at the actual yield (yld) values (called equal width binning), or 2) by splitting on the data points themselves (called equal sample binning). The first method looks at the minimum and maximum yield and protein values and creates an even split of these values based on the desired number of bins. Without loss of generality, we consider yield. Then based on the number of bins k, we calculate an offset as

$$offset_k = \frac{1}{k}(max_{yld} - min_{yld}).$$

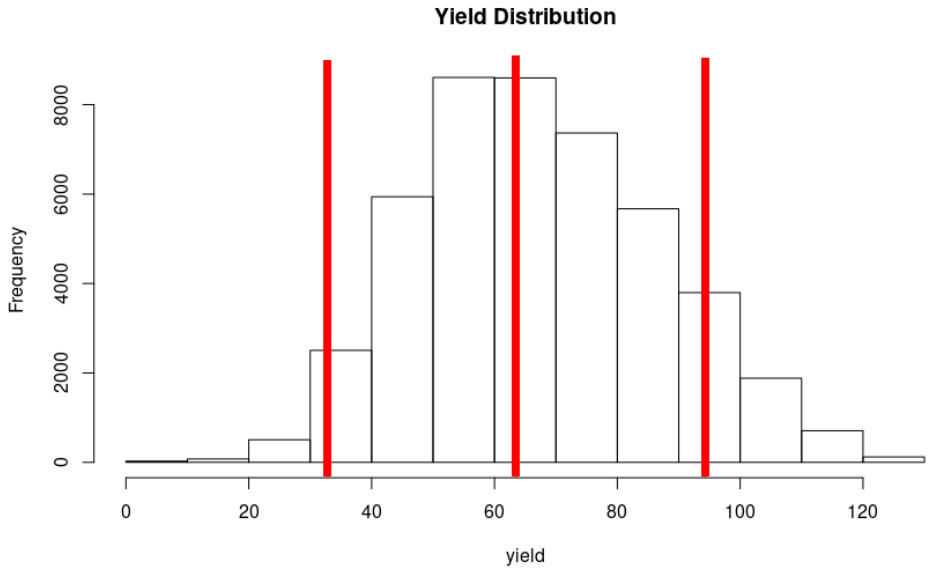
Thus, we get bin boundaries at

$$min_{yld}, \ldots, min_{yld} + j \cdot offset_k, \ldots, max_{yld}.$$

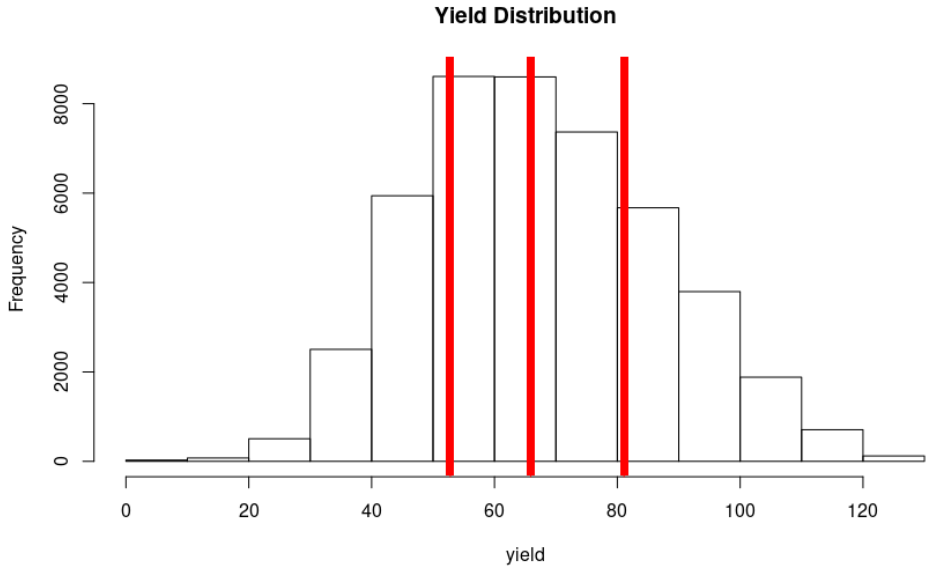
Equal sample binning, where we split on m data points, does not take the yield or protein values into account but aims to distribute an even number of points into each bin (i.e., m/k points). The differences in binning strategies are illustrated in Figure 7.3. Depending on the implemented bin discretization strategy, the stratification score calculation varies. When splitting on the actual data values, there will be an even distribution of cells in each of the bins. This means the target stratification will be the same for each bin combination. However, when splitting on the yield values to create bins, it is likely there will be fewer cells belonging to lower and higher bins due to the way the data is distributed, as can be seen in Figure 7.3a. In this case, the number of cells belonging to each bin has to be counted to determine how many cells each nitrogen rate should have for that specific bin.

Let l denote a specific yield bin, then $\#cells_l$ corresponds to the number of cells in the field that map to a specific bin. We then determine the target stratification as

$$tstrat_l = \#cells_l/k$$



(a) Equal Width Binning



(b) Equal Sample Binning

Figure 7.3: Example of different bin discretization types using a histogram representation of the yield values. The vertical red lines indicate bin boundaries using each discretization type.

since our goal is to distribute the fertilizer evenly over exactly k bins. The stratification score looks for an even distribution of fertilizer rates across cells belonging to the same bins k:

$$Fitness_{strat} = \frac{\sum_{l=1}^{k} |tstrat_l - astrat_l| - min_{strat}}{max_{strat} - min_{strat}},$$

where $tstrat_l$ is the target stratification and $astrat_l$ is the actual stratification of the same bin. The maximum stratification is determined by the worst case scenario, which occurs when all of the cells in the field have the same fertilizer rate. This is done to determine how many cells each fertilizer rate should have for that specific bin, since there could be an uneven number of cells. If there are sixteen cells and three bins, one bin will have 6 cells, whereas the others will have five. The actual stratification for a yield bin is calculated by counting the number each fertilizer rate occurs in each of the bins. For example, if we have 45 cells with three fertilizer rates and three total bins, we know the target stratification is five if each of the bins contains fifteen cells.

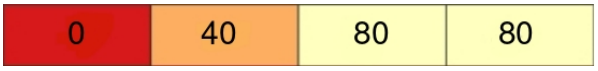
However, when creating prescriptions, large jumps in fertilizer rate between consecutive cells could occur. This puts strain on the farming equipment, increasing wear and tear on the equipment. In turn, this leads to the farmer having to replace equipment more frequently, increasing cost and waste, which has negative ecological impacts. To alleviate this issue, we incorporated a second objective to minimize jump magnitudes in the maps, resulting in a positive environmental impact, as illustrated in Figure 7.4.

The jump score sums over the absolute difference in fertilizer levels between adjacent cells determined by an "as-applied" map, which shows how the farmer applied fertilizer to the field, where each prescription map has c cells:

$$Fitness_{jumps} = \frac{\sum_{i=1}^{c-1} \Delta jumps_i}{max_{jumps}},$$

where $\Delta jumps_i = |F(map_i) - F(map_{i+1})|$, and $F(map_i)$ corresponds to the fertilizer index

(a) Example of consecutive cells with large jumps.



(b) Example of consecutive cells with small jumps.

Figure 7.4: Example of four consecutive cells in a field with large and small jumps. The values are in pounds of fertilizer/acre.

of cell i. If we apply the following rates $N = \{0, 40, 80, 120\}$, the fertilizer indeces would correspond to $F(0) = 0, F(40) = 1, F(80) = 2$, and $F(120) = 3$. Each individual jump score is then normalized to be within a [0,1] range using the worst case scenario where each consecutive cell goes from the minimum to the maximum fertilizer rate or vice versa (Figure 7.4a). If the unnormalized jump difference $\Delta jumps_i$ is less than or equal to 1, it is not added into the jump score, as this is the most desirable rate change between cells.

To mitigate the effect fertilizer has on the environment, we reduced the overall amount of fertilizer applied to a field, which served as our third objective when applying the Pareto-based approach. This should reduce pollution of the atmosphere by limiting greenhouse gas emissions and can help avoid polluting waterways, which can result in a loss of drinkable water and the death of aquatic life. The overall fertilizer rate was calculated by summing all the fertilizer prescribed in each of the cells and dividing by the maximum amount of fertilizer, $max_{fert} = \max(F) \times c$:

$$Fitness_{fert} = \sum_{i=1}^{c} F(map_i)/max_{fert}.$$

7.3.2 Genetic Algorithm and Weighted Sum

With the jump minimization and stratification fitnesses defined, we implemented the weighted sum method into a vanilla Genetic Algorithm since this was a fairly straightforward bi-objective problem. We introduced the representation of the prescription map as a chromosome for the GA. Then, we evaluated how adjusting the weights of the objectives influences the fitness scores. We also looked at two different mutation approaches, swap and scramble, to assess their influence on the convergence and diversity trade-off.

7.3.2.1 Experimental Approach

We hypothesize that we can apply a genetic algorithm to generate experiment prescriptions that effectively maintain stratification and minimize jumps in fertilizer rate application. In this study, we tested this hypothesis by considering a variety of genetic operators and by examining the effects of these operators on overall fitness as well as individual impact on stratification and smoothness.

Specifically, we applied a GA to optimize a fertilizer experiment prescription map, which dictates the fertilizer application rate for each cell on a field. In the case of winter wheat, we prescribed nitrogen in pounds/acre, and the farmers decided which k nitrogen rates they wished to apply and how the field was to be subdivided. We used three different farming fields: Sec35Mid, Davidsonmidwest, and Sre1314. The ultimate goal was to minimize jumps (Figure 7.4) while maintaining stratification.

For our GA, we used completed prescription maps as individual chromosomes in our population with each cell in the map as a gene and its corresponding nitrogen rate as the gene's allele, where the cells are ordered based on the as-applied map. Each nitrogen rate maps to an index that was used to calculate the jump score. A bi-objective fitness function was then applied, taking both jumps and stratification into account. The initial population consisted of n prescription maps, where each map is a chromosome such that its c cells are genes in the chromosome. Once the population has been generated and evaluated,

Parameter	Pop	OS	CR	MR	TS
Value	400	40	0.9	0.1	3

Table 7.1: Chosen values for all hyper parameters. The parameters are population (Pop), offspring created (OS), crossover rate (CR), mutation rate (MR), and tournament size (TS).

tournament selection was performed to identify two parents, choosing a predefined number of pairs by selecting the best map (lowest fitness score) from a chosen number of individuals from the current population. For each of these pairs, two-point crossover was performed by randomly selecting two indices and swapping the cells between these indices to create two new child prescription maps.

Finally, mutation was applied to the offspring to maintain diversity in the population. Two mutation operators were considered. Swap mutation chooses two random indices and switches the values of these two cells, and scramble mutation is performed by selecting all cells between two randomly chosen cells and performing a random permutation. These new maps replaced the maps in the original population with the worst fitness score.

There are several parameters that can influence the performance of the GA: the population size, the number of offspring to create, the number of candidates in tournament selection, and the crossover and mutation rates. The mutation rate (0.05, 0.10, and 0.15) and crossover rate (0.90, 0.92, 0.95, 0.98) were tuned simultaneously, where each combination of the two was tested. Population size (200, 400, and 800), tournament size (2, 3, 5, 10, and 20), and offspring (20, 40, 80, 100) were tuned individually; the best result was used while tuning the other parameters. After tuning, all experiments were run using a tournament size of 3. The final values for each of the hyper parameters are shown in Table 7.1.

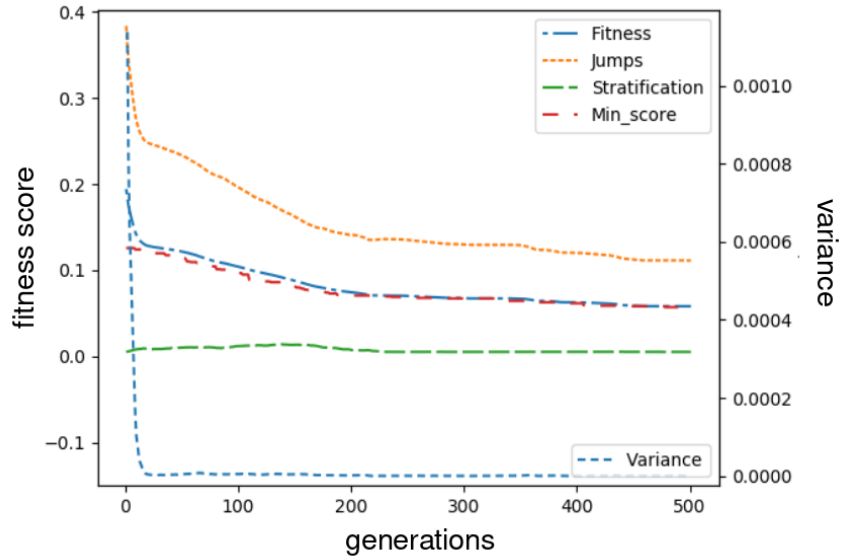
7.3.2.2 Results In Figures 7.5a and 7.5b the jump and stratification scores give equal weight to the final fitness score. The plots show that the GA moves towards convergence,

which is the desired result. We can also see that using scramble seemed to explore more of the search space as there was a slightly larger change in the jump and stratification scores for the population. More importantly, there were much larger changes in variance of the population across the generations, indicating better diversity while still converging.

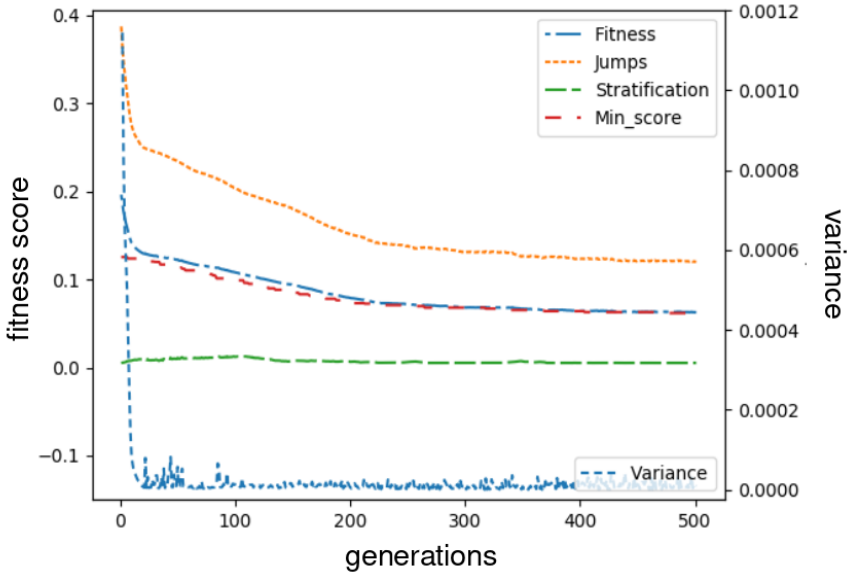
When setting the jump score weight to 75%, the results again indicated convergence for both the scramble and swap mutation methods. It is interesting to note that the swap mutation seemed to find lower jump scores than scramble. This might indicate that scrambling changes the maps too much, producing offspring that do not reduce the jump or stratification score. However, when we look at variance, we can see that scramble has lower overall variance with this weight scheme as compared to the equally weighted objectives. This would also explain why convergence is slower, as it would take longer to find better prescriptions. Swap mutation makes smaller adjustments, thereby possibly providing maps that better maintain the overall stratification while exploring a minimization in jumps.

In all cases, there is a substantial drop in variance of the fitness scores early on in the process. The initial variance is small to begin with but becomes almost negligible after a few generations. However, a clear change in variance is evident when applying scramble mutation. This indicates that the population fitness becomes very similar early on. Considering the initial prescriptions are being created with the goal of laying out a randomly stratified prescription map for nitrogen rates based on yield and protein bins, it makes sense that there would not be much variance in the overall fitness. Once the jumps start to drop, the fitness scores would become even more similar. The plots show that the largest drop in jump score also occurs early in the process, making the drop in variance a logical consequence.

The average total fitness score results for ten runs of the GA are shown in Table 7.2. A paired T-test was performed to confirm that the scores for scramble and swap mutation are statistically different from each other for all fields and both binning methods at the $\alpha = 0.05$ level. Furthermore, the results show that the fitness scores for swap mutation are

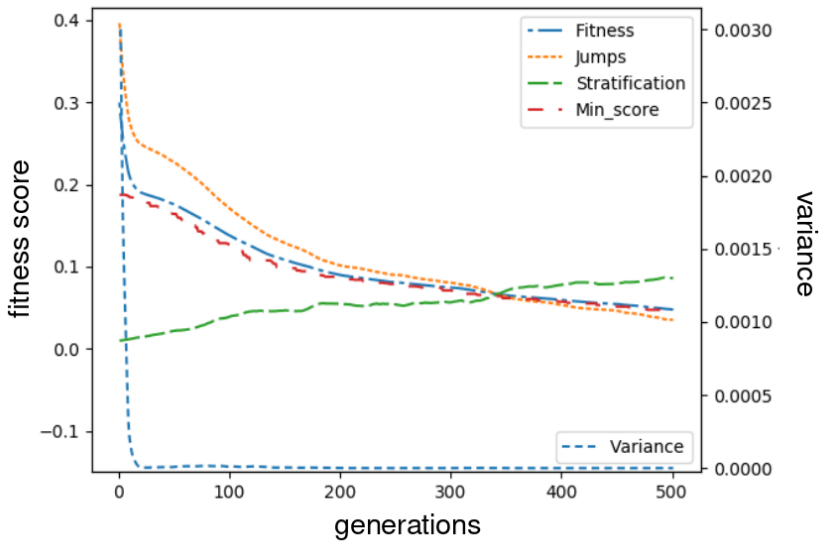


(a) Swap mutation using equal weight ($w = 0.5$).

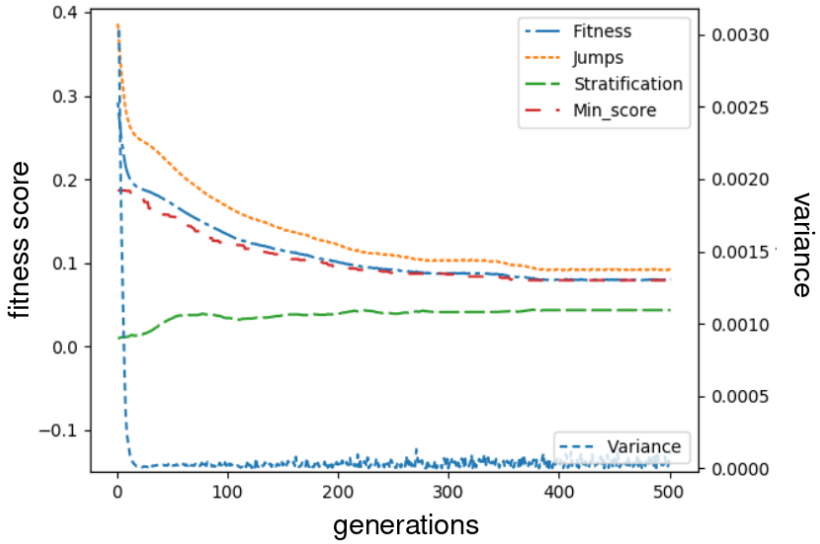


(b) Scramble mutation using equal weight ($w = 0.5$).

Figure 7.5: Field "sre 1314" results for 500 generations of the GA with equally weighted objectives using the two different mutation types and equal sample binning, with tournament size 3. The left y-axis shows the fitness score values, while the right y-axis details the variance value.



(a) Swap mutation with $w = 0.75$, emphasizing jump score minimization.



(b) Scramble mutation with $w = 0.75$, emphasizing jump score minimization.

Figure 7.6: Field "sre 1314" results for 500 generations of the GA with a stronger focus on the jump score using the two different mutation types and equal sample binning, with tournament size 3. The left y-axis shows the fitness score values, while the right y-axis details the variance value.

		Sec35Mid		Davidsonmidwest		Sre1314	
		3	**20**	**3**	**20**	**3**	**20**
Equal Width	**Swap**	0.0282	0.0207	0.0493	0.0195	0.0582	0.0578
	Scramble	0.0520	0.0401	0.0385	0.0601	0.0627	0.0756
Equal Sample	**Swap**	0.0342	0.0309	0.0425	0.0213	0.0679	0.0525
	Scramble	0.0364	0.0468	0.0489	0.0578	0.0613	0.0744

Table 7.2: Average fitness score of the best maps after ten runs of the GA for scramble and swap mutation, using equal width and equal sample binning, on three different fields. The jump weight is set to $w = 0.5$.

consistently lower, and that the GA achieves a lower fitness score for both discretization methods.

7.3.3 Experimental Prescription Maps with an Ethical Objective

As a precursor to creating optimal prescription maps, we wanted to expand the experimental trials to include a fertilizer minimization objective to address the environmental impact of fertilizer. Having three objectives, the weighted sum method did not provide sufficient insight into the trade-off between the objectives, and MOFEA was applied.

7.3.3.1 Experimental Approach

We pose the following hypothesis: Including an ethical objective to minimize overall fertilizer rate does not significantly degrade Montana winter wheat yield. To evaluate our hypothesis we examined the cropping of three different fields using three MOO-algorithms, which we denote NSGA2, CC-NSGA2, and F-NSGA2 for basic NSGA2, cooperative coevolutionary NSGA2, and factored NSGA2 respectively.

For our experiments, we collected data on three fields from two farms. We used the farmer designations for these fields (Henrys, Sec35Mid, and Sec35West). Previously, we trained a convolutional neural network (CNN) based on prior experiments to predict yield

from the wheat harvested on these fields [104, 105]. Because we use this trained CNN to predict yield and analyse the effects of minimizing fertilizer, we had to use fields for which we had the appropriate data available; thus the change for two of the fields for which we are creating experimental prescription maps. We created an initial, random prescription based on the field boundary. The farmer provided information on the width of their fertilizer application equipment and which fertilizer rates to apply across the field. For our experiments, the cell size for Henrys was 300 ft by 450 ft, and the cell size for both Sec35Mid and Sec35West were 200 ft by 300 ft. For all prescriptions, 6 different fertilizer rates were specified in pounds per acre: $F = \{20, 40, 60, 80, 100, 120\}$. Once the initial grid was created, the cells were ordered based on the "as-applied" route the farmer takes across the field to apply fertilizer.

In order to limit runtime and reduce parameter tuning, each of the algorithms was set to terminate after the non-dominated archive did not change for five iterations. Mutation rate, using swap mutation, and crossover rate were set to 0.1 and 0.9 respectively, based on results in the original experimental prescription design paper [113]. The parents for crossover were selected using tournament selection with tournament size 5. The remaining parameters are the population sizes for all three algorithms and the number of iterations NSGA2 needs to be run on the subpopulations for F-NSGA2 and CC-NSGA2. To determine these parameter settings, a grid search was performed. Four different population sizes were considered, $\{100, 200, 500, 800\}$, and three different iteration limits, $\{50, 100, 200\}$. Based on the results of the grid search, a population size of 500 was chosen for all algorithms, and an iteration limit of 100 was chosen for both CC-NSGA2 and F-NSGA2.

The factor architecture for F-NSGA2 was determined using a linear grouping approach, where each group/factor has a size of 10 cells with 5 overlapping cells [138]. For CC-NSGA2, where the subpopulations do not overlap, the factors were determined based on the length of a single strip, i.e., one group includes the cells from one side of the field to the opposite

side where the applicator has to turn around, which is a common approach used by farmers today.

Pareto Front Evaluation The used evaluation metrics are the hypervolume indicator (HV), the spread indicator (S), and the coverage (C) of the fronts. We chose four different non-dominated solutions from the approximate Pareto Front created by each algorithm for each field. These solutions are based on the three extreme points in the Pareto Optimal set: minimum jump score, maximum stratification score, and minimum fertilizer rate. The centroid for these three solutions, $\mathbf{x}_c$, was found as follows, where k represents the objectives:

$$\mathbf{x}_c^j = \frac{1}{3}\sum_{i=1}^{3} \mathbf{x}_i^j, \ \forall j \in k \tag{7.1}$$

The non-dominated solution closest to this centroid (based on the Euclidean distance), is used as the fourth solution.

The four prescriptions were processed by a trained, lean Convolutional Neural Network (CNN) known as Hyper3DNetReg [104]. The CNN was built using field-specific data from 2016-2020. This data includes satellite images, elevation, applied fertilizer rates, and historical yield data. The CNN returned a yield prediction based on the experimental prescription map. Based on this prediction, we assessed whether yield varied significantly across the different non-dominated solutions. To do this, we fit a linear model to the yield points for the three different factor variables: algorithm, field, and objective. An ANOVA was then applied to the resulting linear model [53].

7.3.3.2 Results Applying the ANOVA test to the yield results, we found that there were significant differences ($\alpha = 5\%$) between yield predicted for each of the fields, as well as the different algorithms for each field. However, no significant difference was found between the results for different objectives, confirming our hypothesis that ethical objectives do not impact yield. In this section we present and discuss these results.

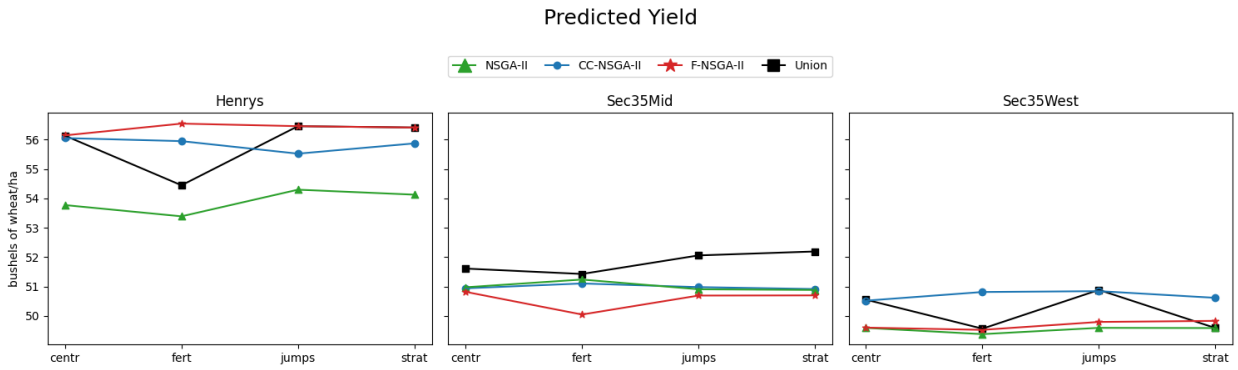


Figure 7.7: Yield prediction results averaged across the entire field based on the four different prescription maps, each focusing on different objectives. The results are connected to show how they are positioned relative to each other.

For each field, the yield predictions for a specific prescription were averaged to create Figure 7.7. A summary for each field of the HV and S for each algorithm's non-dominated sets averaged over ten runs are given in Table 7.3. We also include the union front in these results ($\mathbf{X}^*$), which combines the non-dominated solution sets from the different algorithms into a single non-dominated solution set. Finally, the coverage results are presented in Tables 7.4 and 7.5 using a randomly selected run for each algorithm as to avoid bias. Table 7.4 indicates the percentage that the row algorithm covers the column on the given field. Coverage in relation to the union front is presented in Table 7.5, where the non-dominated sets found by the algorithms ($\mathbf{X}'$) were compared to $\mathbf{X}^*$, using the adjusted coverage (AC) metric.

7.3.3.3 Discussion The prescription maps across all three algorithms, as well as the union front, produced consistent yield predictions with small fluctuations between the different objectives, as can be seen in Figure 7.7. The statistical results confirmed what can be assessed visually in these plots: there is no significant difference between the predicted yield values across the different objectives, including those for the union front. When looking at coverage between algorithms (Table 7.4), we can see that F-NSGA2 has the highest

		Henrys	**Sec35Mid**	**Sec35West**
NSGA2	HV	0.463	0.465	0.469
	Spread	694.405	794.552	637.886
CC-NSGA2	HV	0.387	0.397	0.396
	Spread	574.451	610.866	551.742
F-NSGA2	HV	0.498	0.504	0.474
	Spread	719.386	834.015	747.126
($\mathbf{X}^*$)	HV	0.589	0.593	0.578
	Spread	767.594	934.452	791.959

Table 7.3: Hypervolume (HV) and spread (S) results for the final non-dominated set found by each algorithm, as well as for the union front ($\mathbf{X}^*$), where all three solution sets are combined and evaluated for non-domination. All results were found to be statistically significantly different based on the Kruskal-Wallis and Wilcoxon rank sum tests with $\alpha = 0.005$, with the exception of the S results for NSGA2 and F-NSGA2 for Henrys and Sec35Mid.

coverage for all but one case (for Sec35West classic NSGA2 covers more of F-NSGA2). This indicates that the non-dominated solutions found by F-NSGA2 dominate most of the solutions found by the other algorithms. The union front coverage results confirm that F-NSGA2 seems to cover more non-dominated solutions in the objective space than the other algorithms (Table 7.5), since it contributed the largest percentage of solutions to the union front. NSGA2 also made large contributions to the union front, while CC-NSGA2 had the smallest contribution for all results sets. This could potentially be explained by the use of disjoint subpopulations in CCEA, since its disjoint nature means that a part of the solution space may be left unexplored. On the other hand, FEA uses the overlap to find more diverse solutions across the subpopulations, not only by saving a non-dominated solution from each subpopulation, but through the replacement of single variables in the global solution as well.

The HV and S results (Table 7.3) further confirm our hypothesis that F-NSGA2

		NSGA2	**CC-NSGA2**	**F-NSGA2**
NSGA2	Henrys	N/A	**0.956**	0.823
	Sec35Mid	N/A	0.979	0.937
	Sec35West	N/A	0.863	**0.924**
CC-NSGA2	Henrys	0.614	N/A	0.371
	Sec35Mid	**0.989**	N/A	0.292
	Sec35West	**0.943**	N/A	0.678
F-NSGA2	Henrys	**0.959**	**0.951**	N/A
	Sec35Mid	**1.000**	**1.000**	N/A
	Sec35West	0.857	**0.778**	N/A

Table 7.4: Coverage $C(row, column)$ for the three algorithms for each of the fields, where the algorithm indicated on the left is measured with respect to how much it "covers" the algorithms across the top. **Bold** text indicates which algorithm had the most coverage in the pairwise comparison.

	Henrys	**Sec35Mid**	**Sec35West**
NSGA2	38.9%	30.2%	27.7%
CC-NSGA2	11.3%	9.5%	26.2%
F-NSGA2	49.8%	60.3%	46.1%

Table 7.5: Adjusted coverage results, where each algorithm's non-dominated set is compared to the union front.

explores more of the objective space. Across all fields, F-NSGA2 had the largest S and HV for its approximate Pareto front. According to a Wilcoxon Rank Sum test with $p = 0.05$, the HV results were found to be significantly different for all three fields; however, no significant difference was found for the S results for Henrys and Sec35Mid between F-NSGA2 and NSGA2. When comparing the algorithms' results to the union front, we see that the union front HV and S is not much larger than those found by F-NSGA2, which is in line with the

	Henrys			Sec35Mid			Sec35West		
	NSGA	**CCEA**	**FEA**	**NSGA**	**CCEA**	**FEA**	**NSGA**	**CCEA**	**FEA**
Fert.	9248.75	9815.00	9777.25	10634.62	10087.69	10026.92	8616.90	8129.86	8504.51
Jump	10532.25	10419.00	10305.75	10908.08	10573.85	10482.69	9253.80	8504.51	9141.41
Strat.	10343.5	10343.50	10003.75	10847.31	10421.92	10847.31	9253.80	9141.41	8991.55
Center	10079.25	10494.50	10154.75	10756.15	10786.54	10330.77	9253.80	8392.11	8616.90

Table 7.6: Estimated total applied fertilizer across the field for each prescription type in pounds of nitrogen.

aforementioned coverage results.

Lastly, we evaluated the practical implication of the difference in applied fertilizer. The total amount of nitrogen prescribed based on the different prescription maps generated can be seen in Table 7.6. On average this resulted in a five percent reduction of fertilizer application across the fields regardless of which prescription is compared to the minimized fertilizer rate. However, some prescriptions apply over ten percent more fertilizer than the minimized fertilizer rate prescription. These amounts will not have a large impact on a field's yield, as shown in the yield prediction results, but they could make a difference to the environment by reducing the amount of fertilizer that goes into the water streams, thereby positively impacting pollution [133]. This shows that adding ethical objectives and solving the newly constructed, multi-objective problem using an MOEA is a feasible computational approach to address ethical concerns.

7.4 Optimal Prescription Maps

Optimal prescription maps specify fertilizer rates to apply based on crop response and economic models to maximize expected net return. These maps depend upon the ability to predict yield based on the prescribed inputs, general field information, and satellite data

such as the normalized difference vegetation index (NDVI). The classic way to approach yield prediction is to use linear regression or quadratic plateau regression [110]; however, this approach is limited in its ability to represent the yield response curves. As a result, machine learning approaches such as Random Forests [72] and Deep Learning [151] have become more popular. We use these models to predict the yield and use the result to determine the expected net return. The net return is then used as one of the objective functions in the multi-objective optimization process.

7.4.1 Optimal Prescription Objective Functions

The optimal prescriptions optimize for the following three objectives:

1. Minimize jumps in consecutive cells
2. Maximize net return
3. Minimize overall fertilizer rate

The jump and fertilizer minimization objectives carry over from the experimental maps, but we are now trying to maximize net return instead of stratification. Since we are dealing with a continuous optimization problem when creating optimal prescription maps, the jump score calculation is adjusted to take the sum of the difference in rates of all consecutive cells:

$$Fitn_{jumps} = \sum_{i=1}^{c-1} |F(map_i) - F(map_{i+1})|,$$

where $F(map_i)$ is the fertilizer rate for the ith cell on the field. The jump score now sums over the absolute difference in applied fertilizer between adjacent cells determined by an "as-applied" map.

We use the following definition to maximize net return (NR):

$$NR = Y \times P - AA \times CA - FC, \tag{7.2}$$

where Y is the expected crop yield, obtained through a predictive model, P is the crop selling price, AA is the "as-applied" fertilizer rate, CA is the fertilizer cost, and FC reflects any fixed costs associated with production. As previously mentioned, the expected yield is predicted using a machine learning model. We train this model using the original data points, where we use the most recent yield as the label to predict. Each of the original data points has an "as-applied" fertilizer rate; for our optimization model, we adjust this rate to be the fertilizer rate we are prescribing for the cell each data point belongs to. We process adjusted data points with the predictive model to obtain yield predictions for our prescription map. We calculate the total expected yield (bushels per acre) and input that number into Equation 7.2. We used 2022 economic data provided by the US Department of Agriculture to determine crop price and fertilizer cost [148].

7.4.2 Yield Prediction Dataset Reduction

When integrating predictive machine learning models into the optimization process, we found that the process was computationally expensive when predicting yield for every point on the field. To help reduce the time cost of the optimization process, we wanted to reduce the number of points for which to make predictions, i.e., the number of points being processed by the trained predictive model (the "adjusted data points" in Figure 7.8). To this end, we set up an experiment comparing predictions using different data reduction methods to predictions using the full dataset for two fields: Sec35Mid and Henrys. We use the aforementioned Hyper3DNet CNN for the predictions [104].

7.4.2.1 Dataset Reduction Approaches

We used three different approaches to reduce the data set: random sampling, spatial sampling, and aggregation. The first two approaches select a subset of data points from each cell in the field to make predictions for, while the aggregation approach averages the information of all the data points per cell. An example of each case is shown in Figure 7.9. We have a field with 16 cells and 63 data points, where

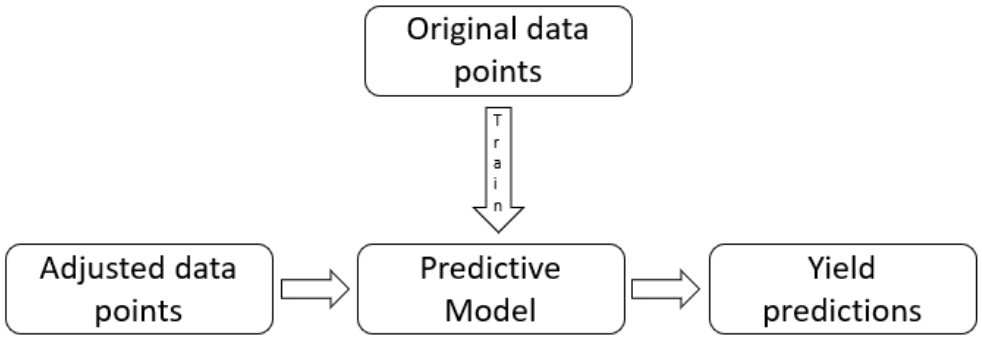


Figure 7.8: Yield prediction data flow. The original data points are used to train a predictive machine learning model. We can then use the trained model to predict yield by sending through adjusted data points.

each cell has two to five data points. For random sampling, we randomly select k data points from each cell. In our experiments, k is determined based on the number of data points in a cell; we select 10% of the total number of data points. In the example in Figure 7.9, we select one or two random data points per cell. Spatial sampling divides a cell into l equally sized sections and selects a point from each of these sections, ensuring we get data from different parts of each cell. Figure 7.9 shows each cell being split in half, and a point is selected from each of these halves. Lastly, the aggregate method does not select subset of data points, rather, it creates a "new" data point by averaging the data of all the points.

7.4.2.2 Results First, we calculated the total yield for each field based on the predictions made by the CNN. For the full data set, random sampling, and spatial sampling, we do this by averaging the predicted yield for the data points in a cell to get a single yield prediction for said cell. We used each cell's yield prediction, in bushels per acre, to calculate the number of bushels of wheat predicted for each cell and sum these predictions to get the total number of bushels predicted for each field (Figure 7.10).

Second, we showed the difference in predicted bushels for the datapoint reduction approaches and the full solution set. Figure 7.11c shows the difference for the full field

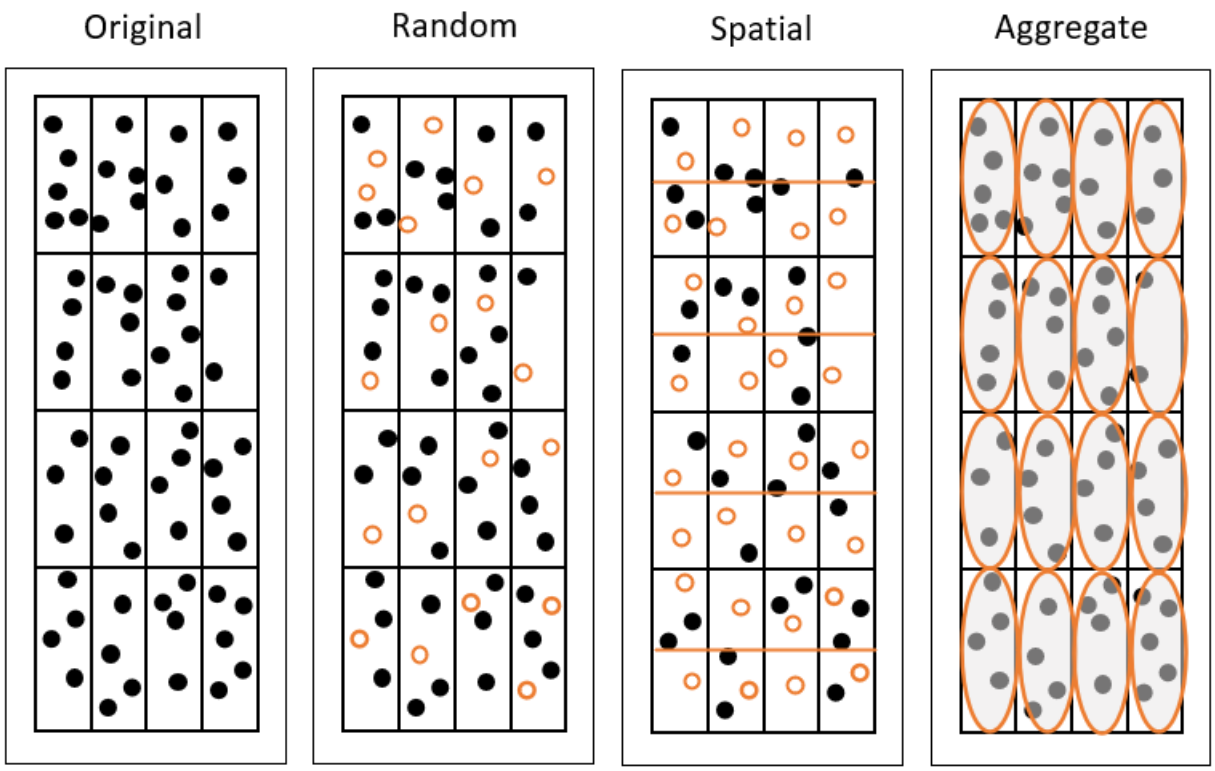
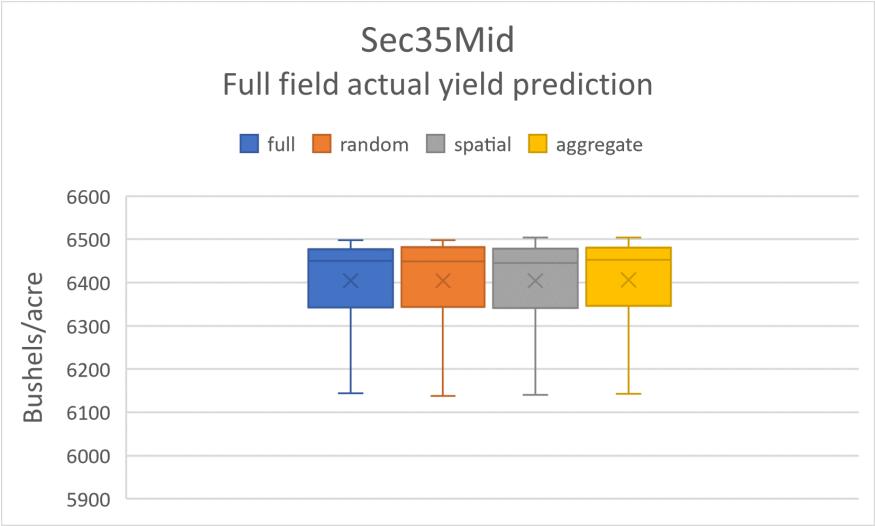


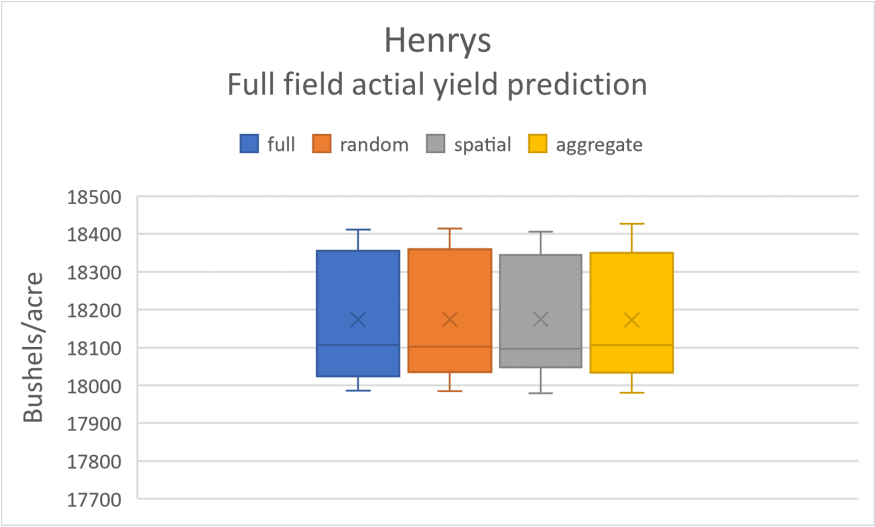
Figure 7.9: The three different types of sampling approaches.

predictions (as presented in Figure 7.10). To give us a more fine grained view of the differences in predicted values, we also report the average of the per cell difference for the different approaches in Figure 7.12. For each cell, we calculated the difference between the prediction made using the full dataset and the reduced dataset, and then we averaged these differences.

7.4.2.3 Discussion Overall, the predictions are similar across the sampling methods, with no significant difference in yield predictions according to the student T-test with $\alpha = 0.05$. Interestingly, spatial sampling had the largest difference from the full data set prediction for both fields. Random sampling had the smallest difference per cell for Henrys and the smallest difference on the full field for Sec35Mid. The reverse is true for aggregate sampling, where Sec35Mid had the smallest difference per cell and Henrys had the smallest



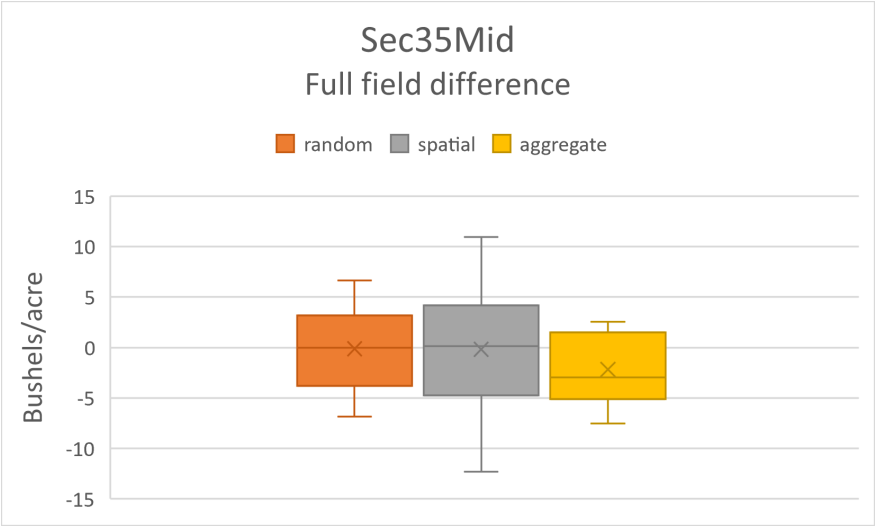
(a) Field Sec35Mid.



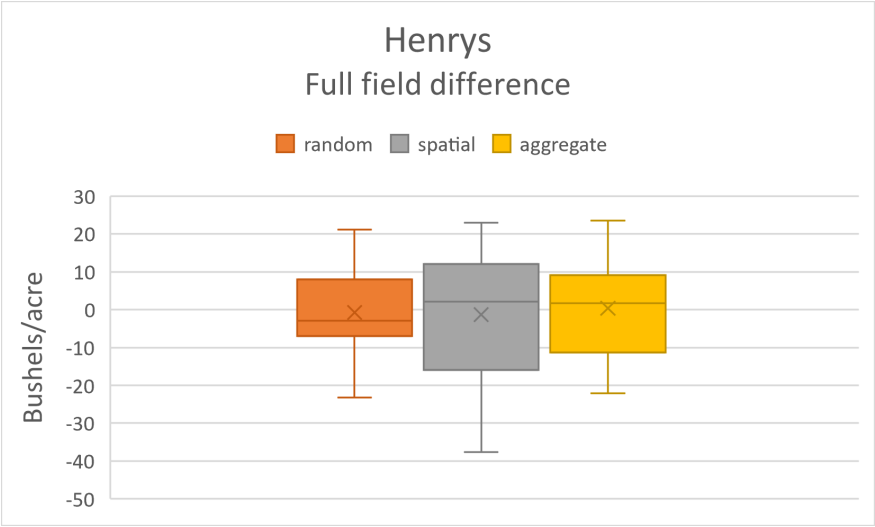
(b) Field Henrys.

Figure 7.10: Total yield prediction results for the entire fields using the different methods.

full field difference. These results appear counter-intuitive; it would make more sense that spatial sampling would have the best results given the amount of spatial variety on a field [61].



(a) Field Sec35Mid.

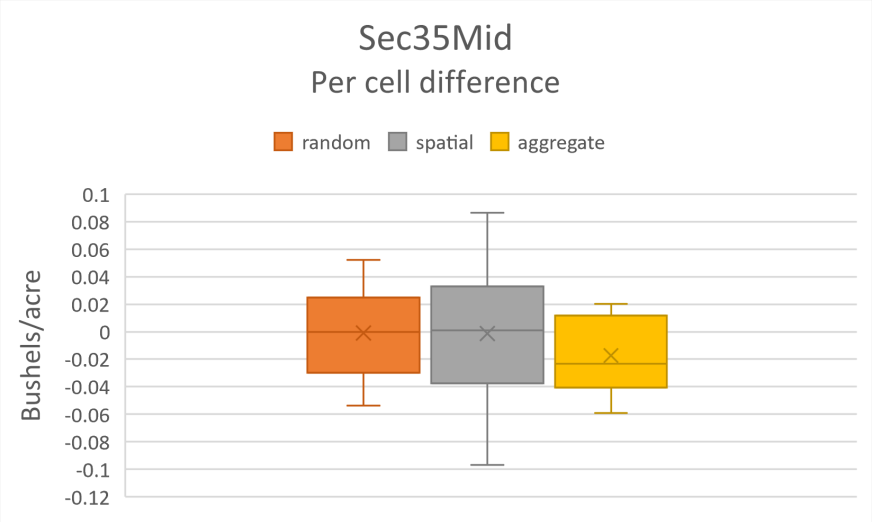


(b) Field Henrys.

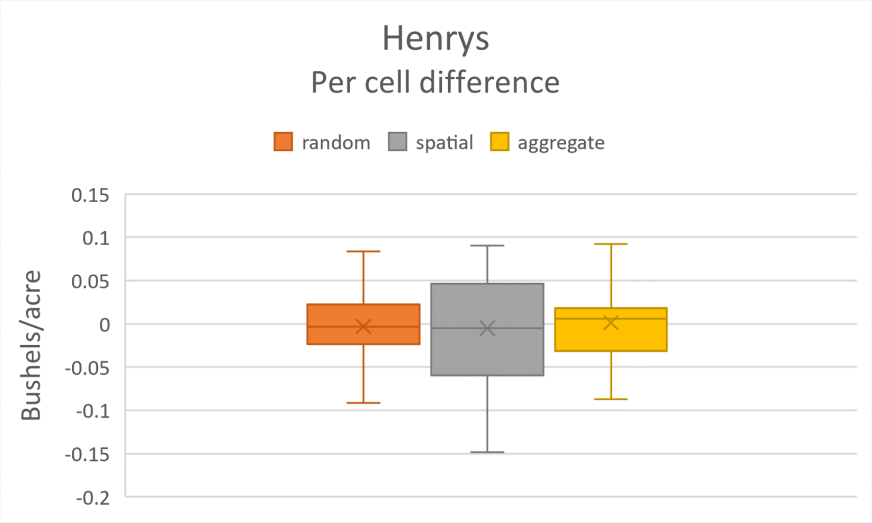
(c) Difference in predicted yield for the entire field as compared to the full field prediction.

7.4.3 Optimal Prescriptions with Ethical Objectives

In this section, we present our results for the creation of optimal prescription maps including the aforementioned ethical objectives. We show results for fields Sec35Mid (shown in Figure 7.2) and Henrys.



(a) Field Sec35Mid.



(b) Field Henrys.

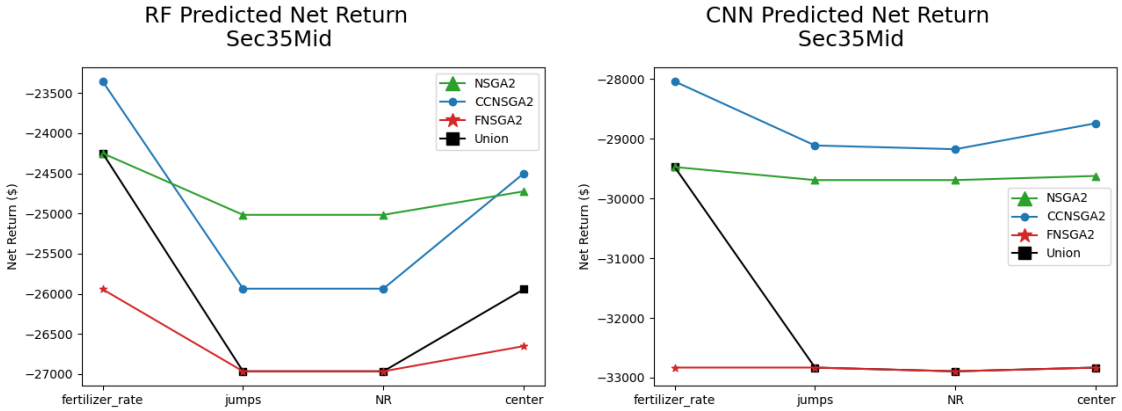
Figure 7.12: Average of the difference in predicted yield per cell.

7.4.3.1 Experimental Approach Mutation rate and crossover rate were set to 0.1 and 0.9 respectively. Swap mutation was used, and the parents for crossover were selected using tournament selection with tournament size 5. The remaining parameters are the population sizes for all three algorithms and the number of iterations NSGA2 needs to be run on the subpopulations for F-NSGA2 and CC-NSGA2. For these experiments, our stopping criterion

is the number of fitness evaluations. This approach led to an increase in runtime; however, since farmers are not yet able to create their own optimal prescription maps, we can generate these maps beforehand, negating the need for a reduced runtime. To achieve this, we set the number of generations and population size such that each algorithm has approximately the same number of function evaluations (FEs). For NSGA2, this resulted in setting a population size of 500 and running the algorithm for 500 iterations, yielding $500 \times 500 = 250,000$ FEs. Our instance of CC-NSGA2 has 24 groups, resulting in CC-NSGA2 being run 10 times where each subpopulation is of size 50, and NSGA2 is run for 20 generations on each subpopulation: $10 \times 20 \times 50 \times 25 = 250,000$. We used the same logic for F-NSGA2, where population size is decreased to 25 to accommodate the increase in the number of subpopulations. We used a Random Forest (RF)[72] and a CNN called Hyper3DNet [104] as regression models to predict yield. Based on the results in the previous section, we used aggregate sampling for Sec35Mid and random sampling for Henrys to reduce the data set for yield predictions.

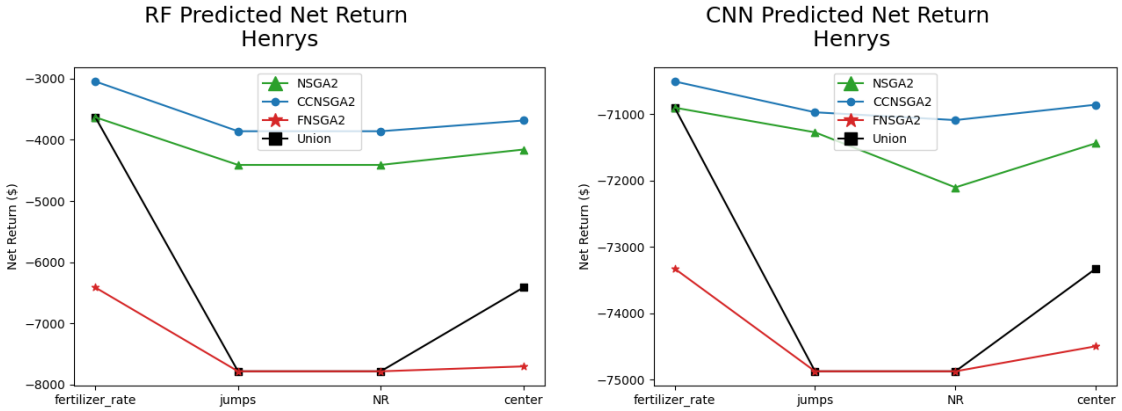
7.4.3.2 Results For each algorithm combination (MOEA and prediction), we chose four different non-dominated solutions from the approximate Pareto Front. These solutions were based on the three extreme points in the Pareto Optimal set: minimum jump score, minimum fertilizer rate, and maximum net return. The centroid for these three solutions, $\mathbf{x}_c$, was found using Equation 7.1, which represents the "center" solution. We also combined the non-dominated solution sets into a union front (re-evaluating non-dominance) and selected the same four solutions from this union front. We present the Net Return results for these optimal prescription solutions in Figures 7.13 and 7.14. Since we are minimizing the objectives, the net return sign has been flipped to go from maximization to minimization. Therefore, solutions that fall lower on the y-axis are better in terms of net return.

We wanted to investigate the difference in NR as compared to the experimental prescription maps, so we used the CNN yield predictions for those experimental maps and



(a) Yield predictions using a RF. (b) Yield predictions using a CNN.

Figure 7.13: Net return for the four different prescription maps for field Sec35Mid.



(a) Yield predictions using a RF. (b) Yield predictions using a CNN.

Figure 7.14: Net return for the four different prescription maps for field Henrys.

plugged them into our NR calculation. This resulted in the predicted NR for fields Sec35Mid and Henrys shown in Table 7.7.

To evaluate algorithm performance, we report the HV, S, and adjusted coverage (AC) results in Tables 7.8 and 7.9. Since the yield predictions influence the net return calculation, we note that we cannot compare algorithm results across the predictive methods. We can only compare the algorithms using the same predictive method.

	Sec35Mid			Henrys		
	NSGA2	CCNSGA2	FNSGA2	NSGA2	CCNSGA2	FNSGA2
Fert	31744	31291	31144	45419	43884	44655
Jump	31681	31024	31269	45468	44033	45036
Strat	31549	31023	31784	45531	42757	43924
Center	31723	31607	31226	45557	44073	44651

Table 7.7: Predicted Net Return in USD ($) based on the yield predicted by the CNN for the different types of experimental prescription maps.

		HV	S	AC
RF	NSGA2	13453.96±206.73	554.02±171.35	85%
	CCNSGA2	9725.35±489.44	281.01±137.28	0%
	FNSGA2	13911.25±104.81	425.20±111.53	15%
CNN	NSGA2	15877.03±424.21	605.87±205.33	75%
	CCNSGA2	11341.00±151.34	489.19±198.67	0%
	FNSGA2	18474.29±1558.07	268.21±205.49	25%

Table 7.8: Optimal prescriptions: Hypervolume (HV), spread (S), and adjusted coverage (AC) results for each algorithm on field Sec35Mid.

Lastly, we present an estimate of the total pounds of nitrogen fertilizer to be applied to the field for the different prescription in Tables 7.10 and 7.11. This gives us an idea of the difference in fertilizer depending on the optimized objective.

7.4.3.3 Discussion Applying an ANOVA test to the net return results for the different types of prescription maps confirms that there is no significant difference in net return for any of the results. When we visually inspect the net return values in Figures 7.13 and 7.14, we

		HV	S	AC
RF	NSGA2	2185.43±83.05	412.54±157.49	75%
	CCNSGA2	1313.48±86.61	175.35±65.33	0%
	FNSGA2	3977.77±272.01	335.85±255.14	25%
CNN	NSGA2	36911.82±543.38	1162.03±297.84	85%
	CCNSGA2	27866.85±251.34	726.19±108.70	0%
	FNSGA2	40676.59±367.49	1540.85±371.14	15%

Table 7.9: Optimal prescriptions: Hypervolume (HV), spread (S), and adjusted coverage (AC) results for each algorithm on field Henrys.

	RF			CNN		
	NSGA2	CCNSGA2	FNSGA2	NSGA2	CCNSGA2	FNSGA2
Fert	9917	10539	8417	10001	10419	6746
Jump	10683	11898	9438	10278	11491	6746
NR	9917	10539	8417	10001	10482	6777
Center	10209	10754	8731	10014	11062	6746

Table 7.10: Estimated applied fertilizer across the field for each prescription type for field Sec35Mid using a Random Forest (RF) and Convolutional Neural Network (CNN) for yield predictions.

can confirm that the difference in net return when focusing on different objectives is minimal for each algorithm. The largest difference in net return is approximately $2,000. We would like to note that currently the net return calculation does not include the cost of wear on equipment. If farmers could gather data on how large jump rates impact them economically, we could refine our net return calculation. When we compare the optimized NR values to the NR found for the experimental maps, it is interesting that F-NSGA2 is the only approach

	RF			CNN		
	NSGA2	CCNSGA2	FNSGA2	NSGA2	CCNSGA2	FNSGA2
Fert	18339	18835	14725	17721	18477	14661
Jump	19118	19649	16095	18898	18703	15956
NR	18339	18835	14725	17915	18624	14661
Center	18590	19011	14806	18074	18645	15083

Table 7.11: Estimated applied fertilizer across the field for each prescription type for field Henrys using a Random Forest (RF) and Convolutional Neural Network (CNN) for yield predictions..

that finds a better net return for the optimal maps for Sec35Mid. Furthermore, the actual difference in NR is not very big. The same does not hold true for Henrys, where there is a difference of over $\$\ 20,000$ when optimizing NR regardless of the optimization method used.

When evaluating the algorithms' performance, we find significant differences for the S and HV results per a Wilcoxon Rank Sum test with $p = 0.05$ (Tables 7.8 and 7.9). We see that F-NSGA2 has a higher HV than NSGA2, but F-NSGA2 has large variance when using the CNN as the predictive algorithm. We believe this could be due to the use of FE's as a stopping criterion. In [44], results indicate that using FE's may result in an unfair stopping condition. Based on our results, we believe F-NSGA2 did not have enough FE's to converge. This could also explain the S results for the algorithms. All the solution sets have large S variance, which indicates that different runs explore different parts of the search space, i.e., there is no clear convergence toward the same Pareto optimal solutions.

NSGA2 covers most of the union front, and F-NSGA2 contributes the remainder of the non-dominated solutions. However, F-NSGA2 finds solutions with a higher net return, where the solutions found by F-NSGA2 dominate those found by CC-NSGA2. Overall, CC-NSGA2 has the poorest performance out of the three approaches. When we look at the plots

in Figures 7.13 and 7.14, this can be visually confirmed; we see that F-NSGA2 found the highest net return values regardless of the implemented prediction algorithm. Considering one of the main concerns for farmers is profit, single population NSGA2 may not be the best choice, since all its solutions had a lower net return than those found by F-NSGA2.

The estimated applied fertilizer for the optimal prescriptions for Sec35Mid (Table 7.10) was similar to that for the experimental prescriptions, albeit slightly higher (Table 7.6). However, the estimated applied fertilizer for Henrys is higher by several thousands of pounds (7.11). We can once again see that when focusing on minimizing fertilizer rate, the amount of nitrogen applied is reduced by hundreds of pounds. This confirms that a multi-objective approach can indeed be beneficial to find solutions that have environmental benefit. Furthermore, the F-NSGA2 approach consistently found solutions with the highest net return and the lowest fertilizer application, indicating that F-NSGA2 explores relevant areas of the objective space. Lastly, we notice that the CNN generally predicts a higher net return with less fertilizer applied. This is especially apparent for field Henrys, where the CNN predicts a NR that is ten times larger than that predicted by the RF. It would be beneficial to do a more in-depth study on the effects of different predictive algorithms on the optimization process to discover if this is a result of the predictions or the optimization process.

7.5 Concluding Remarks

Multi-Objective Optimization provides a way for sustainability issues to be addressed when optimizing fertilizer prescriptions in precision agriculture. In this research, we investigated adding two sustainability-focused objectives to existing precision agriculture problems, that of creating experimental and optimal fertilizer prescription maps. For both problems three competing objectives were optimized: the base objective, stratification and net return maximization respectively, and two sustainability objectives, fertilizer rate jump

minimization and overall fertilizer rate minimization. We applied three different MOO algorithms, NSGA2, CC-NSGA2, and F-NSGA2, of which the latter is an adaptation of the Factored Evolutionary Algorithm in which overlapping subpopulations are used to find an approximate Pareto front. We found that all three MOO algorithms could find optimized prescription maps successfully, and that including these sustainability objectives had minimal impact on yield and net return. Based on these results, we confirmed our hypothesis that focusing on sustainability need not significantly influence net return, thus indicating a strong justification for modifying farming practices to incorporate such objectives, thereby reducing environmental impact. Furthermore, our results indicated that using overlapping subpopulations increases exploration of the objective space when compared to the single population and disjoint subpopulation alternatives.

CHAPTER EIGHT

CONCLUSION

For our concluding remarks, we summarize the contributions of the dissertation and identify directions for future work.

8.1 Contributions

In this dissertation, we introduced the Multi-Objective Factored Evolutionary Algorithm (MOFEA) to solve Multi-Objective Optimization (MOO) problems, explored properties of variables grouping for both Large Scale Optimization (LSO) and Large Scale Multi- and Many-Objective Optimization, and introduced Objective Archive Management (OAM) for solution set reduction. We also showed how MOO can be used to address environmental concerns in the field of Precision Agriculture.

Chapter 2 introduced fundamental concepts to provide the reader with the necessary tools to understand the work presented in this dissertation. We then presented MOFEA in Chapter 3. MOFEA is an extension of FEA to the multi-objective case. In all previous MOO variable grouping research, only CCEA, which uses disjoint subpopulations, has been applied to MOEA's. MOFEA adds the use of overlapping subpopulations; the overlapping groups are able to account for indirect variable interactions when providing an overlapping architecture without pre-defining variable interacting groups, which is a significant contribution to the field of MOO. While MOFEA allows for overlapping subpopulations, we explained how the framework also works with disjoint subpopulations. Furthermore, we applied MOFEA to three well known MOEAs: NSGA2, SPEA2, and MOEA/D, to demonstrate its general applicability. We first applied MOFEA to NSGA2 and run an empirical analysis on the multi-objective knapsack (MO-KS) problem. We introduced a novel, more complex version

of MOKS. While the classic MOKS optimizes separate knapsacks as separate objectives, we proposed optimizing a single knapsack that maximizes value, minimizes weight, volume, and the difference in weight/volume of the items in the knapsack. We found that this problem is more difficult to solve, thus providing the field of MOO with a new, many-objective combinatorial benchmark problem with 3, 4, or 5 objectives. F-NSGA2 outperformed regular NSGA2 and CC-NSGA2 on all version of the MOKS.

In Chapters 4 and 5, we dove deeper into variable decomposition by performing an empirical analysis on LSO and Large-Scale Multi- and Many-Objective Optimization. For our single population experiments, we extended Differential Grouping (DG) to create Overlapping DG (ODG), and we created a tree-based grouping approach that generates a connected architecture through overlap. In our LSO experiments, we confirmed the benefits of overlapping groups [14, 138]. The results also indicated that variable interaction learning may not be necessary, since the connected architecture created by the Tree-based approach provided high quality results. We affirmed these results in our MOO and MaOO experiments. We applied the MOFEA framework to NSGA2, SPEA2, and MOEA/D, each with disjoint and overlapping subpopulations. We found that overlapping linear and random grouping improve results (compared to disjoint and single population implementations) even though no variable interaction learning is performed. These variable grouping experiments are the first empirical results comparing the three most commonly used grouping strategies for both the disjoint and overlapping case. Furthermore, we are the first to draw connections between different function characteristics and the different grouping strategies, providing valuable insights to the field of MOO and MaOO. Previous research had only investigated the importance of variable interaction learning for partially non-separable problems. Through the use of connected overlapping architectures, we show that variable interaction learning may not be as important as was previously believed. Moreover, we applied DG along the different objectives of a single MOO function, as presented in [85]. Li *et al.* showed that the

DTLZ benchmarks consist of non-separable variable groups along the objectives but noted that no good approach had been developed to address these overlapping groups. MOFEA offers a solution to this problem.

The MOO variable grouping research led us to consider ways of reducing large solution sets in many-objective optimization, which is the focus of Chapter 6. Inspired by the idea of overlapping subpopulations, we created the Objective Archive Management (OAM) strategy, a new solution set reduction strategy. Existing solution set reduction strategies either require MOO specific knowledge to select the appropriate environmental selection operator to reduce the solution set [142], or the use of expensive hypervolume calculations [131] which becomes more difficult as the objectives increase. Our algorithm offers a different approach without these limitations, thus providing MaOO research with a new, easy to use solution set reduction strategy. We find that our approach selected solutions that strike a good balance between the different objectives. Furthermore, we used radar graphs to visualize MaOO solutions; providing a way for humans to visually assess the found non-dominated solutions.

Finally, we applied MOFEA to the real world problem of creating experimental and optimal fertilizer prescription maps. We showed that multi-objective optimization can be used to incorporated environmental objectives. In this specific case, we aimed to maximize net return for farmers; however, excessive fertilizer application can have negative environmental impacts, so we wished to minimize the overall fertilizer to be applied. Additionally, the farmers we work with wanted to reduce strain on their farming equipment by not having large jumps in fertilizer between consecutive cells on a field. This resulted in three competing objectives to be optimized, making MOEA's a natural choice to solve this problem. Through empirical analysis using NSGA2, CC-NSGA2, and F-NSGA2, we find that each algorithm successfully reduces the amount of fertilizer applied without significantly impacting net return. This shows that MOO can be a useful tool in addressing environmental and sustainability concerns. Previously, ethical concerns were not being

addressed throughout the optimization process directly, our research is the first to show that such an integration is possible as well as beneficial.

8.2 Future Work

As shown in our MOFEA experiments, the hypervolume results leave room for improvement. Currently we select a non-dominated solution from each of the subpopulations based on the selection criterion the base algorithm uses to tie-break when filling out the population for the next generation. However, we would like to explore adding a second diversity-based selection criterion for a more guided way of picking a representative solution. For example, we could sort based on the crowding distance for all base-algorithms and not just NSGA2. This would mean that two different sorting mechanisms are employed. We can then either select the "best" solution for each criterion, or we can find the overall best given both. We expect that adding a diversity-based selection procedure to choose the non-dominated solutions from the subpopulations will further improve hypervolume results. We would also like to investigate effective parallelization of MOFEA to help improve efficiency of the algorithm.

In terms of variable grouping, we have mostly looked at the difference between disjoint and overlapping subpopulations, and the effects of variable learning versus a connected architecture. However, it would be interesting to consider the impact of the amount of overlap and the size of individual factors on optimization performance. Our LSO results may indicate improved performance on smaller group sizes due to the success of CPSO-S. Limited hyperparameter tuning was performed in order to maintain consistency across the experiments, so further tuning to improve performance of the individual methods would be beneficial for both the LSO and MOO experiments.

In our research, we proposed some new decomposition strategies to create overlapping groups; however, there are several other decomposition strategies that can be created. We

would like to look at expanding some of the proposed methods as well as exploring other existing methods. For example, in the MOO experiments, we find that DG performs well from an optimization perspective but consumes large amounts of memory and CPU. It would be beneficial to look at combining some of the smaller groups created by DG along the objectives to reduce the number of groups generated as well as overlap size. Alternatively, the graph-based approach by Cao *et al.* could be extended to create overlapping groups. Additionally, because of the increased flexibility introduced by the overlap, we can imagine many methods that better consider the function landscape and can lead to increased performance. For example, by employing a hierarchical decomposition strategy based on local variable interactions across the function's domain.

The aforementioned methods mostly focus on continuous optimization. Another interesting research direction would be looking at ways to perform variable decomposition for combinatorial optimization. We show that overlap allows for objective-wise decomposition with DG, but we have not yet explored how to apply this idea to combinatorial problems. For the knapsack problem, we could sort variables based on their value, weight, and volume and use an aggregation function to group items based on each of these characteristics, hopefully resulting in different groups for each characteristic thus creating overlap.

Our proposed OAM strategy was shown to work well to reduce solution sets to a more manageable size; however, we did not explore the use of OAM as an external archive to re-inject solutions into the next generation. We believe that such solution injection could help improve MaOO algorithm performance in higher objective spaces. Furthermore, we only applied the external archive management strategy to NSGA2, and the single run OAM to NSGA2 and NSGA3. In future work, we would like to investigate how the reduced solution sets change depending on the MOEA used. Lastly, we would like to ensure the return of a non-empty solution set by including an archive weighting strategy. If OAM's overlap does not select any solutions, the solution(s) with the highest weighted archive score would be

returned.

For the PA research, we plan to investigate adding temporal objectives, such as minimizing variation in net return across several years, and including the impact climate change might have on crop response [83]. Another goal is to investigate the effect of different yield prediction approaches when creating optimized prescription maps. In other words, how much influence does accurate yield prediction have on prescribing the correct fertilizer rate? Or is it more important to use a model that accurately describes the shape of the yield response curve? We are currently also looking at how to incorporate experimental rates into the optimal prescription maps. There are two main questions we are trying to address: which cells should be used to apply experimental rates to, and what fertilizer rates should be applied?

Lastly, as a general note for population-based algorithmic research, Engelbrecht shows that using function evaluations as a stopping criterion could impact results in an unfair way [44]. To this end, it would be useful to explore different ways to evaluate how long an algorithm should run. Alternative stopping criteria include the amount of change in the non-dominated archive, a lack of change in hypervolume or other evaluation metrics, or convergence of the non-dominated solutions' fitnesses.

REFERENCES CITED

[1] *MOP Evolutionary Algorithm Approaches*, pages 61–130. Springer US, Boston, MA, 2007.

[2] Hussein A Abbass, Ruhul Sarker, and Charles Newton. PDE: a pareto-frontier differential evolution approach for multi-objective optimization problems. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 971–978, 2001.

[3] Salem F. Adra and Peter J. Fleming. A diversity management operator for evolutionary many-objective optimisation. In *Evolutionary Multi-Criterion Optimization*, page 81–94. Springer, 2009.

[4] Johannes Bader and Eckart Zitzler. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76, 2011.

[5] Reza Behmanesh, Iman Rahimi, and Amir H. Gandomi. Evolutionary many-objective algorithms for combinatorial optimization problems: A comparative study. *Archives of Computational Methods in Engineering*, 28:673–688, Mar 2021.

[6] Julian Blank and Kalyanmoy Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8, 2020.

[7] Aymeric Blot, Marie-Éléonore Kessaci, and Laetitia Jourdan. Survey and unification of local search techniques in metaheuristics for multi-objective combinatorial optimisation. *Journal of Heuristics*, 24(6):853–877, Dec 2018.

[8] R. Bongiovanni and J. Lowenberg-Deboer. Precision Agriculture and Sustainability. *Precision Agriculture*, 5(4):359–387, August 2004.

[9] Karl Bringmann and Tobias Friedrich. Approximation quality of the hypervolume indicator. *Artificial Intelligence*, 195:265–290, Feb 2013.

[10] Andre Britto and Aurora Pozo. Using reference points to update the archive of MOPSO algorithms in many-objective optimization. *Neurocomputing*, 127:78–87, 2014.

[11] Dimo Brockhoff, Tobias Friedrich, and Frank Neumann. Analyzing hypervolume indicator based algorithms. In *International Conference on Parallel Problem Solving from Nature*, page 651–660. Springer, 2008.

[12] Dimo Brockhoff and Eckart Zitzler. Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 2086–2093, 2007.

[13] David S Bullock, Maria Boerngen, Haiying Tao, Bruce Maxwell, Joe D Luck, Luciano Shiratsuchi, Laila Puntel, and Nicolas F Martin. The data-intensive farm management project: Changing agronomic research through on-farm precision experimentation. *Agronomy Journal*, 111(6):2736–2746, 2019.

[14] Stephyn G. W. Butcher, Shane Strasser, Jenna Hoole, Benjamin Demeo, and John W. Sheppard. Relaxing consensus in distributed factored evolutionary algorithms. In *ACM Genetic and Evolutionary Computation Conference (GECCO)*, pages 5–12, July 2016.

[15] Stephyn G.W. Butcher, John W. Sheppard, and Shane Strasser. Pareto improving selection of the global best in particle swarm optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2018.

[16] Lei Cai, Shiru Qu, and Guojian Cheng. Two-archive method for aggregation-based many-objective optimization. *Information Sciences*, 422:305–317, 2018.

[17] Bin Cao, Jianwei Zhao, Yu Gu, Yingbiao Ling, and Xiaoliang Ma. Applying graph-based differential grouping for multiobjective large-scale optimization. *Swarm and Evolutionary Computation*, 53:100626, 2020.

[18] Fabio Caraffini, Ferrante Neri, and Lorenzo Picinali. An analysis on separability for memetic computing automatic design. *Information Sciences*, 265:1–22, 2014.

[19] Matheus Carvalho and André Britto. Influence of reference points on a many-objective optimization algorithm. In *7th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 31–36, 2018.

[20] Wenxiang Chen and Ke Tang. Impact of problem decomposition on cooperative coevolution. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 733–740, 2013.

[21] Wenxiang Chen, Thomas Weise, Zhenyu Yang, and Ke Tang. Large-scale global optimization using cooperative coevolution with variable interaction learning. In *International Conference on Parallel Problem Solving from Nature*, pages 300–309, 2010.

[22] Ran Cheng, Yaochu Jin, Markus Olhofer, and Bernhard sendhoff. Test problems for large-scale multiobjective and many-objective optimization. *IEEE Transactions on Cybernetics*, 47(12):4108–4121, 2017.

[23] CA Coello Coello and Maximino Salazar Lechuga. MOPSO: A proposal for multiple objective particle swarm optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, volume 2, pages 1051–1056, 2002.

[24] Carlos A. Coello Coello. Constraint-handling techniques used with evolutionary algorithms. In *ACM Genetic and Evolutionary Computation Conference (GECCO)*, page 563–587. Association for Computing Machinery, Jul 2016.

[25] Carlos A. Coello Coello, Clarisse Dhaenens, and Laetitia Jourdan. Multi-objective combinatorial optimization: Problematic and context. In Carlos A. Coello Coello, Clarisse Dhaenens, and Laetitia Jourdan, editors, *Advances in Multi-Objective Nature Inspired Computing*, page 1–21. Springer, 2010.

[26] Carlos A Coello Coello, Silvia González Brambila, Josué Figueroa Gamboa, Ma Guadalupe Castillo Tapia, and Raquel Hernández Gómez. Evolutionary multi-objective optimization: open research areas and some challenges lying ahead. *Complex & Intelligent Systems*, 6(2):221–236, 2020.

[27] Carlos A Coello Coello and Gary Lamont. *Applications of multi-objective evolutionary algorithms*. Advances in Natural Computation. World Scientific, 2004.

[28] Cai Dai, Xiujuan Lei, and Xiaoguang He. A decomposition-based evolutionary algorithm with adaptive weight adjustment for many-objective problems. *Soft Computing*, 24(14):10597–10609, 2020.

[29] Indraneel Das and John E Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM journal on optimization*, 8(3):631–657, 1998.

[30] EA Davidson, Mark B David, James N Galloway, Christine L Goodale, Richard Haeuber, John A Harrison, Robert W Howarth, Dan B Jaynes, R Richard Lowrance, Nolan B Thomas, et al. Excess nitrogen in the us environment: trends, risks, and solutions. *Issues in Ecology*, (15), 2011.

[31] TJ De Koeijer, GAA Wossink, and FJHM Verhees. Environmental and economic effects of spatial variability in cropping: nitrogen fertilization and site-specific management. In *The Economics of Agro-Chemicals*, pages 187–200. 2018.

[32] Olivier L De Weck. Multiobjective optimization: History and promise. In *Invited Keynote Paper, GL2-2, The Third China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems, Kanazawa, Japan*, volume 2, page 34, 2004.

[33] Kalyanmoy Deb. Multi-objective optimization. In *Search Methodologies*, pages 403–449. Springer, 2014.

[34] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part 1: solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2013.

[35] Kalyanmoy Deb, Manikanth Mohan, and Shikhar Mishra. Evaluating the -domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. *Evolutionary computation*, 13(4):501–525, 2005.

[36] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[37] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable multi-objective optimization test problems. In *IEEE Congress on Evolutionary Computation (CEC)*, volume 1, pages 825–830. IEEE, 2002.

[38] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. *Scalable Test Problems for Evolutionary Multiobjective Optimization*, pages 105–145. Springer London, London, 2005.

[39] Bernabe Dorronsoro, Grégoire Danoy, Pascal Bouvry, and Antonio Nebro. *Multi-objective Cooperative Coevolutionary Evolutionary Algorithms for Continuous and Combinatorial Optimization*, volume 362, pages 49–74. 07 2011.

[40] Tyrone E Duncan. On the calculation of mutual information. *SIAM Journal on Applied Mathematics*, 19(1):215–220, 1970.

[41] Russell Eberhart and James Kennedy. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[42] Matthias Ehrgott and Xavier Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, 22(4):425–460, Nov 2000.

[43] Alwalid N. Elshafei. Hospital layout as a quadratic assignment problem. *Operational Research Quarterly (1970-1977)*, 28(1):167–179, 1977.

[44] A. P. Engelbrecht. Fitness function evaluations: A fair stopping condition? In *IEEE Swarm Intelligence Symposium*, 2014.

[45] Absalom E Ezugwu, Amit K Shukla, Rahul Nath, Andronicus A Akinyelu, Jeffery O Agushaka, Haruna Chiroma, and Pranab K Muhuri. Metaheuristics: a comprehensive overview and classification along with bibliometric analysis. *Artificial Intelligence Review*, 54:4237–4316, 2021.

[46] Rui Fan, Lixin Wei, Xin Li, Jinlu Zhang, and Zheng Fan. Self-adaptive weight vector adjustment strategy for decomposition-based multi-objective differential evolution algorithm. *Soft Computing*, 24(17):13179–13195, 2020.

[47] Marco Farina, Kalyanmoy Deb, and Paolo Amato. Dynamic multiobjective optimization problems: Test cases, approximations, and applications. *IEEE Transactions on Evolutionary Computation*, 8(5):425–442, 2004.

[48] Nasim Ferdosian, Mohamed Othman, Kweh Yeah Lun, and Borhanuddin Mohd Ali. Optimal solution to the fractional knapsack problem for LTE overload-state scheduling. *IEEE 3rd International Symposium on Telecommunication Technologies (ISTT)*, pages 97–102, 2016.

[49] Carlos M. Fonseca and Peter J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, March 1995.

[50] Nathan Fortier, John Sheppard, and Shane Strasser. Abductive inference in bayesian networks using distributed overlapping swarm intelligence. *Soft Computing*, 19(4):981–1001, April 2015.

[51] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

[52] Laura Garcia, Lorena Parra, Jose M. Jimenez, Jaime Lloret, and Pascal Lorenz. Iot-based smart irrigation systems: An overview on the recent trends on sensors and IoT systems for irrigation in precision agriculture. *Sensors*, 20(4):1042, Jan 2020.

[53] Andrew Gelman. Analysis of variance—why it is more important than ever. *The Annals of Statistics*, 33(1):1–53, 2005.

[54] Chi-Keong Goh and Kay Chen Tan. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 13(1):103–127, 2009.

[55] C.K. Goh, K.C. Tan, D.S. Liu, and S.C. Chiam. A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design. *European Journal of Operational Research*, 202:42—-54, Apr 2010.

[56] Dunwei Gong, Fenglin Sun, Jing Sun, and Xiaoyan Sun. Set-based many-objective optimization guided by a preferred region. *Neurocomputing*, 228:241–255, 2017.

[57] Wenyin Gong and Zhihua Cai. A multiobjective differential evolution algorithm for constrained optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 181–188, 2008.

[58] Brian K. Haberman and John W. Sheppard. Overlapping particle swarms for energy-efficient routing in sensor networks. *Wireless Networks*, 18:351–363, 2012.

[59] Dong Han, Wenli Du, Wei Du, Yaochu Jin, and Chunping Wu. An adaptive decomposition-based evolutionary algorithm for many-objective optimization. *Information Sciences*, 491:204–222, 2019.

[60] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

[61] Paul B Hegedus, Bruce Maxwell, John Sheppard, Sasha Loewen, Hannah Duff, Giorgio Morales-Luna, and Amy Peerlinck. Towards a low-cost comprehensive process for on-farm precision experimentation and analysis. *Agriculture*, 13(3):524, 2023.

[62] John Henry Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[63] Wen-Jing Hong, Peng Yang, and Ke Tang. Evolutionary computation for large-scale multi-objective optimization: A decade of progresses. *International Journal of Automation and Computing*, 18(2):155–169, 2021.

[64] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19:601–608, 2006.

[65] Simon Huband, Philip Hingston, Luigi Barone, and Lyndon While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.

[66] Kokolo Ikeda, Hajime Kita, and Shigenobu Kobayashi. Failure of pareto-based MOEAs: Does non-dominated really mean near to optimal? In *IEEE Congress on Evolutionary Computation (CEC)*, volume 2, pages 957–962. IEEE, 2001.

[67] Hisao Ishibuchi, Naoya Akedo, and Yusuke Nojima. Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems. *IEEE Transactions on Evolutionary Computation*, 19(2):264–283, Apr 2015.

[68] Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, and Yusuke Nojima. Modified distance calculation in generational distance and inverted generational distance. In António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos Coello Coello, editors, *Evolutionary Multi-Criterion Optimization*, pages 110–125, Cham, 2015. Springer International Publishing.

[69] Hisao Ishibuchi and Youhei Shibata. Mating scheme for controlling the diversity-convergence balance for multiobjective optimization. In *ACM Genetic and Evolutionary Computation Conference (GECCO)*, page 1259–1271, 2004.

[70] Hisao Ishibuchi, Noritaka Tsukamoto, and Yusuke Nojima. Evolutionary many-objective optimization: A short review. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 2419–2426, 2008.

[71] Haider Mahmood Jawad, Rosdiadee Nordin, Sadik Kamel Gharghan, Aqeel Mahmood Jawad, and Mahamod Ismail. Energy-efficient wireless sensor networks for precision agriculture: A review. *Sensors*, 17(8):1781, 2017.

[72] Jig Han Jeong, Jonathan P Resop, Nathaniel D Mueller, David H Fleisher, Kyungdahm Yun, Ethan E Butler, Dennis J Timlin, Kyo-Moon Shim, James S Gerber, Vangimalla R Reddy, et al. Random forests for global and regional crop yield predictions. *Public Library of Science*, 11(6), 2016.

[73] F. Jimenez, A.F. Gomez-Skarmeta, G. Sanchez, and K. Deb. An evolutionary algorithm for constrained multi-objective optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, volume 2, pages 1133–1138, 2002.

[74] Rong juan Luo, Shou feng Ji, and Bao lin Zhu. A pareto evolutionary algorithm based on incremental learning for a kind of multi-objective multidimensional knapsack problem. *Computers Industrial Engineering*, 135:537–559, 2019.

[75] Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. The traveling salesman problem. *Handbooks in operations research and management science*, 7:225–330, 1995.

[76] Nattavut Keerativuttitumrong, Nachol Chaiyaratana, and Vara Varavithya. Multi-objective co-operative co-evolutionary genetic algorithm. In *International Conference on Parallel Problem Solving from Nature*, pages 288–297, 2002.

[77] Joshua Knowles and David Corne. Instance generators and test suites for the multiobjective quadratic assignment problem. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Lothar Thiele, and Kalyanmoy Deb, editors, *Evolutionary Multi-Criterion Optimization*, Lecture Notes in Computer Science, page 295–310. Springer, 2003.

[78] Joshua D Knowles, David W Corne, and Mark Fleischer. Bounded archiving using the lebesgue measure. In *IEEE Congress on Evolutionary Computation (CEC)*, volume 4, pages 2490–2497. IEEE, 2003.

[79] Brad Koch, R Khosla, WM Frasier, DG Westfall, and D Inman. Economic feasibility of variable-rate nitrogen application utilizing site-specific management zones. *Agronomy Journal*, 96(6):1572–1580, 2004.

[80] Hans-Peter Kriegel, Erich Schubert, and Arthur Zimek. The (black) art of runtime evaluation: Are we comparing algorithms or implementations? *Knowledge and Information Systems*, 52:341–378, 2017.

[81] Marco Laumanns, Günter Rudolph, and Hans-Paul Schwefel. A spatial predator-prey approach to multi-objective optimization: A preliminary study. In *International Conference on Parallel Problem Solving from Nature*, pages 241–249, 1998.

[82] Eugene L Lawler. The quadratic assignment problem. *Management science*, 9(4):586–599, 1963.

[83] Niklaus Lehmann and Robert Finger. Optimizing whole-farm management considering price and climate risks. In *123rd European Association of Agricultural Economists Seminar*, February 2012.

[84] Bingdong Li, Jinlong Li, Ke Tang, and Xin Yao. Many-objective evolutionary algorithms: A survey. *ACM Comput. Surv.*, 48(1), sep 2015.

[85] Ke Li, Mohammad Nabi Omidvar, Kalyanmoy Deb, and Xin Yao. Variable interaction in multi-objective optimization problems. In Julia Handl, Emma Hart, Peter R. Lewis, Manuel López-Ibáñez, Gabriela Ochoa, and Ben Paechter, editors, *International Conference on Parallel Problem Solving from Nature*, pages 399–409. Springer International Publishing, 2016.

[86] Longmei Li, Hao Chen, Jun Li, Ning Jing, and Michael Emmerich. Preference-based evolutionary many-objective optimization for agile satellite mission planning. *IEEE Access*, 6:40963–40978, 2018.

[87] Mo Li, Qiang Fu, Ping Guo, Vijay P Singh, Chenglong Zhang, and Gaiqiang Yang. Stochastic multi-objective decision making for sustainable irrigation in a changing environment. *Journal of Cleaner Production*, 223:928–945, 2019.

[88] Xiaodong Li. A non-dominated sorting particle swarm optimizer for multiobjective optimization. In *ACM Genetic and Evolutionary Computation Conference (GECCO)*, pages 37–48, 2003.

[89] Xiaodong Li, Ke Tang, Mohammad Nabi Omidvar, Zhenyu Yang, Kai Qin, and Hefei China. Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization. *IEEE Congress on Evolutionary Computation (CEC)*, 7(33):8, 2013.

[90] Zhengping Liang, Xuyong Wang, Qiuzhen Lin, Fei Chen, Jianyong Chen, and Zhong Ming. A novel multi-objective co-evolutionary algorithm based on decomposition approach. *Applied Soft Computing*, 73:50–66.

[91] Yingbiao Ling, Haijian Li, and Bin Cao. Cooperative co-evolution with graph-based differential grouping for large scale global optimization. In *2016 12th international conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD)*, pages 95–102. IEEE, 2016.

[92] Yingbiao Ling, Haijian Li, and Bin Cao. Cooperative co-evolution with graph-based differential grouping for large scale global optimization. In *International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 95–102, 2016.

[93] Junhua Liu, Yuping Wang, Ninglei Fan, Shiwei Wei, and Wuning Tong. A convergence-diversity balanced fitness evaluation mechanism for decomposition-based many-objective optimization algorithm. *Integrated Computer-Aided Engineering*, 26(2):159–184, 2019.

[94] Thibaut Lust and Jacques Teghem. The multiobjective traveling salesman problem: A survey and a new approach. In Carlos A. Coello Coello, Clarisse Dhaenens, and Laetitia Jourdan, editors, *Advances in Multi-Objective Nature Inspired Computing*, page 119–141. Springer, 2010.

[95] Robert J Lygoe, Mark Cary, and Peter J Fleming. A real-world application of a many-objective optimisation complexity reduction process. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 641–655. Springer, 2013.

[96] Lianbo Ma, Min Huang, Shengxiang Yang, Rui Wang, and Xingwei Wang. An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization. *IEEE Transactions on Cybernetics*, 52(7):6684–6696, 2022.

[97] Xiaoliang Ma, Xiaodong Li, Qingfu Zhang, Ke Tang, Zhengping Liang, Weixin Xie, and Zexuan Zhu. A survey on cooperative co-evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 23(3):421–441, 2018.

[98] Xiaoliang Ma, Fang Liu, Yutao Qi, Xiaodong Wang, Lingling Li, Licheng Jiao, Minglei Yin, and Maoguo Gong. A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables. *IEEE Transactions on Evolutionary Computation*, 20(2):275–298, 2016.

[99] Wouter Maes and Kathy Steppe. Perspectives for remote sensing with unmanned aerial vehicles in precision agriculture. *Trends in Plant Science*, 24(2):152–164, Feb 2019.

[100] Kuntinee Maneeratana, Kittipong Boonlong, and Nachol Chaiyaratana. Multi-objective optimisation by co-operative co-evolution. In *International Conference on Parallel Problem Solving from Nature*, page 772–781. Springer, 2004.

[101] Bruce Maxwell, Paul Hegedus, Philip Davis, Anton Bekkerman, Robert Payn, John Sheppard, Nicholas Silverman, and Clemente Izurieta. Can optimization associated with on-farm experimentation using on-farm experimentation using site-specific technologies improve producer management decisions. In *International Conference on Precision Agriculture*, 2018.

[102] Adriana Menchaca-Mendez and Carlos A Coello Coello. An alternative hypervolume-based selection mechanism for multi-objective evolutionary algorithms. *Soft Computing*, 21(4):861–884, 2017.

[103] Seyedali Mirjalili and Andrew Lewis. The whale optimization algorithm. *Advances in Engineering Software*, 95:51–67, 2016.

[104] Giorgio Morales and John W Sheppard. Two-dimensional deep regression for early yield prediction of winter wheat. In *SPIE Future Sensing Technologies 2021*, volume 11914, pages 31–45, 2021.

[105] Giorgio Morales, John W. Sheppard, Bryan Scherrer, and Joseph A. Shaw. Reduced-cost hyperspectral convolutional neural networks. *Journal of Applied Remote Sensing*, 14(3), Sep 2020.

[106] Kaname Narukawa and Tobias Rodemann. Examining the performance of evolutionary many-objective optimization algorithms on a real-world application. In *2012 Sixth International Conference on Genetic and Evolutionary Computing*, pages 316–319. IEEE, 2012.

[107] Mohammad Nabi Omidvar, Xiaodong Li, Yi Mei, and Xin Yao. Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Transactions on Evolutionary Computation*, 18(3):378–393, 2013.

[108] Mohammad Nabi Omidvar, Xiaodong Li, and Xin Yao. Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1762–1769, 2010.

[109] Mohammad Nabi Omidvar, Ming Yang, Yi Mei, Xiaodong Li, and Xin Yao. Dg2: A faster and more accurate differential grouping for large-scale black-box optimization. *IEEE Transactions on Evolutionary Computation*, 21(6):929–942, 2017.

[110] S.L. Osborne, J.S. Schepers, D.D. Francis, and M.R. Schlemmer. Use of spectral radiance to estimate in-season biomass and grain yield in nitrogen- and waterstressed corn. *Crop Science*, 42:165–171, 2002.

[111] Lucie Pansart, Nicolas Catusse, and Hadrien Cambazard. Exact algorithms for the order picking problem. *Computers Operations Research*, 100:117–127, 2018.

[112] Abhishek Patel. Working of differential evolution. *Medium*, May 2018. `https://medium.com/@b516002/differential-evolution-sounds-cool-right-a5c245cbe6d9`.

[113] Amy Peerlinck, John Sheppard, Julie Pastorino, and Bruce Maxwell. Optimal design of experiments for precision agriculture using a genetic algorithm. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1838–1845, 2019.

[114] Sebastian Peitz and Michael Dellnitz. A survey of recent trends in multiobjective optimal control—surrogate models, feedback control and objective reduction. *Mathematical and Computational Applications*, 23(2):30, 2018.

[115] Guang Peng and Katinka Wolter. A decomposition-based evolutionary algorithm with adaptive weight vectors for multi-and many-objective optimization. In *International Conference on the Applications of Evolutionary Computation*, pages 149–164. Springer, 2020.

[116] Francis J. Pierce and Peter Nowak. Aspects of precision agriculture. volume 67 of *Advances in Agronomy*, pages 1–85. 1999.

[117] Mitchell Potter and Kenneth De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.

[118] Elliott Pryor, Amy Peerlinck, and John Sheppard. A study in overlapping factor decomposition for cooperative co-evolution. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 01–08, 2021.

[119] Bo Yang Qu and Ponnuthurai Nagarathnam Suganthan. Constrained multi-objective optimization algorithm with an ensemble of constraint handling methods. *Engineering Optimization*, 43(4):403–416, 2011.

[120] Aurora Ramírez, José Raúl Romero, and Sebastián Ventura. A survey of many-objective optimisation in search-based software engineering. *Journal of Systems and Software*, 149:382–395, 2019.

[121] William R Raun, John B Solie, Gordon V Johnson, Marvin L Stone, Robert W Mullen, Kyle W Freeman, Wade E Thomason, and Erna V Lukina. Improving nitrogen use efficiency in cereal grain production with optical sensing and variable rate application. *Agronomy Journal*, 94(4):815–820, 2002.

[122] Nery Riquelme, Christian Von Lucken, and Benjamin Baran. Performance metrics in multi-objective optimization. In *Latin American Computing Conference*, page 1–11. IEEE, 2015.

[123] Keith W Ross and Danny HK Tsang. The stochastic knapsack problem. *IEEE Transactions on communications*, 37(7):740–747, 1989.

[124] H. Mert Sahinkoc and Ümit Bilge. A reference set based many-objective co-evolutionary algorithm with an application to the knapsack problem. *European Journal of Operational Research*, 2021.

[125] Hiroyuki Sato, Hernán E. Aguirre, and Kiyoshi Tanaka. Controlling dominance area of solutions and its impact on the performance of MOEAs. In Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, and Tadahiko Murata, editors, *Evolutionary Multi-Criterion Optimization*, pages 5–20, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[126] Eman Sayed, Daryl Essam, and Ruhul Sarker. Dependency identification technique for large scale optimization problems. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2012.

[127] J David Schaffer and Amy Morishima. An adaptive crossover distribution mechanism for genetic algorithms. In *Second International Conference on Genetic Algorithms*, pages 36–40, 1987.

[128] James David Schaffer. *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms (Artificial Intelligence, Optimization, Adaptation, Pattern Recognition)*. PhD thesis, USA, 1984.

[129] Christiaan Scheepers, Andries P Engelbrecht, and Christopher W Cleghorn. Multi-guide particle swarm optimization for multi-objective optimization: Empirical and stability analysis. *Swarm Intelligence*, 13(3):245–276, 2019.

[130] Uferah Shafi, Rafia Mumtaz, José Garcia-Nieto, Syed Ali Hassan, Syed Ali Raza Zaidi, and Naveed Iqbal. Precision agriculture techniques and practices: From considerations to applications. *Sensors*, 19(17), Jan 2019.

[131] Ke Shang, Hisao Ishibuchi, Lie Meng Pang, and Yang Nan. Reference point specification for greedy hypervolume subset selection. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 168–175, 2021.

[132] Min Shi. *An Exploration and Optimization of Cooperative Coevolution.* PhD thesis, Norwegian Institute of Science and Technology, Department of Computer and Information Science, 2012.

[133] W Adam Sigler, Stephanie A Ewing, Clain A Jones, Robert A Payn, EN Jack Brookshire, Jane K Klassen, Douglas Jackson-Smith, and Gary S Weissmann. Connections among soil, ground, and surface water chemistries characterize nitrogen loss from an agricultural landscape in the upper missouri river basin. *Journal of Hydrology*, 556:247–261, 2018.

[134] Kenneth Sorensen. Metaheuristics – the metaphor exposed. *International Transactions on Operations Research*, 01 2013.

[135] Nidamarthi Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[136] Theodor Stewart, Oliver Bandte, Heinrich Braun, Nirupam Chakraborti, Matthias Ehrgott, Mathias Göbelt, Yaochu Jin, Hirotaka Nakayama, Silvia Poles, and Danilo Di Stefano. *Real-World Applications of Multiobjective Optimization*, pages 285–327. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[137] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, Dec 1997.

[138] Shane Strasser, John Sheppard, Nathan Fortier, and Rollie Goodman. Factored evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 21(2):281–293, 2017.

[139] Shane Tyler Strasser. *Factored Evolutionary Algorithms: Cooperative Coevolutionary Optimization with Overlap.* PhD thesis, Gianforte School of Computing, Montana State University, 2017.

[140] Yuan Sun, Michael Kirley, and Saman K Halgamuge. Quantifying variable interactions in continuous optimization problems. *IEEE Transactions on Evolutionary Computation*, 21(2):249–264, 2017.

[141] Yuan Sun, Mohammad Nabi Omidvar, Michael Kirley, and Xiaodong Li. Adaptive threshold parameter estimation with recursive differential grouping for problem decomposition. In *ACM Genetic and Evolutionary Computation Conference (GECCO)*, page 889–896, New York, NY, USA, 2018.

[142] Tomoaki Takagi, Keiki Takadama, and Hiroyuki Sato. Non-dominated solution sampling using environmental selection in EMO algorithms. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–9, 2020.

[143] Ke Tang, Xiaodong Li, P. N. Suganthan, Zhenyu Yang, and Thomas Weise. Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, 2009.

[144] Sunil Thrikawala, Alfons Weersink, Glenn Fox, and Gary Kachanoski. Economic feasibility of variable-rate technology for nitrogen on corn. *American Journal of Agricultural Economics*, 81(4):914–927, 1999.

[145] Ye Tian, Langchun Si, Xingyi Zhang, Ran Cheng, Cheng He, Kay Chen Tan, and Yaochu Jin. Evolutionary large-scale multi-objective optimization: A survey. *Journal of the ACM*, 1(1), 2021.

[146] Ye Tian, Langchun Si, Xingyi Zhang, Ran Cheng, Cheng He, Kay Chen Tan, and Yaochu Jin. Evolutionary large-scale multi-objective optimization: A survey. *ACM Computing Surveys (CSUR)*, 54(8):1–34, 2021.

[147] Van Truong Vu, Lam Thu Bui, and Trung Thanh Nguyen. A competitive co-evolutionary approach for the multi-objective evolutionary algorithms. *IEEE Access*, 8:56927–56947.

[148] United States Department of Agriculture. Agricultural prices. `https://usda.library.cornell.edu/concern/publications/c821gj76b?locale=en`, Jan 2022.

[149] Stefan Van Aelst, Xiaogang Steven Wang, Ruben H Zamar, and Rong Zhu. Linear grouping using orthogonal regression. *Computational Statistics & Data Analysis*, 50(5):1287–1312, 2006.

[150] Frans Van den Bergh and Andries Petrus Engelbrecht. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):225–239, 2004.

[151] Thomas Van Klompenburg, Ayalew Kassahun, and Cagatay Catal. Crop yield prediction using machine learning: A systematic literature review. *Computers and Electronics in Agriculture*, 177, 2020.

[152] David A Van Veldhuizen. Multiobjective evolutionary algorithms: classifications, analyses, and new innovations [ph. d. thesis]. *Department of Electrical and Computer Engineering. Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio*, 1999.

[153] Shanu Verma, Millie Pant, and Vaclav Snasel. A comprehensive review on nsga-ii for multi-objective combinatorial optimization problems. *IEEE Access*, 9:57757–57791, 2021.

[154] Tobias Wagner and Heike Trautmann. Integration of preferences in hypervolume-based multiobjective evolutionary algorithms by means of desirability functions. *IEEE Transactions on Evolutionary Computation*, 14(5):688–701, 2010.

[155] Dongshu Wang, Dapei Tan, and Lei Liu. Particle swarm optimization algorithm: An overview. *Soft Computing*, 22(2):387–408, Jan 2018.

[156] Handing Wang, Licheng Jiao, and Xin Yao. Two_arch2: An improved two-archive algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 19(4):524–541, 2014.

[157] Lyndon While, Lucas Bradstreet, and Luigi Barone. A fast way of calculating exact hypervolumes. *IEEE Transactions on Evolutionary Computation*, 16(1):86–95, 2012.

[158] K. M. Whitley, J. R. Davenport, and S. R. Manley. Differences in nitrate leaching under variable and conventional nitrogen fertilizer management in irrigated potato systems. In P. C. Robert, R. H. Rust, and W. E. Larson, editors, *Proceedings of the 5th International Conference on Precision Agriculture*, pages 1–9, Madison, USA, 2000. American Society of Agronomy.

[159] Gerhard J Woeginger. Exact algorithms for NP-hard problems: A survey. In *Combinatorial Optimization—Eureka, You Shrink! Papers Dedicated to Jack Edmonds 5th International Workshop Aussois, France, March 5–9, 2001 Revised Papers*, pages 185–207. Springer, 2003.

[160] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[161] Stephen J. Wright. Optimization. *Encyclopedia Britannica*, February 2023. `https://www.britannica.com/science/optimization`.

[162] Yi Xiang, Yuren Zhou, Miqing Li, and Zefeng Chen. A vector angle-based evolutionary algorithm for unconstrained many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 21(1):131–152, 2017.

[163] Biao Xu, Yong Zhang, Dunwei Gong, Yinan Guo, and Miao Rong. Environment sensitivity-based cooperative co-evolutionary algorithms for dynamic multi-objective optimization. *IEEE/ACM Transactions on Computational Biology and Bio-Informatics*, 15(6):1877–1890, 2017.

[164] Chonghuan Xu. A big-data oriented recommendation method based on multi-objective optimization. *Knowledge-Based Systems*, 177:11–21, 2019.

[165] Rishika Yadav. Selection of multiple objectives for multi objective optimization in precision agriculture. In *International Conference on Advances in Chemical Engineering*, Nov 2020.

[166] Feng Yang, Liang Xu, Xiaokai Chu, and Shenwen Wang. A new dominance relation based on convergence indicators and niching for many-objective optimization. *Applied Intelligence*, 51(8):5525–5542, 2021.

[167] Ming Yang, Aimin Zhou, Changhe Li, and Xin Yao. An efficient recursive differential grouping for large-scale continuous problems. *IEEE Transactions on Evolutionary Computation*, 2020.

[168] Xin-She Yang and Xingshi He. Firefly algorithm: Recent advances and applications. *International Journal of Swarm Intelligence*, 1(1):36–50, 2013.

[169] Zhenyu Yang, Ke Tang, and Xin Yao. Differential evolution for high-dimensional function optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 3523–3530, 2007.

[170] Zhenyu Yang, Ke Tang, and Xin Yao. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences*, 178(15):2985–2999, 2008.

[171] Xue Yu, Wei-Neng Chen, Tianlong Gu, Huaxiang Zhang, Huaqiang Yuan, Sam Kwong, and Jun Zhang. Set-based discrete particle swarm optimization based on decomposition for permutation-based multiobjective combinatorial optimization problems. *IEEE Transactions on Cybernetics*, 48(7):2139–2153, 2017.

[172] Yuan Yuan, Hua Xu, Bo Wang, Bo Zhang, and Xin Yao. Balancing convergence and diversity in decomposition-based many-objective optimizers. *IEEE Transactions on Evolutionary Computation*, 20(2):180–198, 2016.

[173] Zhaoyu Zhai, José-Fernán Martínez Ortega, Néstor Lucas Martínez, and Jesús Rodríguez-Molina. A mission planning approach for precision farming systems based on multi-objective optimization. *Sensors*, 18(6):1795, Jun 2018.

[174] Maoqing Zhang, Lei Wang, Weian Guo, Wuzhao Li, Junwei Pang, Jun Min, Hanwei Liu, and Qidi Wu. Many-objective evolutionary algorithm based on dominance degree. *Applied Soft Computing*, 113, 2021.

[175] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, Dec 2007.

[176] Qian Zhang, Yanmin Liu, Huayao Han, Meilan Yang, and Xiaoli Shu. Multi-objective particle swarm optimization with multi-archiving strategy. *Scientific Programming*, 2022, 2022.

[177] Xingyi Zhang, Ye Tian, Ran Cheng, and Yaochu Jin. A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 22(1):97–112, 2018.

[178] Shi-Zheng Zhao, Jing J Liang, Ponnuthurai N Suganthan, and Mehmet Fatih Tasgetiren. Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 3845–3852, 2008.

[179] Feifei Zheng, Angus Simpson, and Aaron Zecchin. Improving the efficiency of multi-objective evolutionary algorithms through decomposition: An application to water distribution network design. *Environmental Modelling & Software*, 69:240–252, 2015.

[180] Yu-Jun Zheng, Qin Song, and Sheng-Yong Chen. Multiobjective fireworks optimization for variable-rate fertilization in oil crop production. *Applied Soft Computing*, 13(11):4253–4263, 2013.

[181] Chenwen Zhu, Lihong Xu, and Erik D. Goodman. Generalization of pareto-optimality for many-objective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 20(2):299–315, 2016.

[182] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V.G. da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

[183] Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Department of Computer Science, Swiss Federal Institute of Technology, 1999.

[184] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, Jun 2000.

[185] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. *TIK-report Swiss Federal Institute of Technology*, 103, 2001.

[186] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International Conference on Parallel Problem Solving from Nature*, pages 292–301, 1998.

[187] Eckart Zitzler and Lothar Thiele. An evolutionary algorithm for multiobjective optimization: The strength pareto approach. *TIK-report Swiss Federal Institute of Technology*, 43, 1999.

[188] Djaafar Zouache, Abdelouahab Moussaoui, and Fouad Ben Abdelaziz. A cooperative swarm intelligence algorithm for multi-objective discrete optimization with application to the knapsack problem. *European Journal of Operational Research*, 264(1):74–88, 2018.

APPENDICES

APPENDIX A

MULTI-OBJECTIVE CONTINUOUS BENCHMARK FUNCTIONS

Name	Function
DTLZ1	$g(\mathbf{x}_m) = 100\left(\lvert\mathbf{x}_m\rvert + \sum_{x_i \in \mathbf{x}_m}\left((x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))\right)\right)$ $f_1(\mathbf{x}) = \frac{1}{2}(1 + g(\mathbf{x}_m)) \prod_{i=1}^{m-1} x_i$ $f_2(\mathbf{x}) = \frac{1}{2}(1 + g(\mathbf{x}_m))(1 - x_{m-1}) \prod_{i=1}^{m-2} x_i$ $\vdots$ $f_{m-1}(\mathbf{x}) = \frac{1}{2}(1 + g(\mathbf{x}_m))(1 - x_2)x_1$ $f_m(\mathbf{x}) = \frac{1}{2}(1 - x_1)(1 + g(\mathbf{x}_m))$
DTLZ2	$g(\mathbf{x}_m) = \sum_{x_i \in \mathbf{x}_m}(x_i - 0.5)^2$ $f_1(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \prod_{i=1}^{m-1} \cos(0.5x_i\pi)$ $f_2(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \sin(0.5x_{m-1}\pi) \prod_{i=1}^{m-2} \cos(0.5x_i\pi)$ $\vdots$ $f_m(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \sin(0.5x_1\pi)$
DTLZ3	$g(\mathbf{x}_m) = 100\left(\lvert\mathbf{x}_m\rvert + \sum_{x_i \in \mathbf{x}_m}\left((x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))\right)\right)$ $f_1(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \prod_{i=1}^{m-1} \cos(0.5x_i\pi)$ $f_2(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \sin(0.5x_{m-1}\pi) \prod_{i=1}^{m-2} \cos(0.5x_i\pi)$ $\vdots$ $f_m(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \sin(0.5x_1\pi)$
DTLZ4	$g(\mathbf{x}_m) = \sum_{x_i \in \mathbf{x}_m}(x_i - 0.5)^2$ $f_1(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \prod_{i=1}^{m-1} \cos(0.5x_i^\alpha\pi)$ $f_2(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \sin(0.5x_{m-1}^\alpha\pi) \prod_{i=1}^{m-2} \cos(0.5x_i^\alpha\pi)$ $\vdots$ $f_m(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \sin(0.5x_1^\alpha\pi)$

Table A.1: List of DTLZ multi-objective optimization benchmark functions.

Name	Function
DTLZ5	$g(\mathbf{x}_m) = \sum_{x_i \in \mathbf{x}_m} (x_i - 0.5)^2$ $\theta_i = \frac{\pi}{4(1+g(\mathbf{x}_m))}(1 + 2g(\mathbf{x}_m)x_i$, for $i = 2, 3, \ldots, (m-1)$ $f_1(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \cos(\theta_1 \pi/2) \ldots \cos(\theta_{m-2}\pi/2) \cos(\theta_{m-1}\pi/2)$ $f_2(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \cos(\theta_1 \pi/2) \ldots \cos(\theta_{m-2}\pi/2) \sin(\theta_{m-1}\pi/2)$ $f_3(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \cos(\theta_1 \pi/2) \ldots \sin(\theta_{m-2}\pi/2)$ $\vdots$ $f_m(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \sin(\theta_1 \pi/2)$
DTLZ6	$g(\mathbf{x}_m) = \sum_{x_i \in \mathbf{x}_m} x_i^{0.1}$ $\theta_i = \frac{\pi}{4(1+g(\mathbf{x}_m))}(1 + 2g(\mathbf{x}_m)x_i$, for $i = 2, 3, \ldots, (m-1)$, $f_1(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \cos(\theta_1 \pi/2) \ldots \cos(\theta_{m-2}\pi/2) \cos(\theta_{m-1}\pi/2)$ $\vdots$ $f_m(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \sin(\theta_1 \pi/2)$
DTLZ7	$g(\mathbf{x}_m) = 1 + \frac{9}{\lvert\mathbf{x}_m\rvert} \sum_{x_i \in \mathbf{x}_m} x_i$ $h(f_1, f_2, \ldots, f_{M-1}, g) = m - \sum_{i=1}^{m-1} \left[\frac{f_i}{1+g}(1 + \sin(3\pi f_i)) \right]$ $f_1(\mathbf{x}_1) = x_1$ $f_2(\mathbf{x}_2) = x_2$ $\vdots$ $f_{m-1}(\mathbf{x}_{m-1}) = x_{m-1}$ $f_m(\mathbf{x}) = (1 + g(\mathbf{x}_m)) h(f_1, f_2, \ldots, f_{m-1}, g$

Table A.2: List of DTLZ multi-objective optimization benchmark functions.

APPENDIX B

LARGE-SCALE CONTINUOUS BENCHMARK FUNCTIONS

Name	**Function**
Shifted Ackley's	$F_3(x) = F_{ackley}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)$ - $\exp\left(\frac{1}{D}\cos(2\pi x_i)\right) + 20 + e$
Shifted m-rotated Rastrigin's	$F_5(x) = F_{rot\_rastrigin}[x(P_1 : P_m)] \times 10^6$ + $\mathrm{F}_{rastrigin}x(P_{m+1} : P_D)$ $F_{rastrigin}(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$
$\frac{D}{2m}$ shifted $m$rotated Ackley's	$F_{11}(x) = \sum_{k=1}^{\frac{D}{2m}} F_{rot\_ackley}[x(P_{(k-1)\times m+1} : P_{k\times m})]$ + $\mathrm{F}_{ackley}[x(P_{\frac{D}{2}+1} : P_D)]$
$\frac{D}{m}$ shifted m-rotated Schwefel's 1.2	$F_{17}(x) = \sum_{k=1}^{\frac{D}{m}} F_{schwefel}[x(P_{(k-1)\times m+1} : P_{k\times m})]$ $F_{schwefel}(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_i\right)^2$
Shifted Rosenbrock	$F_{20}(x) = \sum_{i=1}^{D-1}[100(x_i^2 - x_{i+1})^2 + (z_i - 1)^2]$

Table B.1: List of CEC 2010 LSO benchmark functions. Dimension $D = 1000$, group size $m = 50$, P : random permutation of $\{1, 2, \ldots, D\}$, F_{rot} refers to rotation of the variables based on a $D \times D$ orthogonal matrix [143].

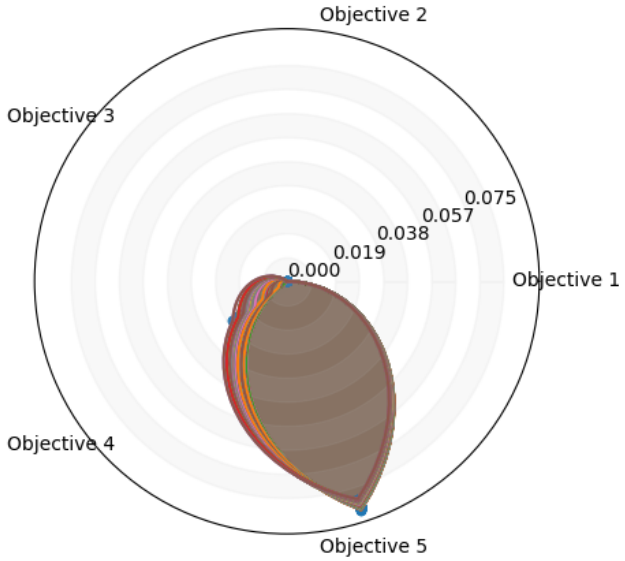
APPENDIX C

SOLUTION SET REDUCTION RADAR GRAPHS

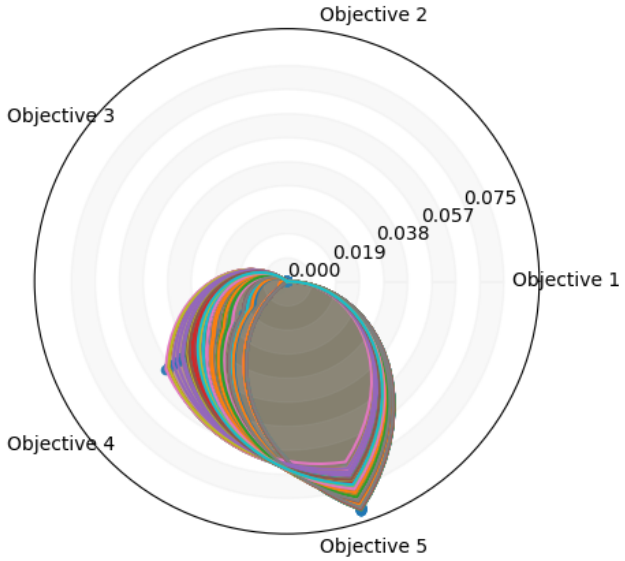
A selection of additional radar graphs for each of the problems. These are graphs generated from a single randomly selected solution set. Since we ran each algorithm 30 times, this is only a small sampling of the total results.

Terminology:

1. OAM: Objective Archive Management
2. ES: Environmental Selection
3. Single: Refers to the one time application of OAM to a solution set
4. External: Refers to the continuous updating of an external OA throughout the optimization process, and using the resulting OA to find overlapping solutions

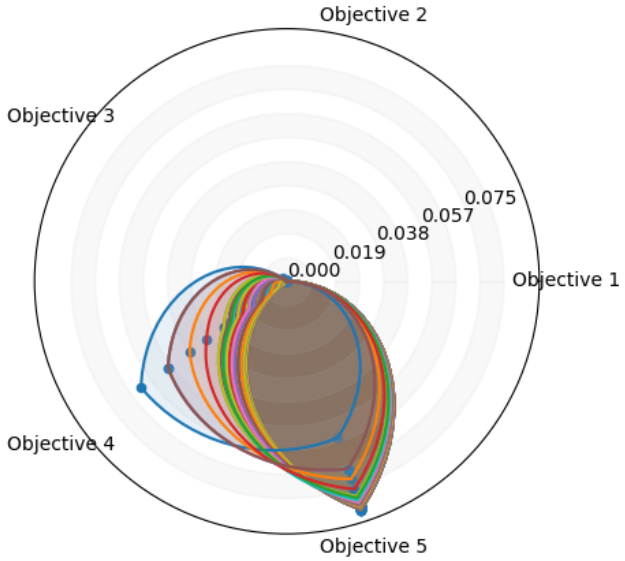


(a) External

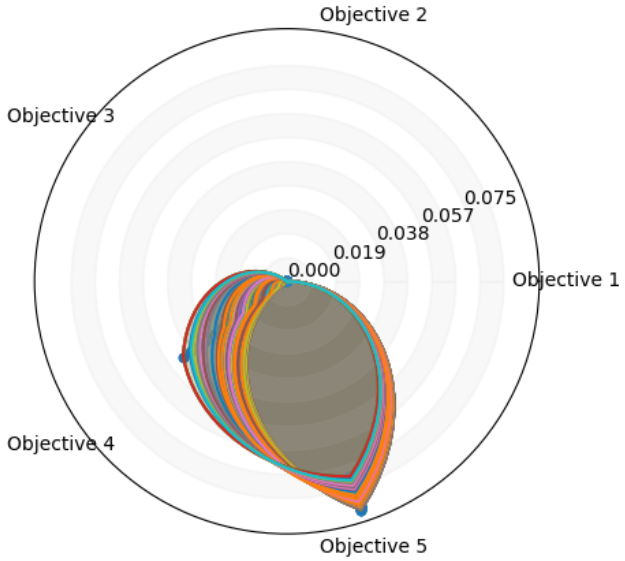


(b) Single

Figure C.1: DTLZ6 5 objectives NSGA2 ES

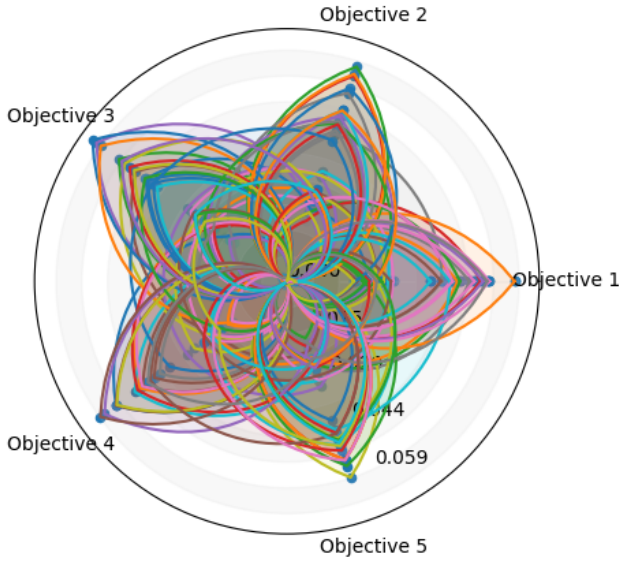


(a) External

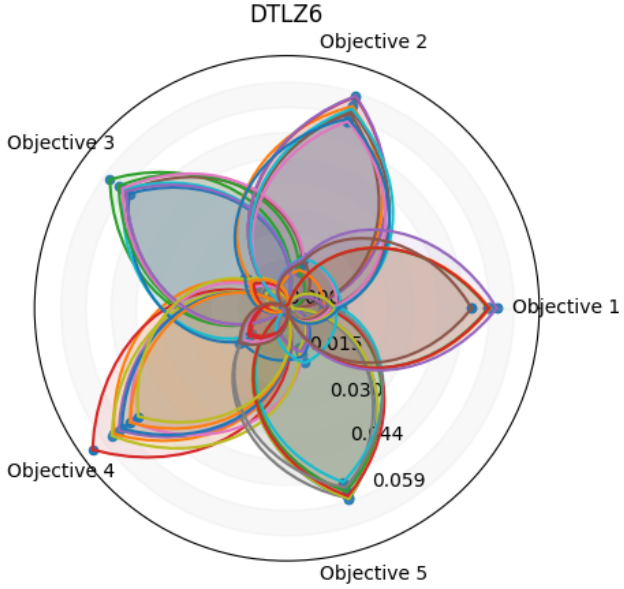


(b) Single

Figure C.2: DTLZ6 5 objectives NSGA2 OAM

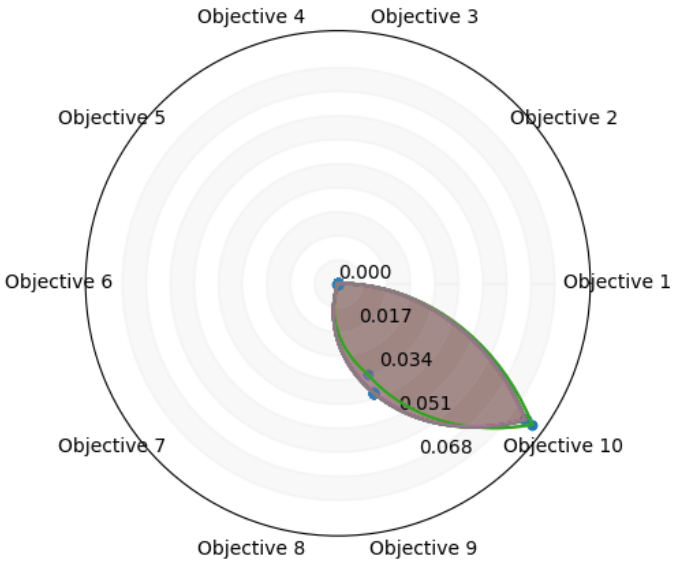


(a) NSGA3

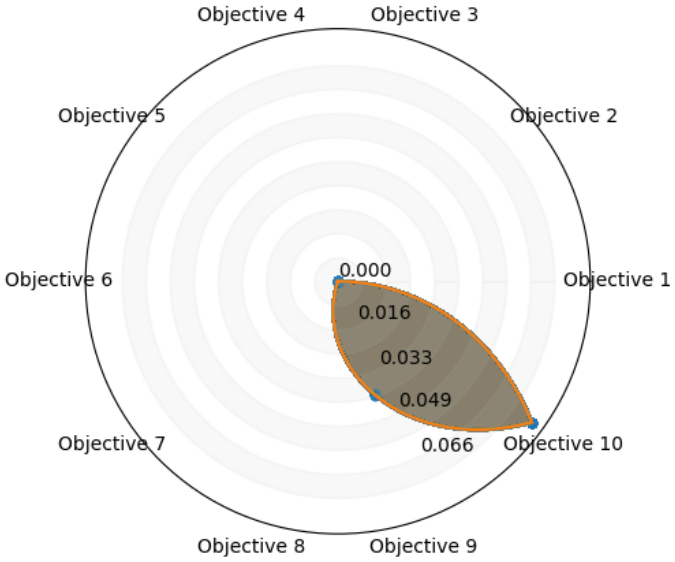


(b) OAM Single

Figure C.3: DTLZ6 5 objectives NSGA3

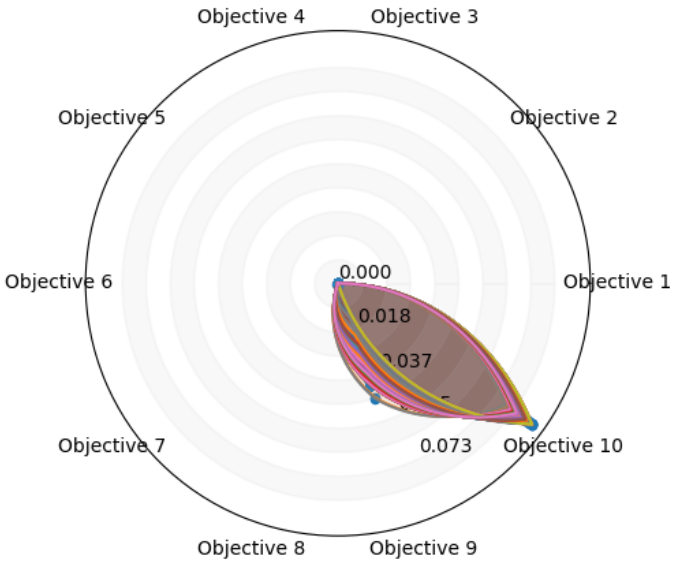


(a) External

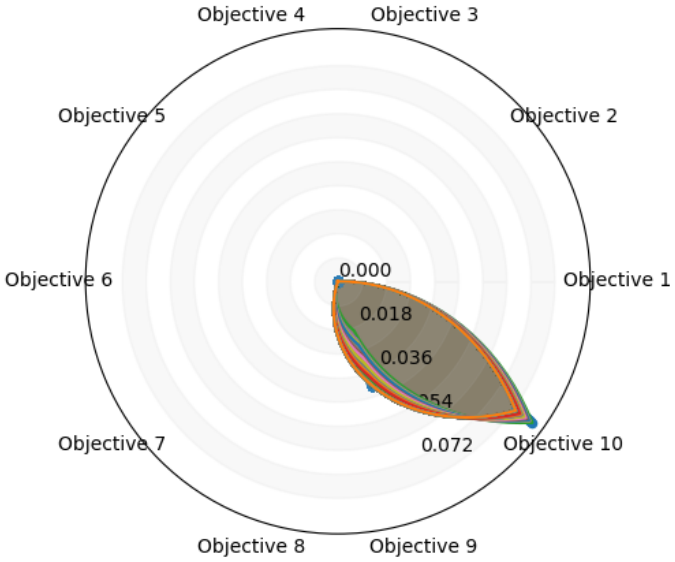


(b) Single

Figure C.4: DTLZ6 10 objectives NSGA2 ES

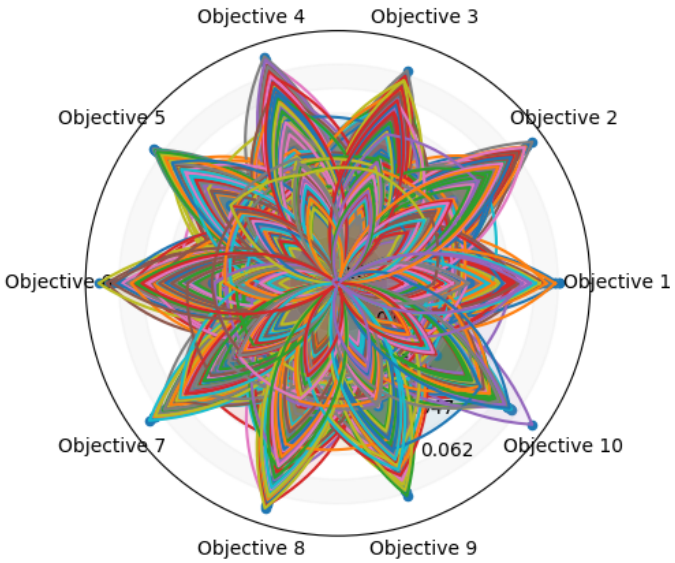


(a) External

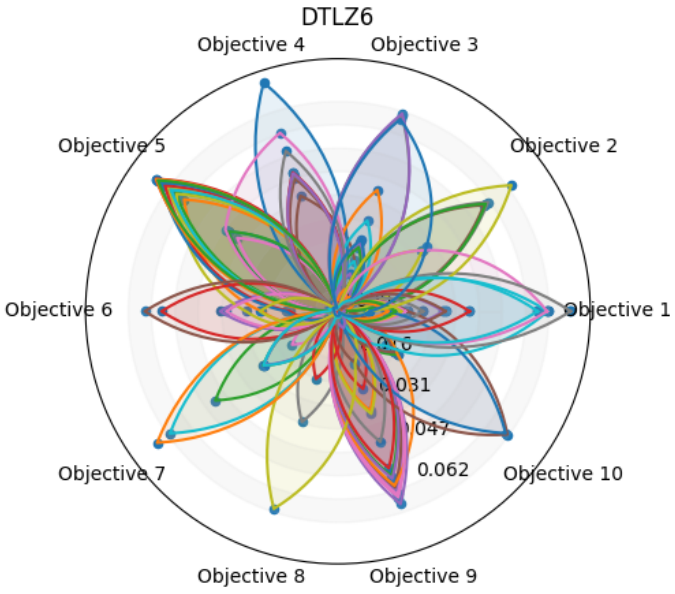


(b) Single

Figure C.5: DTLZ6 10 objectives NSGA2 OAM

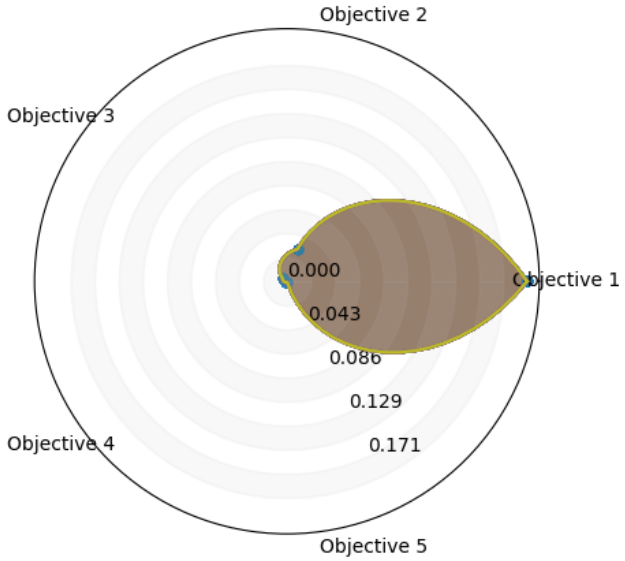


(a) NSGA3

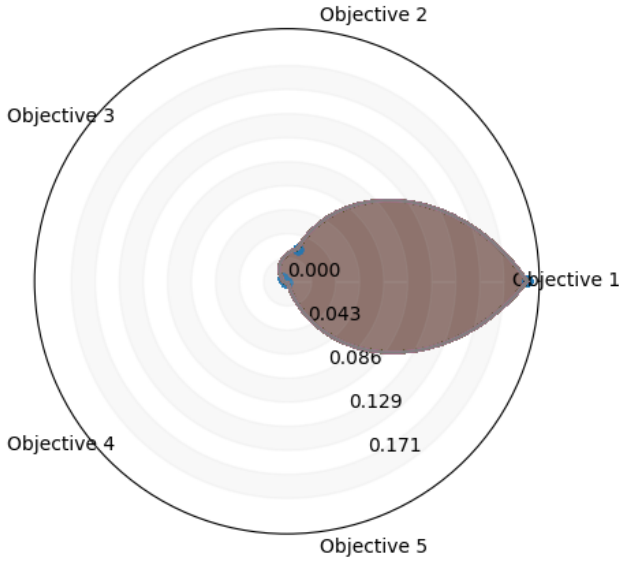


(b) OAM Single

Figure C.6: DTLZ6 10 objectives NSGA3

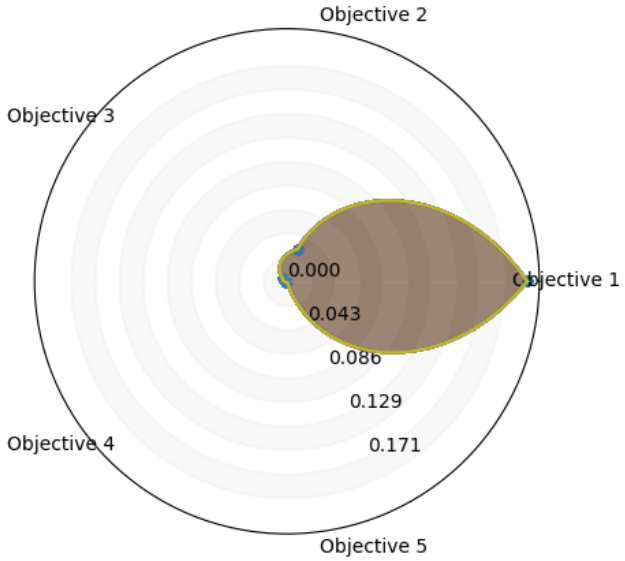


(a) External

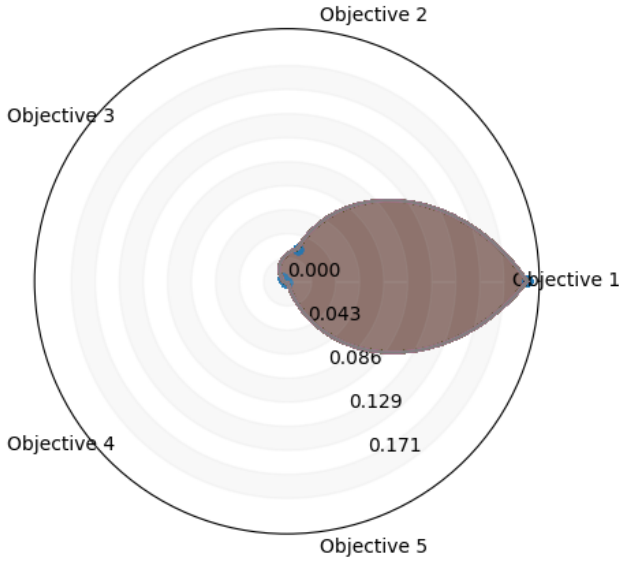


(b) Single

Figure C.7: WFG3 5 objectives NSGA2 ES

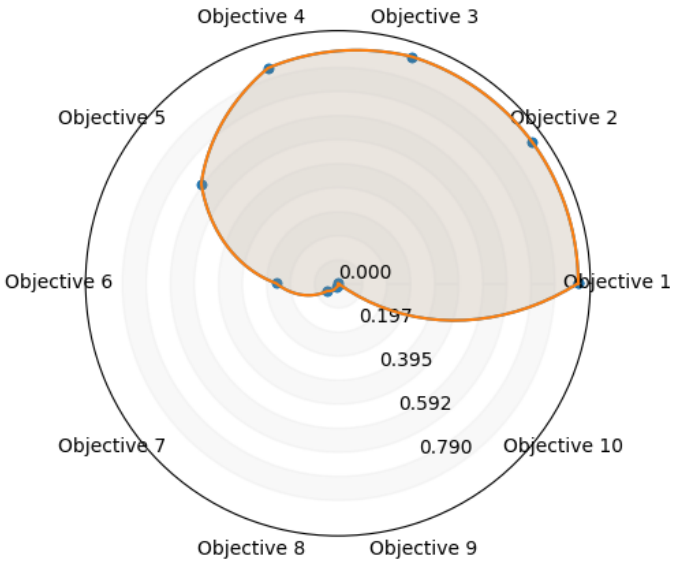


(a) External

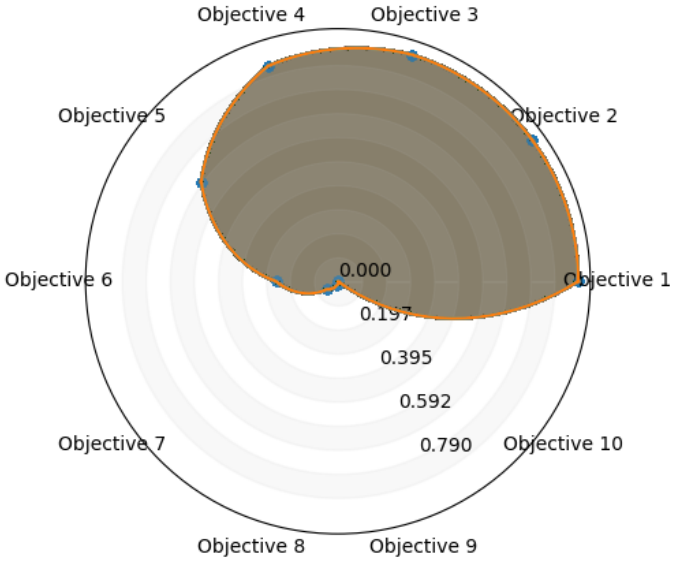


(b) Single

Figure C.8: WFG3 5 objectives NSGA2 OAM

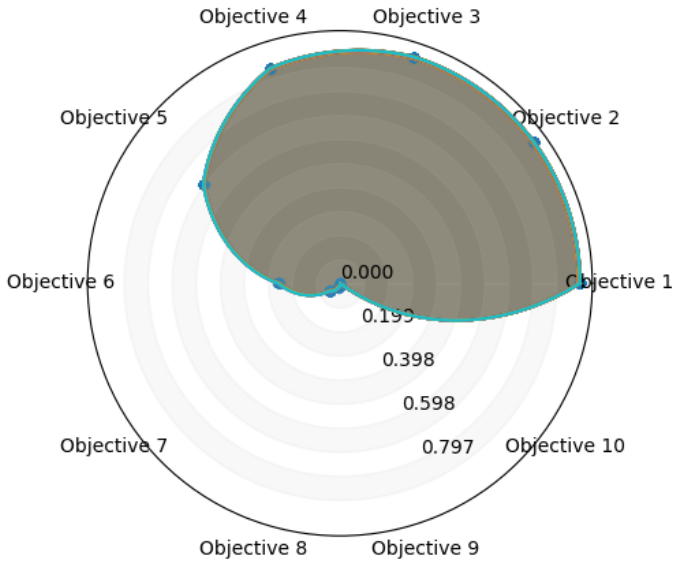


(a) External

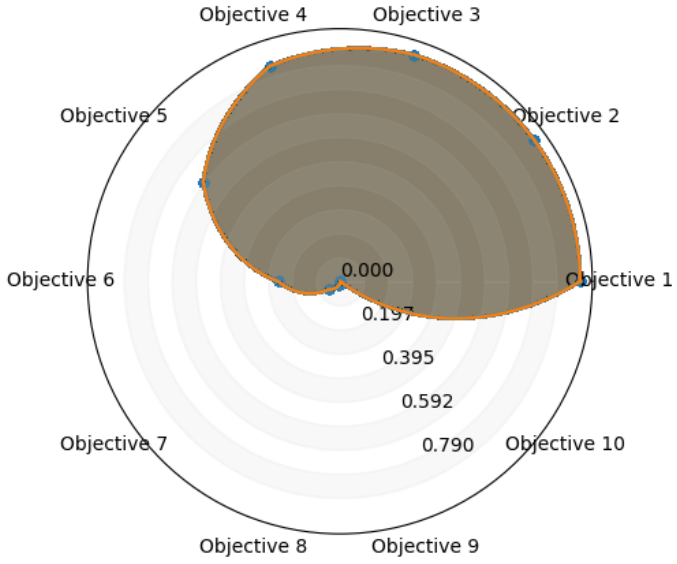


(b) Single

Figure C.9: WFG3 10 objectives NSGA2 ES

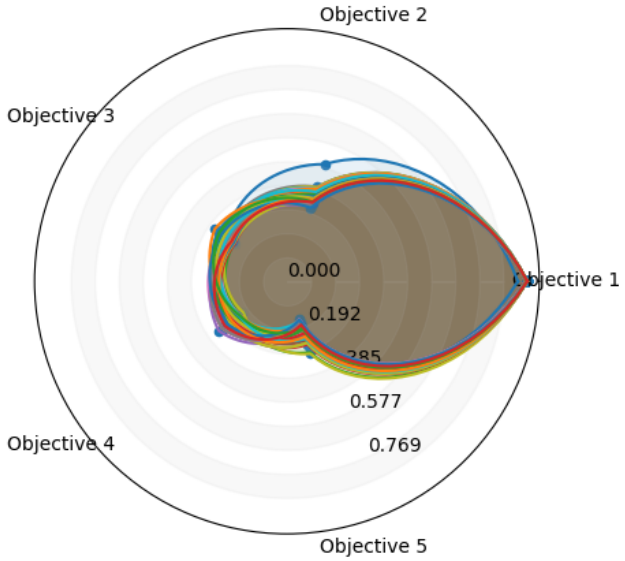


(a) External

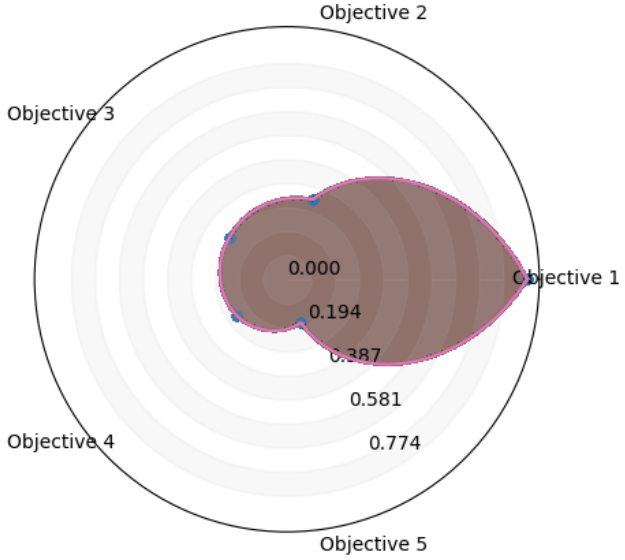


(b) Single

Figure C.10: WFG3 10 objectives NSGA2 OAM

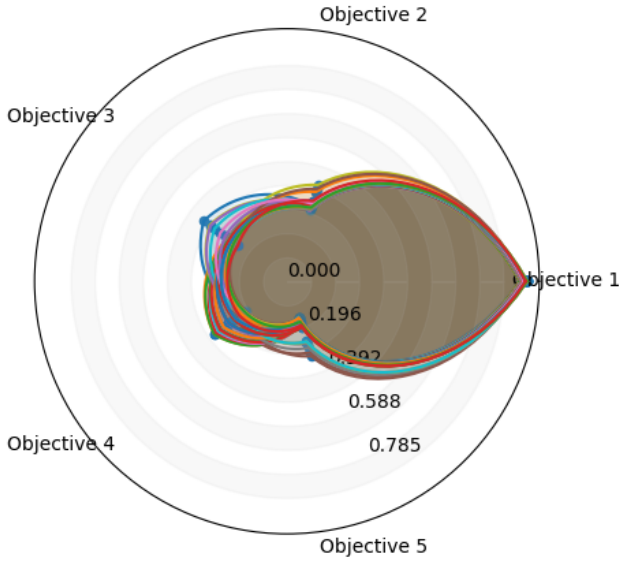


(a) External

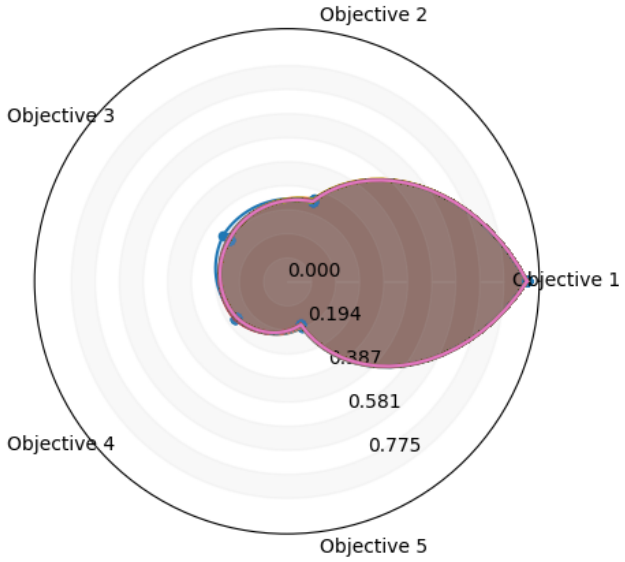


(b) Single

Figure C.11: WFG7 5 objectives NSGA2 ES

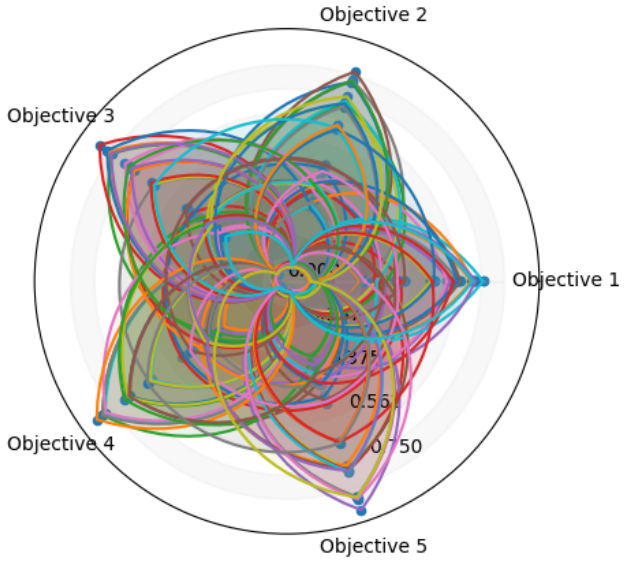


(a) External

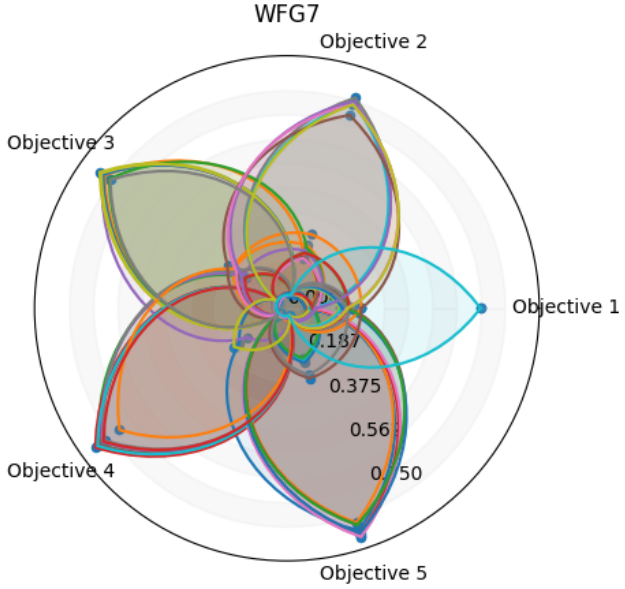


(b) Single

Figure C.12: WFG7 5 objectives NSGA2 OAM

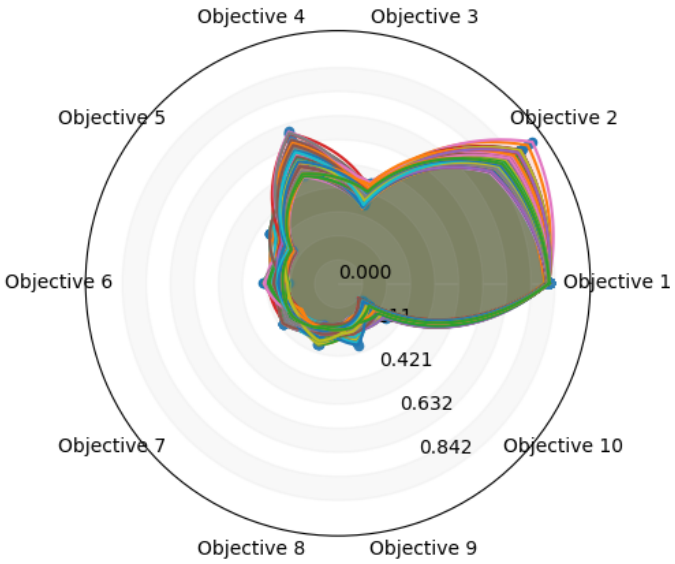


(a) NSGA3

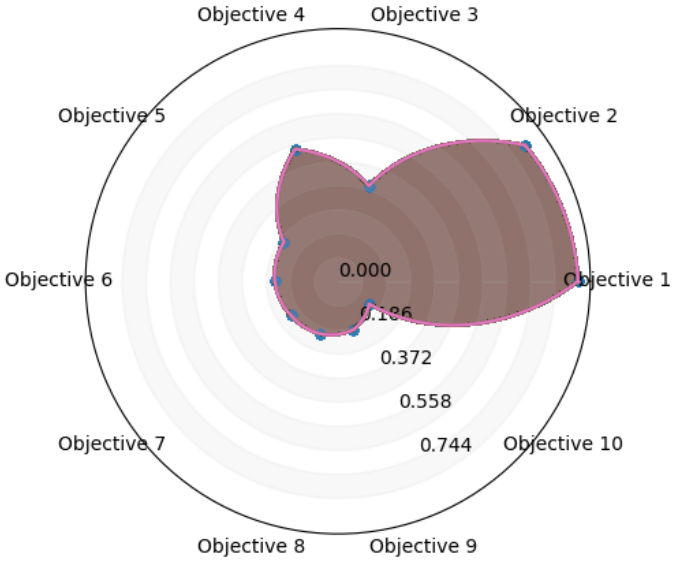


(b) OAM Single

Figure C.13: WFG7 5 objectives NSGA3

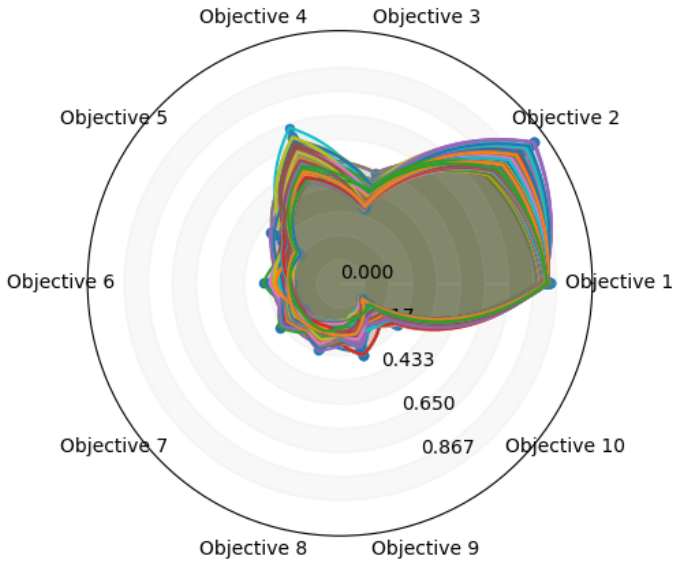


(a) External

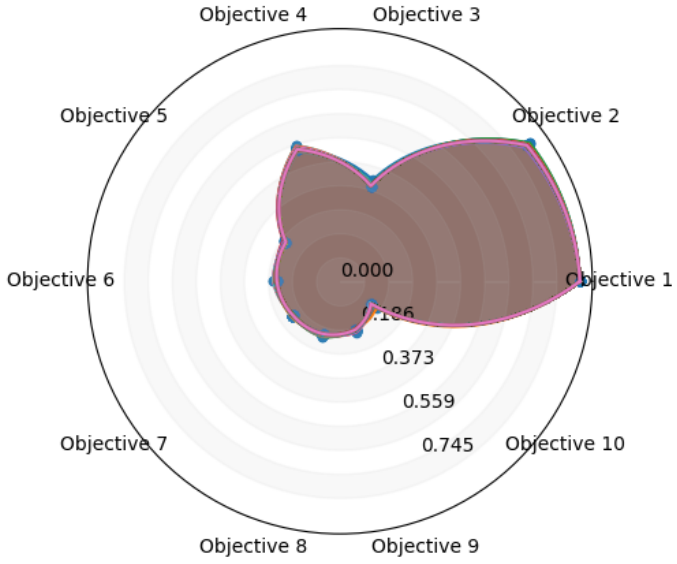


(b) Single

Figure C.14: WFG7 10 objectives NSGA2 ES

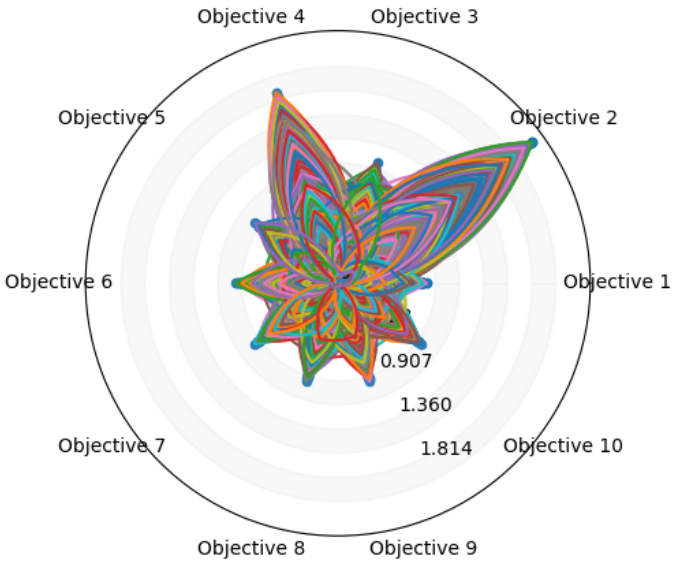


(a) External

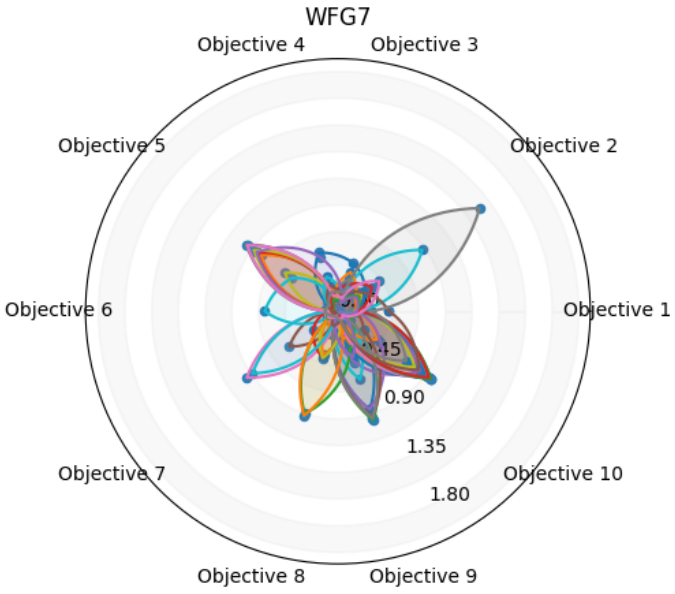


(b) Single

Figure C.15: WFG7 10 objectives NSGA2 OAM



(a) NSGA3



(b) NSGA3 - OAM Single

Figure C.16: WFG7 10 objectives NSGA3