MULTI- AND MANY-OBJECTIVE OPTIMIZATION: SOME ADVANCES TOWARDS THEORETICAL ASPECTS IN PERFORMANCE QUALITY INDICATORS AND EVOLUTIONARY FRAMEWORKS

CLÁUDIO LÚCIO DO VAL LOPES

MULTI- AND MANY-OBJECTIVE OPTIMIZATION: SOME ADVANCES TOWARDS THEORETICAL ASPECTS IN PERFORMANCE QUALITY INDICATORS AND EVOLUTIONARY FRAMEWORKS

Thesis presented to the Graduate Program in Mathematical and Computational Modeling of the CEFET-MG in partial fulfillment of the requirements for the degree of Doctor in Mathematical and Computational Modeling.

Advisor: Flávio Vinícius Cruzeiro Martins
Co-Advisors: Elizabeth Fialho Wanner,
Kalyanmoy Deb

Belo Horizonte

November 2022

Lopes, Cláudio Lúcio do Val
L864m Multi- and many-objective optimization: some advances towards theoretical aspects in performance quality indicators and evolutionary frameworks / Cláudio Lúcio do Val Lopes. – 2022.
196 f.

Tese de doutorado apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional.
Orientador: Flávio Vinícius Cruzeiro Martins.
Coorientadores: Elizabeth Fialho Wanner e Kalyanmoy Deb.
Tese (doutorado) – Centro Federal de Educação Tecnológica de Minas Gerais.

1. Processo decisório por critérios múltiplos – Teses. 2. Otimização matemática – Teses. 3. Algoritmos – Teses. 4. Computação evolutiva – Teses. I. Martins, Flávio Vinícius Cruzeiro. II. Warner, Elizabeth Fialho. III. Deb, Kalyanmoy. IV. Centro Federal de Educação Tecnológica de Minas Gerais. V. Título.

CDD 519.72

Elaboração da ficha catalográfica pela bibliotecária Jane Marangon Duarte, CRB 6ª 1592 / Cefet/MG

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
COORDENAÇÃO DO PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

"MULTI- AND MANY-OBJECTIVE OPTIMIZATION: SOME ADVANCES TOWARDS THEORETICAL ASPECTS IN PERFORMANCE QUALITY INDICATORS AND EVOLUTIONARY FRAMEWORKS".

Tese de Doutorado apresentada por **Cláudio Lúcio do Val Lopes**, em 30 de novembro de 2022, ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG, e aprovada pela banca examinadora constituída pelos professores:

Prof. Dr. Flávio Vinícius Cruzeiro Martins
Centro Federal de Educação Tecnológica de Minas Gerais

Prof. Dr. Kalyanmoy Deb (Coorientador)
Michigan State University

Profª. Drª. Elizabeth Fialho Wanner (Coorientadora)
Centro Federal de Educação Tecnológica de Minas Gerais

Prof. Dr. Ricardo Hiroshi Caldeira Takahashi
Universidade Federal de Minas Gerais

Prof. Dr. Carlos Manuel Mira da Fonseca
Universidade de Coimbra

Profª. Drª. Elisângela Martins de Sá
Centro Federal de Educação Tecnológica de Minas Gerais

Prof. Dr. Adriano Chaves Lisboa
Centro Federal de Educação Tecnológica de Minas Gerais

Visto e permitida a impressão,

Profª. Drª. Elizabeth Fialho Wanner
Presidenta do Colegiado do Programa de Pós-Graduação em Modelagem Matemática e Computacional

Acknowledgments

In Brazil, starting a new technology-based business and doing quality research are two very arduous and complex tasks. Doing both simultaneously can be seen as impossible or something completely irrational. A few years ago, I made this decision. I am sure now that it is not impossible, solely and exclusively, because many people contributed throughout this period.

Flávio and Elizabeth, my advisors, I thank them for their trust, for posing and encouraging me to solve grand challenges. I have always had their presence and guidance by my side. We have had failures, successes, and many discussions. Thank you for the incredible journey until now. I have learned a lot from you, and I am still learning. You have always encouraged and offered me great opportunities.

In the middle of the course, I had the opportunity to present an idea to Prof. Kalyanmoy Deb. An author who I had already read several works, and I must confess, for me, a living legend in optimization and computing area. Observing how Deb works, listening, discussing with him, and implementing our ideas has been a great honor. There is no way to express my gratitude and admiration for his work and person.

In my qualification, I had another opportunity. I met Prof. Carlos Fonseca, whom I must thank for the extensive review of the preliminary text of my thesis; his contributions to this work were fundamental to its final quality. A great honor, once again, to be able to work with a leading researcher in our area.

Thanks to my parents, sister, and niece. My sincerest and deepest gratitude to my wife, Michelle, and my son, Francisco. You are my vital force. The hours of absence on weekends, parties, and holidays have always been to benefit our family. I know you understand my journey, and your unconditional love and support never made me doubt what I needed to do.

To colleagues and partners at A3Data, your support was important at this stage. I also thank PPGMMC and CEFET-MG staff for their support during these four years.

Abstract

Many-Objective Optimization (MaO) refers to optimization problems having four or more objectives, the increase in objective dimensionality brings some complex issues, such as the ineffectiveness of the Pareto dominance relation, quality indicators calculation, solution sets visualization, balancing convergence and diversity, and others.

A key issue in many- and multi-objective optimization is comparing and assessing solution sets obtained by optimization algorithms. This is not a simple task; the outcome of many-objective optimization algorithms is typically a set of incomparable solutions. Using a quality indicator to reflect the inner Pareto front characteristics requires careful design/selection of such indicators. In the first part of this thesis, we deal with the Dominance move (DoM) quality indicator. We propose novel approaches to calculate DoM using mixed-integer programming (MIP) models and an approximate method using machine learning techniques. In general, our attempts uses the dominance move quality indicator as a suitable way to measure, compare, and assess many-objective problems.

Another challenge in MaO is to provide a true representative set with the desired number of Pareto-optimal solutions in a reliably well-distributed set. In the second part of this work, we propose a multi-stage framework involving reference-vector-based evolutionary multi- and many-objective algorithms that attempt to rectify previous stages' shortcomings by careful executions of subsequent stages so that a prescribed number of well-distributed and well-converged solutions are achieved.

The results presented in this thesis come from the attempts to address challenges in evolutionary many- and multi-objective optimization. This research has analyzed and systematically evaluated existing methods. It has also extended them in innovative directions related to quality indicators and improvements concerning the multi-stage approach in balance convergence and diversity in evolutionary algorithms.

Keywords: Multi- and many-objective optimization, quality indicators, evolutionary algorithms.

List of Figures

List of Tables

Contents

Part I

Context

Chapter 1

Introduction

1.1 Context

In our daily lives, we face common decisions: buying a car, choosing an internet provider, making a course, taking a route to go to work, and others. The options available have some characteristics, and they can be assessed from many perspectives: comfort, monetary, utility, speed, distance, and quality, to name a few. Typically, these options are conflicting, and we have to decide and find a trade-off among the alternatives in these situations.

When two or three objective functions are optimized simultaneously, we deal with multi-objective optimization problems (MOPs). When the number of objectives equals or exceeds four, we deal with many-objective optimization problems (MaOPs). Real-world optimization problems are composed of multiple and conflicting objectives. Therefore, we usually want to either minimize and maximize some objective functions concurrently.

In general, an improved solution for one objective function often means a worse solution for another objective function. Due to the conflicting nature of the objective functions, there is no single optimal solution but a set of optimal solutions. This set of solutions is referred to as the Pareto optimal solutions. We consider the solutions optimal so that no other solutions in the search space are 'superior' for all objectives considered in the set of optimal solutions.

Although traditional approaches can combine the objective functions into a single one and solve the resulting problem, several multi- and many-objective optimization techniques have proven to be efficient in dealing with the true multi- and many-objective nature of such problems [Chand and Wagner, 2015].

Evolutionary multi-objective (EMO) or many-objective optimization (EMaO)

techniques have been recognized to be well suited for MOPs and MaOPs [Chugh et al., 2019]. They find a set of Pareto-optimal solutions to such problems, generating a solution set as a final result. EMO and EMaO algorithms have been applied in different domains, coming from health sciences and engineering problems, for example, [González-Álvarez and Vega-Rodríguez, 2013; Deb, 2008].

Various approaches have been proposed in the evolutionary optimization community to solve many-objective problems [Li et al., 2018]. However, in contrast to single-objective optimization, the solutions generated by EMO/EMaO algorithms can be tough to obtain and compare. Such difficulty grows as the need for an increasingly large number of candidate solutions and an increase in the number of objectives. When there are two or three objectives only, some graphical techniques visually examine the solution set. However, when the number of objectives is greater than three, this task is challenging, needing more advanced visualization techniques that can show features like location, shape, and the distribution of the solution set [Li et al., 2015a; Ibrahim et al., 2018; Bhattacharjee et al., 2020; Talukder and Deb, 2020].

Quality indicators are suitable for situations when we need to compare two or more solution sets [Zitzler et al., 2003]. They have been used to compare the outcomes of multi- and many-objective algorithms. In Li and Yao [2019], 100 quality indicators are discussed. The paper considers some state-of-the-art indicators with diverse characteristics and focuses on which quality aspects they have, highlighting their strengths and weaknesses.

There is no quality indicator able to represent all the solution set features: *cardinality*, *convergence*, *spread*, and *uniformity*. It is an open question to find if such a quality indicator exists. An additional difficulty is related to the quality indicator computation; hypervolume is an example in which hard efforts have been made related to its calculation on high dimensions [Guerreiro et al., 2021].

Additionally to the quality indicators issue, there are several known shortcomings of EMO and EMaO algorithms which have been recognized in the literature in achieving adequate convergence and distribution of non-dominated solutions: I) EMO and EMaO algorithms are stochastic, and a single application may not always produce a well-distributed and well-converged set of non-dominated (ND) solutions, II) some algorithms are expected to produce a pre-specified number of non-dominant (ND) solutions, but every EMO or EMaO run may not produce the exact number of ND points as desired, and III) a set of ND solutions may indicate specific gaps in the apparent ND front discovered by an EMO or EMaO algorithm, the such gap may be artificial (from discovering fewer points in a particular area of the solution set), or a gap may genuinely exist in the efficient front.

In fact, it would be desired if the EMO/EMaO algorithms could find a more reliable and reproducible ND set having exactly the desired number of a uniformly distributed set of points on the entire efficient set with a clear indication of true gaps and holes, if any.

Our general goal here is to discuss and present some novel proposals to deal with these challenges. In this chapter, firstly, we describe the motivation behind this research. Next, we outline the text structure, and finally, we detail some publications generated from this work.

1.2 Motivation

The research motivation starts based on the quality indicators used in MaOPs. They typically map a solution set to a real number and should reveal some features of the solution set. The common use of quality indicators is to compare results from multi- or many-objective algorithms. Additionally, it can be applied in other scenarios as well. For example, it can be used in search strategies or to improve the individual selection criteria during the evolutionary process.

There are many indicators available, and they have been used in numerous situations in literature [Li and Yao, 2019]. The hypervolume (HV) [Deng and Zhang, 2019; Yang et al., 2019; Bradford et al., 2018; Zitzler and Thiele, 1999], inverted generational distance plus (IGD+) [Ishibuchi et al., 2015b], and ϵ-indicator [Zitzler et al., 2003], generational distance (GD) [Ishibuchi et al., 2015b], and KKTPM [Deb and Abouhawwash, 2016] are just some examples.

HV, IGD+ and ϵ-indicator are commonly used in many-objective problems. Nonetheless, some handicaps can be observed:

- Indicators which are based on Pareto dominance relation typically return similar results of two solution sets with high dimensions;

- HV is hard to exactly calculate in many-objective problems [Guerreiro et al., 2021] . It is still very sensitive to extreme points [Ishibuchi et al., 2018];

- In many-objective test problems and real cases, obtaining a Pareto front or a reference set can be difficult. This information is vital to IGD+, for example; The same happens to HV, in [Ishibuchi et al., 2018], in which is necessary to specify a reference point;

- Essentially, a quality indicator applied in MaOPs must have no severe information loss. In other words, consider as much information as possible from all objectives and solutions.

Based on these observations, some provocative and ambitious questions arise when thinking about many-objective scenarios. Our research questions and hypothesis are:

1. What are the quality indicators suitable for dealing with problem sets in high dimensional space, avoiding the drawbacks previously related?

2. What features (geometrical properties, distributions, and others) characterize the solution sets and quality indicators in high dimensional spaces?

3. Given two solutions sets $\boldsymbol{P}$ and $\boldsymbol{Q}$ in high dimensional space, is it possible to compare them using as much as possible available information? If it is possible, can it be done in a reasonable time (computational complexity)?

These questions are crucial in this work. The first part of this thesis tries to bring innovative approaches to address these challenges in the evolutionary many-objective optimization context. Our main objective is to assess many-objective solution sets using a quality indicator, the Dominance Move, dubbed DoM. It brings up the dominance move concept and utilization. It also proposes efficient ways to calculate it in a reasonable computational time.

In the second part of this work, we continue to deal with MOPs and MaOPs. Performance quality indicators measure convergence and uniformity since that aspects are relevant in EMO and EMaO area. Our focus is on a fundamental evolutionary algorithm design question: balancing convergence and diversity. Some algorithms put more attention on maintaining diversity, for example. However, it may hamper the convergence to the optimal Pareto front, especially for some hard MOPs/MaOPs instances. To alleviate this fact, some multi-phase or multi-stage algorithms have been proposed, such as MOEA/D-AWA [Qi et al., 2014], B-NSGA-III [Seada et al., 2019], CMOEA-MS [Tian et al., 2021b], CLIA [Ge et al., 2019], RVRL-EA [Ma et al., 2021], MSEA [Tian et al., 2021a], and others. This approach generally divides the optimization process into a certain number of stages and applies different selection strategies in these stages. In this way, the diversity performance of the evolutionary algorithm is improved.

Similarly, it is well known that the usage of reference vectors (RV's) in EMO/EMaO helps to guide the evolutionary process towards convergence and uniformity [Deb and Jain, 2014; Zhang and Li, 2007].

Considering such context, we would like to push the boundaries in evolutionary algorithm design. With MOPs and MaOPs we would like to investigate:

1. Is there a non-invasive method that can be coupled with RV's based EMO/EMaO algorithms bringing benefits (enhancing convergence and diversity) for this 'new method'?

2. Is there a reliable method to treat some hard problem instances, dealing with natural or 'artificial' gaps?

3. Is there an approach to get precisely N well-distributed non-dominant solutions?

More importantly, it does not treat each of the mentioned research questions isolated but how they can be treated simultaneously, coupling all these features in current and most used RV's based EMO and EMaO algorithms.

Our Multi-Stage Reference-vector-based Framework can find exactly N uniformly distributed solutions with a straightforward process to deal with true gaps and holes using current EMO or EMaO algorithms. Essentially, we use a reference vector-based EMO/EMaO algorithm in a non-invasive manner with a multi-stage approach coupled with a procedure to make global adjustments in the reference vectors.

Steps towards a better understanding of performance quality indicators and improving the balance between convergence and uniformity are our pillars. We believe that such contribution helps to advance the current evolutionary multi- and many-objective approach, allowing us to tackle more challenges.

1.3 Thesis structure

The thesis is organized as follows.

Chapter 2 provides the necessary background material. It introduces multi- and many-objective optimization with evolutionary algorithms and some contextualized relevant concepts. It also discusses the role of quality indicators in this scenario, with some commonly used indicators found in the literature (focused on convergence and uniformity). Some test problems are also listed and presented. Another section is related to the mathematical programming area, furthering the mixed-integer programming and some concepts and classical problems. In the last section, machine learning

as a concept is also discussed. We are focusing on unsupervised learning techniques related to clustering tasks. In general, all these concepts and methods presented in this chapter are introduced and are used as basic 'blocks' to solve problems and extend EMO and EMaO shortcomings.

In the first thesis' part, we will present and discuss the Dominance Move (DoM) concept as a viable way to assess, compare, and analyze results produced by EMO and EMaO algorithms. In Chapter 3 DoM is introduced as a performance quality indicator, and the benefits and drawbacks are also presented. Until now, DoM was calculated just for bi-objective problems. We propose a novel approach to calculate DoM using a mixed-integer linear programming (MIP) approach, which can handle solution sets in multi- and many-objective problems. Experiments using three to 30-objective problems are performed using 50 to 400 solutions in the set to show how the model behaves in higher-dimensional cases. Algorithms such as IBEA, MOEA/D, NSGA-III, NSGA-II, and SPEA2 are used to generate the solution sets. Further extensions are discussed to handle particular idiosyncrasies with some solution sets and improve the quality indicator and its use in other situations.

Next, in Chapter 4 and 5 the computational time spent by the former model will be assessed and improved, enabling broader use of the DoM. In Chapter 4 a compact mixed-integer formulation will be presented; it is based on a different perspective adopted in Chapter 3 (initially proposed by Prof. Carlos Fonseca). Some experiments will show that this compact formulation is able to calculate DoM; we also compare the former and the compact models. The compact one presents better computational time and is much more straightforward in its formulation. However, there are still some limits to the use of the compact formulation, mainly related to the number of solutions in each set.

In Chapter 5 we propose an approximation method that joins our mixed-integer programming model with a machine learning clustering technique. It brings an interesting tradeoff between DoM value precision and the time spent by the method. Lastly, we test some limits, using hard instance problems that take hours to be computed by the compact formulation. Our approximate method is able to transpose such limitations. This chapter closes our first thesis part with insights into the practical use of DoM and different ways to calculate it in MOP and MaOP scenarios.

The next part of this thesis treats the balance between convergence and diversity in evolutionary multi-objective and many-objective (that is a question that DoM, as a concept, tries to address as well). Since EMO and EMaO algorithms are stochastic, a single application may not provide a true representative set with the desired number of Pareto-optimal solutions reliably and, importantly, with a well-

distributed set of solutions. In Chapter 6, we propose a multi-stage framework involving reference-vector-based evolutionary multi- and many-objective algorithms (MuSt-EMO and MuSt-EMaO) that attempts to rectify shortcomings of previous stages. By carefully executing subsequent stages, an exact prescribed number of well-distributed and well-converged solutions are achieved at the end. The working of the proposed Must-EMO/EMaO algorithms is implemented in a number of popular reference-based EMO/EMaO algorithms and is demonstrated working on various multi- and many-objective tests and real-world problems.

In Chapter 7, we summarise the work presented in this thesis and look at how it has contributed to the field of evolutionary multi- and many-objective optimization. Additionally, some suggestions are provided as comments and directions for future research.

The Appendix A brings additional experiments and further information about our multi-stage framework presented in Chapter 6.

1.4 Publications

The work resulting from this thesis has been published in the following paper:

- Claudio. L. d. V. Lopes, Flávio. V. C. Martins and Elizabeth. F. Wanner, An Assignment Problem Formulation for Dominance Move Indicator, 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, United Kingdom, 2020, pp. 1-8, doi: 10.1109/CEC48606.2020.9185597.

- Claudio. L. d. V. Lopes, Flávio. V. C. Martins and Elizabeth. F. Wanner, and Kalyanmoy Deb. An Approximate MIP-DoM Calculation for Multi-objective Optimization using Affinity Propagation Clustering Algorithm, 2021 The Genetic and Evolutionary Computation Conference (GECCO), Lille, France 2021.

- Claudio. L. d. V. Lopes, Flávio. V. C. Martins, Elizabeth. F. Wanner and K. Deb, Analyzing Dominance Move (MIP-DoM) Indicator for Multi-and Many-objective Optimization, in IEEE Transactions on Evolutionary Computation, doi: 10.1109/TEVC.2021.3096669.

- Claudio. L. d. V. Lopes, Flávio. V. C. Martins and Elizabeth. F. Wanner, and Kalyanmoy Deb. A Computationally Fast but Approximate MIP-DoM Calculation for Multi-Objective Optimization, 2022 The Genetic and Evolutionary Computation Conference (GECCO), Boston, USA 2022.

The work was presented at this conference:

- Carlos M. Fonseca, Claudio. L. d. V. Lopes, Flávio. V. C. Martins and Elizabeth. F. Wanner, and Kalyanmoy Deb. Calculating Dominance Move using mixed-integer programming: a new compact formulation, 2022, 26th International Conference on Multiple Criteria Decision Making, Portsmouth, UK, 2022.

The following work has been submitted:

- Kalyanmoy Deb , Claudio. L. d. V. Lopes, Flávio. V. C. Martins, and Elizabeth. F. Wanner. 'Identifying Efficient Fronts Reliably Using a Multi-Stage Reference-vector-based Framework' which has been submitted for consideration for publication in the IEEE Transactions on Evolutionary Computation with the ID: TEVC-00242-2022

This other one has been prepared to be submitted:

- Carlos M. Fonseca, Claudio. L. d. V. Lopes, Flávio. V. C. Martins and Elizabeth. F. Wanner, and Kalyanmoy Deb. 'A Compact Formulation for Calculating Dominance Move Using Mixed-Integer Programming' which has been submitted for consideration for publication in the IEEE Transactions on Evolutionary Computation with the ID: TEVC-xxxxx-xxxx

Chapter 2

Background

This chapter reviews relevant topics in multi- and many-objective evolutionary algorithms, mixed-integer programming, and machine learning. Then, these topics' issues are merged and used as the blocks to solve some challenges in multi- and many-objective optimization scenarios.

In the multi- and many-objective evolutionary optimization area, there are some open research questions. It brings some new opportunities, and solving such questions can allow us to extend the approach for applications in complex real-world problems [Li et al., 2015a]. Some common concepts, methods, and shortcomings will be presented as a starting point.

As part of the mathematical programming area, integer programming refers to the class of constrained optimization problems in which some or all of the variables are required to be integers. In most studied and used integer programs, the objective function is linear, and the constraints are also linear. The method has been used in academic and business problems [Jünger et al., 2010]. This section will discuss definitions, terms, and some classical problems in mathematical programming, with a general mixed-integer programming model emphasis.

Machine learning is an artificial intelligence sub-area in which, using experience, computer algorithms learn automatically, [Mitchell, 1997]. It builds a model based on sample data and makes predictions or decisions without being explicitly programmed. We will use some machine learning tasks, such as clustering, in this work. The section will give some contextual concepts that will be used later.

All these topics are vast in literature, and here we present some reviews with fundamental concepts that are most relevant in our work.

This chapter is organized as follows. First, in Section 2.1, we present basic concepts, properties, and suitable techniques to deal with multi- and many-objective evo-

lutionary optimization problems. Then, in Section 2.2, the mathematical programming method is introduced with some mixed-integer programming concepts, as well as some typical problems. Finally, in section 2.3, a machine learning view is presented, further discussing the clustering task.

2.1 Multi- and many-objective optimization

Many real-world optimization problems are composed of multiple and conflicting objectives. Although traditional approaches can combine the objectives into a single one and solve the resulting problem, several multi and many-objective optimization methods have proven to be efficient techniques dealing with the true multiobjective nature of such problems [Chand and Wagner, 2015].

In general, a multi- or many-objective optimization problem (MOP, MaOP, respectively) includes N decision variables, $\mathbf{x}$, from a feasible decision space $\Omega \subseteq \mathbb{R}^N$, and a set of M objective functions. Without loss of generality, the minimization of an MOP can be simply defined as [Yuan et al., 2018]:

$$\text{Minimize} \quad F(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_M(\mathbf{x})]^T, \quad \mathbf{x} \in \Omega. \tag{2.1}$$

$F : \Omega \to \Theta \subseteq \mathbb{R}^M$ is a mapping from the feasible decision space Ω to vectors in the M-dimensional objective space Θ. When $M \geq 4$, the problem is commonly called a many-objective problem.

The possible relations between two or more solutions are essential in a comparison task. The dominance concept plays an indispensable role in this sense [Zitzler et al., 2003]:

Definition 1. *Dominance:*

Consider two vectors, $\boldsymbol{p}$, $\boldsymbol{q} \in \Theta$. $\boldsymbol{p}$ is said to dominate $\boldsymbol{q}$ if $p_m \leq q_m$ for $1 \leq m \leq M$ and necessarily $p_m < q_m$ for at least one m. It is denoted as $\boldsymbol{p} \prec \boldsymbol{q}$.

A vector $\boldsymbol{p} \in \Theta$ is called *Pareto-optimal*, if there is no $\boldsymbol{q} \in \Theta$ that dominates $\boldsymbol{p}$. Such solutions constitute a Pareto-optimal front in the objective space. We are interested in evaluating these objective vectors.

The result of a MOP or MaOP is a solution set, and we want to compare each one. Therefore, it is possible to state the following definition considering solution sets:

Definition 2. *Dominance of sets:*

a set $\boldsymbol{P}$ is said to dominate another set $\boldsymbol{Q}$, if every member of $\boldsymbol{Q}$ is dominated by at least one member of $\boldsymbol{P}$. It is denoted as $\boldsymbol{P} \prec \boldsymbol{Q}$.

In addition to the dominance definition, there are other relations found in literature, such as better dominance, strict and weak dominance [Kaisa, 1999; Zitzler et al., 2003; Branke et al., 2008].

EMO researchers have been rapidly increasing interest in the use of evolutionary algorithms on MaOPs, focusing efforts in algorithm analysis and design, empirical studies, visualization, performance indicators, real-world application, and others, [Li et al., 2015b]. These enable many-objective optimization to become one of the most active research topics in the EMO area.

2.1.1 Evolutionary multi- and many-objective optimization

There are different ways to deal with multi- and many-objective optimization problems, e.g., ϵ-constraint method [HAIMES, 1971], method of global criterion [Yu, 1973] and weighting method [Kaisa, 1999]. Evolutionary optimization is one of the methods available and used to treat multi- and many-objective problems (EMO, and EMaO, respectively). They have shown certain success in finding well-converged and well-diversified non-dominated solutions [Deb, 2001].

Evolutionary or population-based optimization methods collect information from the objective functions at more than one point in the solution space; each one is called an individual. The individuals evolve towards the next generation through the information collected, using evolutionary procedures that mimic evolution concepts in nature, such as selection, mutation, and crossover.

Some successful algorithms using this inspired evolutionary approach include, for example, the strength Pareto evolutionary algorithm: SPEA2 [Zitzler et al., 2001]; non-dominated sorting genetic algorithm: NSGA-II [Deb et al., 2002] and NSGA-III [Jain and Deb, 2014; Deb and Jain, 2014]; the multi-objective evolutionary algorithm based on decomposition: MOEA/D [Zhang and Li, 2007]; indicator-based evolutionary algorithm: IBEA [Zitzler and Künzli, 2004]; and many others.

There are two main goals in the most evolutionary algorithms [Deb, 2001]: find a set of non-dominated solutions that define a Pareto-optimal frontier, or an adequate approximation of this frontier; and, simultaneously, find a set of non-dominated solutions with diversity, which means a good approximation able to represent the entire range covered by the Pareto-optimal boundary.

2.1.2 Quality indicators

Comparing two or more solutions is a common task in EMO/EMaO area. Quality performance indicators have been used in this sense, and hence considerable efforts have been made to define different quality performance indicators [Li and Yao, 2019]. The indicators can compare the outcomes of multi- and many-objective algorithms or even assist an algorithm during the search process for non-dominated candidates.

Additionally, it is paramount to make more precise statements when applying a quality indicator comparison; for example, if one algorithm is better than another, how much better is it? The following definition formalizes the quality indicators [Zitzler et al., 2003]:

Definition 3. *Quality indicator: An k-ary quality indicator I is a function I:*$\Theta^k \rightarrow \mathbb{R}$*, which assigns each vector of k objective vectors (*$\boldsymbol{P_1}$*,* $\boldsymbol{P_2}$*, ...,* $\boldsymbol{P_k}$*) a real value I(*$\boldsymbol{P_1}$*,* $\boldsymbol{P_2}$ *,..., *$\boldsymbol{P_k}$*).*

The quality indicators can be unary, binary, or k-ary, defining a value to one solution set, two solution sets, or k solution sets, respectively.

It is expected that quality indicators can be easily interpretable also [Zitzler et al., 2003], e.g., if $\boldsymbol{P}$ dominates some points of $\boldsymbol{Q}$, and $\boldsymbol{Q}$ does not dominate any point of $\boldsymbol{P}$, it is reasonable to expect that the indicator value $I(\boldsymbol{P}, \boldsymbol{Q})$ is less than $I(\boldsymbol{Q}, \boldsymbol{P})$.

In [Li and Yao, 2019], 100 indicators are listed and some discussed in detail. The performance quality indicator properties should have to be able to capture some relevant approximation set aspects: *convergence*, *spread*, *uniformity*, and *cardinality*.

Dominance must be a central criterion in reflecting the *convergence* of solution sets, which measures the closeness of a solution set to the Pareto-optimal frontier.

The *spread* of a solution set must consider the region that the set is covering. It involves finding boundaries and the distribution of the points in the interior of the Pareto set.

The number of solutions in the set is another property, known as *cardinality*. In general, solutions sets with more candidates and generated with the same computational resources are preferred.

Finally, a good indicator must prefer a set with uniformly distributed points, *uniformity*, showing an equidistant spacing (measured in Euclidean or Manhattan distance, for example) amongst solutions. *Spread* and *uniformity* are closely related, and they collectively are known as the diversity of a set.

Some indicators present all quality aspects but brings some drawbacks [Li and Yao, 2019]. It is plausible to add more perspectives about quality indicators, such as

computational cost. Notably, some indicators are not viable (computationally speaking) to be calculated in high dimensions or for high-cardinality sets. Other indicator details also deserve to be mentioned, such as the necessity for a reference point or set, additional parameters, how to deal with scale, and normalization.

There are many indicators available, and they have been used in numerous situations in literature [Li and Yao, 2019]. Hypervolume (HV) [Zitzler and Thiele, 1999; Deng and Zhang, 2019; Yang et al., 2019; Bradford et al., 2018], inverted generational distance plus (IGD+) [Ishibuchi et al., 2015b], and ϵ-indicator [Zitzler et al., 2003] are some examples of commonly used quality indicators and they are going to be briefly described below:

- *Hypervolume (HV)*: Let $\boldsymbol{r'}$ be a reference point in the objective space that is dominated by all approximation sets. Let $\boldsymbol{P}$ be one approximation set. The HV value of $\boldsymbol{P}$ with regard to $\boldsymbol{r'}$ represents the volume of the region which is dominated by $\boldsymbol{P}$ and dominates $\boldsymbol{r'}$. It can be expressed as:

$$HV(\boldsymbol{P}, \boldsymbol{r'}) = \boldsymbol{\lambda}\left(\bigcup_{\boldsymbol{p}\in\boldsymbol{P}} [\boldsymbol{p}; \boldsymbol{r'}]\right), \tag{2.2}$$

  in which $\boldsymbol{\lambda}$ is the M-dimensional Lebesgue measure. An extensive analysis of the computational complexity is presented in [Guerreiro et al., 2021]. Chan's algorithm presents the best time complexity, considering $m \geq 4$: $O(|\boldsymbol{P}|^{\frac{M}{3}}$ polylog $|\boldsymbol{P}|)$.

- *Inverted generational distance plus (IGD+)* [Ishibuchi et al., 2015b]: Let $\boldsymbol{R^*}$ be a reference set on the Pareto front. Considering $\boldsymbol{P}$ as an approximation set not previously defined, the inverted generational distance between $\boldsymbol{R^*}$ and $\boldsymbol{P}$ is defined as:

$$IGD+(\boldsymbol{R}^*, \boldsymbol{P}) = \frac{1}{|\boldsymbol{R}^*|}\left(\sum_{\boldsymbol{r}\in\boldsymbol{R}^*} d(\boldsymbol{r}, \boldsymbol{P})\right)^{1/2}, \tag{2.3}$$

  where for minimization $d(\boldsymbol{r}, \boldsymbol{P}) = \max\{p_i - r_i, 0\}$ representing the distance from p_i to the closest solution in $\boldsymbol{R}^*$ with the corresponding value r_i. This indicator is a measure that represents how far the approximation set is from the reference. Lower values of IGD+ represent a better performance. The IGD+ metric is able to measure both diversity and convergence of $\boldsymbol{P}$ if $|\boldsymbol{R}^*|$ is large enough [Cheng et al., 2018]. The computational cost is $O(M|\boldsymbol{R}^*||\boldsymbol{P}|)$ [Audet et al., 2018].

- *ϵ-additive/multiplicative indicator*: it is a extension to the evaluation of approxi-

mation schemes in operational research and theory [Zitzler et al., 2003]. For two solution sets $\boldsymbol{P}$ and $\boldsymbol{Q}$, the additive ϵ-indicator measures the maximum of minimum distance to move one solution set, such that it dominates another solution set. Formally, the additive ϵ-indicator is calculated as:

$$I_{\epsilon^+}(\boldsymbol{P},\boldsymbol{Q}) = \max_{\boldsymbol{q}\in\boldsymbol{Q}} \min_{\boldsymbol{p}\in\boldsymbol{P}} \max_{m\in\{1,\dots,M\}} (p_m - q_m)\,, \tag{2.4}$$

in which p_m denotes the objective value of solution $\boldsymbol{p}$ in the m-th objective. For the multiplicative ϵ-indicator, the $p_m - q_m$ is replaced by $\frac{p_m}{q_m}$. A value of $I_{\epsilon^+}(\boldsymbol{P},\boldsymbol{Q}) \leq 0$ or $I_{\epsilon^\times}(\boldsymbol{P},\boldsymbol{Q}) \leq 1$ implies that $\boldsymbol{P}$ weakly dominates $\boldsymbol{Q}$. The computational cost is $O(M|\boldsymbol{P}||\boldsymbol{Q}|)$.

Since one of the two main goals in evolutionary algorithms is to find a set of non-dominated well-distributed solutions, it is relevant to mention some quality indicators that focus on this goal. They assess a uniform distribution of points on the efficient front:

- A naive uniformity estimation method is to measure the distance between a solution and its nearest neighbor solution [Kukkonen and Deb, 2006]. However as the number of objectives increase, the number of effective nearest neighbors must be adapted. It is possible to use the Euclidean distance of a point from its k-th nearest neighbor, in which $k = \lfloor\sqrt{M-1}\rfloor$, for example. The uniformity measure for a set of ND points is then computed by mean and standard deviation of k-th nearest neighbor's distances of all members of the set. A large value of mean and a small value of standard deviation are desired.

- *Spacing* (SP), proposed in Schott [1995], measures how uniform the ND solutions are according to the Euclidean distance from their nearest neighbors. Considering $\boldsymbol{S}$ as the obtained ND set, SP is computed as follows:

$$SP(\boldsymbol{S}) = \sqrt{\frac{1}{|\boldsymbol{S}|-1}\sum_{i=1}^{|\boldsymbol{S}|}\left(d_i - \bar{d}\right)^2}, \tag{2.5}$$

in which d_i is the Euclidean distance in objective space between the individual $\boldsymbol{S_i}$ and its nearest neighbor, and $\bar{d}$ is the mean of all d_i. Smaller spacing values indicate better distribution.

- *Uniform Distribution* (UD), proposed in Tan et al. [2001], is a niching-based unary quality indicator. For a given solution set, the indicator is computed as

follows:

$$UD(\boldsymbol{S}) = \frac{1}{1+\Sigma_{nc}}, \tag{2.6}$$

in which $\Sigma_{nc} = \sqrt{\frac{\sum_{i=1}^{|\boldsymbol{S}|}(nc(\boldsymbol{S_i})-\overline{nc}(\boldsymbol{S}))^2}{|\boldsymbol{S}|-1}}$ is the standard deviation of niche count of the overall solution set $\boldsymbol{S}$, $\overline{nc}(\boldsymbol{S})$ is the mean value of niche counts, which is computed for the i-th individual $\boldsymbol{S_i}$, as follows:

$$\begin{aligned} nc(\boldsymbol{S_i}) &= \sum_{j=1,j\neq i}^{|\boldsymbol{S}|} Sh(\boldsymbol{S_i},\boldsymbol{S_j}), \\ \text{where, } Sh(\boldsymbol{S_i},\boldsymbol{S_j}) &= \begin{cases} 1, & \text{if } d(\boldsymbol{S_i},\boldsymbol{S_j}) < \sigma_{share}, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{2.7}$$

The distance $d(\boldsymbol{S_i},\boldsymbol{S_j})$ is the normalized Euclidean distance between i-th and j-th individuals in the objective space, and σ_{share} is a user-specified parameter for specifying a critical threshold. As it is the inverse of standard deviation, larger values indicate a better uniformity.

- *Evenness(ξ)* [Messac, 2004] is an performance quality indicator to measure gaps, if any, in the ND solution set. For each point $\boldsymbol{S_i}$, we define two hyperspheres whose diameters are calculated in the following way. The first hypersphere puts the point $\boldsymbol{S_i}$ and its nearest neighbor in the objective space on the opposite ends of the diameter, meaning the diameter d_l^i is equal to the Euclidean distance of $\boldsymbol{S_i}$ from its nearest neighbor. The second hypersphere has a diameter d_u^i that puts $\boldsymbol{S_i}$ and its farthest neighbor on opposite ends and with no other points inside the hypersphere. Constructing $D = \{d_l^i, d_u^i : \boldsymbol{S_i} \in \boldsymbol{S}\}$, ξ is defined as follows:

$$\xi(\boldsymbol{S}) = \frac{\sigma_D}{\hat{D}}, \tag{2.8}$$

where σ_D and $\hat{D}$ are, respectively, the standard deviation and mean of D. The indicator can be viewed as a coefficient of variation, and the smaller the value, the better is the uniformity.

Hypervolume has been widely applied in the evolutionary community as a unary indicator choice, [Audet et al., 2018] and [Guerreiro et al., 2021]. It is still important to note its drawbacks: the increasing exponential cost (due to the number of objectives) and the need to designate an explicit reference point [Ishibuchi et al., 2018]. The reference point affects the ordering of pairs of incomparable sets. Some community

efforts have focused on discovering new indicators, which can overcome hypervolume's limitations while providing a good set of properties.

An ideal quality indicator must present the four facets (*convergence*, *spread*, *uniformity cardinality*). Besides, it must have a low computational cost in its calculation and should not need a normalization. Scaling invariant is another characteristic in which it only exploits the dominance relation among solutions sets, but not their absolute objective function values [Zitzler et al., 2008]. Finally, an ideal quality indicator should avoid any additional parameters or the necessity to specify a reference point or set.

2.1.3 Some test problem sets

Test problem sets have been used as a performance benchmark for EMO/EMaO algorithms. They are also called test functions, artificial problems with a precise mathematical formulation, varying features, and difficulties. A range of artificial test problems exists, such as ZDT, DTLZ, WFG, and MaF, to name a few, [Huband et al., 2011]. Given the theme relevance, a paper proposes a parameterized generator of scalable and customizable benchmark problems for MaOP, Meneghini et al. [2020], for example.

DTLZ family presents seven problems, and it is possible to configure the number of decision variables and the number of objectives. The Walking Fish Group - WFG brings nine instances, and it is also possible to configure the number of objectives and decision variables. The solution vector contains k position parameters and l distance parameters, so the number of decision variables $n = k + l$. In contrast to DTLZ, some WFG test problems are non-separable.

The MaF is the benchmark problem set from the Competition on Many-Objective Optimization at the 2017 and 2018 IEEE Congress on Evolutionary Computation. Every problem has an instance for five, ten, and fifteen objectives.

As can be seen in Table 2.1, the DTLZ, MaF, and WFG test problem families, for example, present different properties with good properties variation among them. For example, these families can offer geometrical features, separability characteristics, bias evidence, and multi-modality.

A natural extension of these test problem sets is formulated with constraints. Constrained multi-objective optimization problems (CMOPs) are also available in literature. In Deb and Jain [2014] the DTLZ family problems are changed, adding a constraint to the original problems. The idea is to provide an infeasible barrier in approaching the Pareto-optimal front, creating infeasible regions in the objective space. Therefore, the problem sets DTLZ1 and DTLZ3 are modified, turning the C1-DTLZ1

Table 2.1. DTLZ, MaF and WFG are problem set families commonly found in EMO and EMaO. Each problem set is presented with its properties.

Family test problem	*Test problem*	*Properties*
DTLZ	*DTLZ1*	Linear and multimodal
	DTLZ2	Concave
	DTLZ3	Concave and multimodal
	DTLZ4	Biased and Concave
	DTLZ5	Concave and degenerate
	DTLZ6	Biased, concave, and degenerate
	DTLZ7	Disconnected, mixed, and multimodal
MaF	*MaF01*	Inverted and linear
	MaF02	Concave
	MaF03	Convex and multimodal
	MaF04	Concave, inverted, and multimodal
	MaF05	Biased and Convex
	MaF06	Concave and degenerate
	MaF07	Disconnected, mixed, and multimodal
	MaF08	Degenerate and linear
	MaF09	Degenerate and linear
	MaF10	Biased and mixed
	MaF11	Convex, disconnected, and nonseparable
	MaF12	Biased, concave, deceptive, and nonseparable
	MaF13	Concave, degenerate, nonseparable, and unimodal
	MaF14	Linear, large scale, and partially separable
	MaF15	Convex, inverted, large scale, and partially separable
WFG	*WFG1*	Biased, convex and mixed
	WFG2	Convex, disconnected, and multimodal
	WFG3	Degenerate and linear
	WFG4	Concave and multimodal
	WFG5	Concave
	WFG6	Concave
	WFG7	Concave and biased
	WFG8	Concave and biased
	WFG9	Concave, multimodal, and biased

and C1-DTLZ3, respectively. Additionally to the difficulties arriving at the entire Pareto-optimal front, another type of constraint is proposed: introduce infeasibility to a part of the Pareto-optimal front, creating a disconnected Pareto-optimal front. DTLZ2 is altered in this way, making the C2-DTLZ2 problem.

In Fan et al. [2020], a toolkit to make difficulty-adjustable and scalable CMOPs (DAS-CMOPs) is discussed. They suggest nine difficulty-adjustable and scalable CMOPs and nine CMaOPs, named DAS-CMOP1-9 and DAS-CMaOP1-9, respectively.

Real problems also bring additional difficulties and properties. For example, some common real test problems are the crashworthiness, car side impact, the three-bar truss optimization, the mineral processing production planning [Hua et al., 2021], the wind turbine design problem [Fritsche and Pozo, 2020], and the hybrid electric

vehicle controller design problem [Cheng et al., 2017].

Some theoretical problems added with some real problems represent a good test set cohort available in some EMO/EMaO frameworks such as DEAP [Fortin et al., 2012], pymoo [Blank and Deb, 2020a], PlatEMO [Tian et al., 2017] and others.

2.1.4 Challenges in evolutionary many-objective optimization

There are some challenges in many-objective optimization. This section brings some of them, focusing on the difficulties of increasing the number of solutions and objectives, visualizing solution sets in high dimensions, the ineffectiveness of Pareto-based algorithms, and balancing convergence and diversity in evolutionary algorithms.

2.1.4.1 Number of solutions and objectives

Most of the EMO/EMaO algorithms use the concept of Pareto Dominance, comparing and identifying the best solutions. However, considering the MaOP, the proportion of non-dominated solutions increases according to the number of objectives [Li et al., 2015a]. For example, the portion of any two individuals being comparable in an m-dimensional objective space is $\eta = 1/2^{m-1}$, thus in a two- or three-dimensional space, η is equal to 0.5 or 0.25, respectively, meaning that fifty and twenty-five individuals are comparable in a randomly-produced population with 100 individuals. However, when m gets to five or seven, η is almost 0.0625 and 0.0019, with around six and less than one comparable individuals, respectively (considering randomly-produced population with 100 individuals). In this scenario, the selective pressure to find better solutions is reduced, slowing down the overall evolutionary search process.

Many-objective optimization algorithms tend to increase the computational cost (time or/and space requirement) with the increment in the number of objectives. For example, some EMO algorithms, such as PAES and PESA-II, increase exponentially with the number of objectives [Li, 2015]. The same fact also happens to some quality indicators, such as the exact hypervolume calculation and Integrated Preference Functional - IPF calculation [Bozkurt et al., 2010].

A relationship that describes the behavior between the number of solutions and objectives is found in Deb and Saxena [2005]. If there is an increase in the number of objectives in the objective space, the number of solutions representing the Pareto-optimal frontier also increases. Thus, if N points are needed for satisfactorily representing a one-dimensional Pareto-optimal front, N^M points will be necessary to express an M-dimensional Pareto-optimal front, for example.

Some different approaches have been used to deal with the dimensionality in the objective space. Objective reduction is one example, it can alleviate the scenario. In Yuan et al. [2018], for example, the objective reduction is treated as a multiobjective search problem, and three multiobjective formulations of the problem are discussed: the first two formulations are both based on preservation of the dominance structure, and the third utilizes the linear correlation between objectives.

In summary, these aspects pose general challenges in the design and use of EMaO algorithms.

2.1.4.2 Visualization

Visualization is an essential part of optimization and decision-making process. An efficient visualization technique can help understand the solution sets' location and shape. It also helps in assessing conflicts among the objectives, and it is able to support the decision-maker in choosing solutions in real-world situations.

The scatter plot can show some Pareto characteristics in two or three objective space. In Figure 2.1 is it possible to observe the location, shape, and Pareto's diversity.

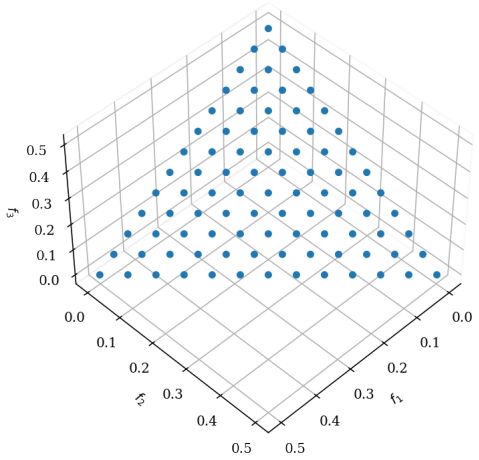

Figure 2.1. A scatter plot for DTLZ1 problem set with $M = 3$

However, as the number of objectives increases beyond three, it becomes difficult to visualize the objective space. Some authors seek to circumvent this situation in several ways. The parallel coordinates plot [Inselberg and Dimsdale, 1990] is used as the y axis represents the dimensions of a point in space, and the x axis presents values of each coordinate. One solution is represented by a line connecting the respective values of each coordinate. As an example, in Figure 2.2 is possible to contrast two different solutions in a objective space with $M = 6$.

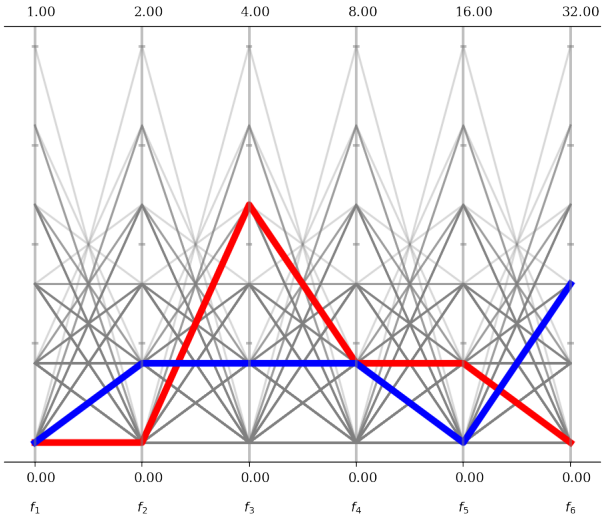

Figure 2.2. A parallel coordinates plot example, contrasting two solutions in an objective space with $M = 6$. It is possible to compare the red solution with the blue one.

Another visual technique is used to represent the solutions through the radar chart. It is similar to parallel coordinates, in which M objectives means M radial axes. It is also called a spider chart, polar chart, or star plot. Typically, the inside of the plot represents small values, while the outside represents high values. Figure 2.3 shows an example of radar plots with three solutions from each approximation set, the red and green ones.

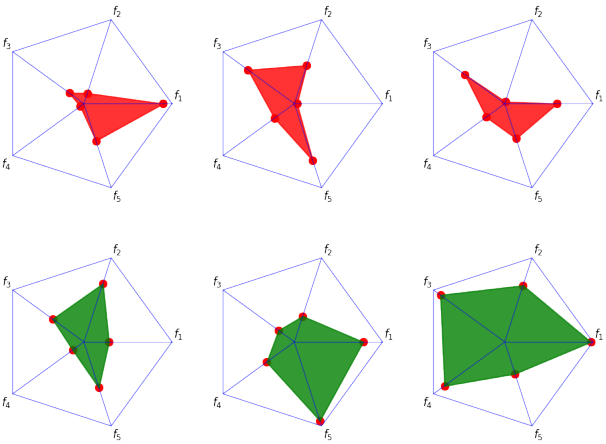

Figure 2.3. An example of a radar plot showing six different solutions from two approximation sets with $M = 5$.

Heatmap and star coordinate plots are other standard techniques to deal with many-objective problems. All these methods try to outline the whole solution set, showing the set's range, location, and shape.

Extracting information like convergence or shape in many-objective problems is

not easy, even using these existing techniques. Some efforts have been employed in this issue; a recent paper, for example, proposes a method (PaletteViz) to visualize several different structures of Pareto-optimal data sets [Talukder and Deb, 2020]. However, visualization in a many-objective scenario is still a significant challenge.

2.1.4.3 Ineffectiveness of Pareto-based algorithms

The solutions found by EMO/EMaO algorithms can be seen from two perspectives: dominated by other solutions or non-dominated and possibly belonging to the Pareto-optimal set. As the number of objectives increases, almost all solutions in the population become non-dominant, which weakens the Pareto-based algorithms. It directly impacts the algorithm selection pressure.

When the Pareto dominance criterion fails to distinguish between individuals, one possible alternative is to use the density-based selection criterion. It will play a relevant role in determining the survival of individuals. Some papers, as Wagner et al. [2006]; Ishibuchi et al. [2008] show that the diversity improvement has an impact on the algorithm's proximity due to its preference for dominance resistant solutions, DRS, meaning as solutions with an extremely poor value in at least one of the objectives, but with near-optimal values in some others.

If dominance resistant solutions are distributed in a larger area in the search space than the actual Pareto-optimal set, Pareto-dominance-based algorithms, during the solution search process, may fail to obtain a good approximation of this set. In this way, all non-dominated solutions are treated as survivors, and the DRS spreads across the search space and lasts for several generations. Some studies have shown that a random search algorithm may even achieve better results than Pareto-based algorithms in problems with around ten objectives [Purshouse and Fleming, 2007; Knowles and Corne, 2006].

This ineffectiveness is still present in some quality indicators. Since the portion of the space that a solution dominates decreases with the number of objectives, most solutions in different approximation sets are likely to be incomparable under the Pareto dominance criteria. Indicators based on comparing solutions' Pareto dominance relation, such as coverage and G-metric [Li, 2015] also present this characteristic. For example, coverage captures the proportion of points in an approximation set $\boldsymbol{A}$ dominated by the approximation set $\boldsymbol{B}$. As we increase the number of objectives, the number of incomparable solutions also increases, turning challenging to use this kind of indicator.

2.1.4.4 Balancing convergence and diversity

Maintaining the balance between convergence and uniformity solutions in EMO/EMaO algorithms is a complex task [Tian et al., 2021a]. Many algorithms focus on these aspects, aiming to improve the diversity and uniformity of the solution set concomitantly with search efficiency and convergence.

In the diversity preservation perspective, the solution proposals can be roughly categorized into three groups: I) approaches that try to preserve diverse subpopulations converging toward different optimal solutions for local exploitations, such as in NSGA-II [Deb et al., 2002]; II) Greedy strategy, such as SPEA2 [Zitzler et al., 2001] which, for example, calculates the Euclidean distance between all solutions in a pairwise manner, and then iteratively excludes the solution that brings the minimum value to the other solutions. III) The last category is based on a concept that decomposes the objective space into some sub-regions using weight vectors where a representative solution is maintained inside each sub-region, such as NSGA-III [Jain and Deb, 2014] and MOEA/D [Zhang and Li, 2007].

It is well known that the usage of reference vectors (RVs) in EMO/EMaO helps to guide the evolutionary process towards convergence and uniformity. According to Messac [2004], an uniformly distributed solution is defined as: "A set of points is evenly distributed over a region if no part of that region is over or underrepresented in that set of points, compared to other parts". Keeping that in mind, as a way to guide the search process, reference vectors can be used as uniform neighborhood axes to convert the original problem into sub-problems with corresponding sub-spaces. In the literature, there are some approaches, such as MOEA/D [Zhang and Li, 2007] and NSGA-III [Jain and Deb, 2014], that explore that idea.

The two-archive evolutionary algorithm for constrained multi-objective optimization, C-TAEA, proposed in Li et al. [2019b], tries to balance convergence and uniformity in a two-archive procedure that simultaneously maintains two populations: the convergence archive (CA) and the diversity archive (DA). The former supports the convergence and feasibility in the evolutionary process while the latter explores areas that the convergence archive has not exploited yet. In this algorithm, the reference vectors are also used in the update mechanism of the CA and DA.

There are generally difficulties in solving MOP/MaOP even when the reference vector approach is employed. The quality of the final solution set depends on choice of reference vectors, and this choice is somehow inherently arduous. Although there are some EMO/EMaO that are primarily focused on these problems, most of them have been demonstrated to have poor versatility [Tian et al., 2021a].

A successfully approach is the multi-stage or multi-phase evolutionary algorithms ([Ge et al., 2019; Qi et al., 2014; Tian et al., 2021b; Seada et al., 2019; Tian et al., 2021a] and [Liang et al., 2021]). In general, this approach divides the optimization process into a certain number of stages and applies different selection strategies in these stages. In this way, the diversity performance of the multi-stage evolutionary algorithm is improved.

The MOEA/D-AWA [Qi et al., 2014] uses a two-stage strategy to deal with the generation of the weight vectors. In the first stage, a set of pre-determined weight vectors are used until the population is considered converged. Then, in the second stage, some weight vectors are adjusted according to the current Pareto optimal solutions based on geometric analysis.

In Seada et al. [2019], B-NSGA-III is presented as a multi-phased many-objective evolutionary optimization algorithm capable of automatically balancing convergence and diversity of population members. The algorithm uses three phases dynamically. In phase 1, it looks for extreme points. In phase 2, it tries to cover gaps found in the non-dominated front. And finally, in phase 3, it concentrates the efforts toward helping poorly converged non-dominated solutions. The algorithm can interchange from one phase to another if some triggers/conditions are observed. Results show superior performance when compared to several state-of-the-art algorithms in a set of benchmark problems.

In Tian et al. [2021b], a multi-stage constrained multi-objective evolutionary algorithm, CMOEA-MS, is proposed. Exclusively focusing on constrained problems, the algorithm balances objective optimization and constraint satisfaction giving different priorities in each phase. The first phase indicates that most solutions are infeasible, and these solutions can support the population to leave the infeasible regions. The second phase points that most solutions are feasible so that more feasible solutions can be found to help the population spread along the feasible boundaries. The general strategy uses different priorities of objectives and constraints in the two stages. The objective optimization and constraint satisfaction are well balanced, tackling different feasible regions. The work presents some experiments based on constrained problem sets.

A many-objective evolutionary algorithm combining two interacting processes, Cascade Clustering (CC) and Reference Point Incremental Learning, dubbed as CLIA, is proposed in Ge et al. [2019]. The cascade clustering (CC) uses the non-dominated and dominated solutions to create clusters, sort, and select solutions for a better convergence and diversity. CC uses reference lines to create and sort the solutions. In the second process, a support vector machine (SVM) model is applied in an incremen-

tal learning approach, classifying the reference points as active/inactive and creating denser reference lines (generated on the unit simplex) and is used in the optimization process. The incremental SVM model adjusts the RVs during the evolution process. Although CLIA hybridizes an evolutionary approach with SVM, due to an adaptive adjustment of RVs to provide a better distribution of ND solutions.

RVRL-EA [Ma et al., 2021] is a recent framework that uses reinforcement learning to adapt reference vectors. However, it only applies to decomposition-based algorithms. The authors have used Q-learning algorithm, and they suggest that other reinforcement learning models should be assessed jointly with a better method to generate uniform reference vectors. Experiments are presented, showing that the algorithm is competitive compared to CLIA [Ge et al., 2019].

MSEA [Tian et al., 2021a], a multi-stage EMaO algorithm, is designed for improving the diversity performance of EMaO. In each generation, firstly, an assessment population is based on convergence, which determines stage 1 with a specific selection algorithm. Then, in stage 2, the diversity is assessed again; a particular selection algorithm generates offspring to replace one solution in the population. Finally, in stage 3, there is a mating selection for a parent with good convergence and diversity. The authors present results for multi-objective problems, and the method presents a cubic computational complexity related to the number of solutions. In summary, a specific selection strategy is adopted in each stage's mating and environmental selection; however, the reported algorithm is not tested, and it is not suitable for many-objective problem sets.

All the algorithms mentioned above address the question of balancing convergence and diversity in a multi-stage or multi-phase approach

2.2 Mathematical programming

Mathematical programming can be seen as a process of assessing a set of available alternatives and select the best elements using some criteria. The mathematical programming problem consists of maximizing or minimizing a function, choosing input values from the alternatives, and computing the function's value. The approach has been used to solve problems as production planning, scheduling, localization, routing, and many others [Sierksma and Zwols, 2018].

2.2.1 Mixed-integer linear programming

Linear programming is one of the most used optimization methods in mathematical programming. It is a technique for optimizing a linear objective function, subject to linear equality and linear inequality constraints [Sierksma and Zwols, 2018]. A common formulation of the linear programming model is presented in Equation (2.9):

$$\text{minimize} \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to:} \begin{cases} \sum_{j=1}^{n} a_{ij} x_j \leq b_i & \forall i \in \{1, 2, .., m\}, \\ x_j \geq 0, & \forall j \in \{1, 2, .., n\}. \end{cases} \tag{2.9}$$

The number of decision variable is defined by n. The x_j represents the *j-th* decision variable. The number of constraints is indicated by m. A coefficient value is a_{ij} related to *i-th* constraint and associated to *j-th* decision variable. Finally, b_i and c_j are model parameters, meaning an independent term associated to the *i-th* constraint, and the cost associated with the *j-th* decision variable, respectively.

Another standard method in the field is mixed-integer linear programming (MIP). It differs from linear programming in which some of the variables are constrained to be integers, while other variables are allowed to be non-integers.

A common mixed-integer linear model is:

$$\text{minimize} \sum_{j=1}^{n_1} c_j^1 x_j + \sum_{j=1}^{n_2} c_j^2 y_j$$

$$\text{subject to:} \begin{cases} \sum_{j=1}^{n_1} a_{ij}^1 x_j + \sum_{j=1}^{n_2} a_{ij}^2 y_j \leq b_i & \forall i \in \{1, 2, .., m\}, \\ x_j \geq 0, & \forall j \in \{1, 2, .., n_1\}, \\ y_j \in \mathbb{Z}_+ & \forall j \in \{1, 2, .., n_2\}. \end{cases} \tag{2.10}$$

In Equation (2.10) there are two types of variables $\boldsymbol{x}$ and $\boldsymbol{y}$. The $\boldsymbol{x}$ are continuous and non-negative variables, and $\boldsymbol{y}$ are non-negative, and integer variables, with n_1 and

n_2 number of variables, respectively. $\boldsymbol{c^1}$ and $\boldsymbol{c^2}$ are the objective function coefficients related to $\boldsymbol{x}$ and $\boldsymbol{y}$. At the same way, the $\boldsymbol{a_j^1} \in \mathbb{R}^{n_1}$ and $\boldsymbol{a_j^2} \in \mathbb{R}^{n_2}$ are the left-hand side constraint coefficients associated with the *i-th* constraint. Lastly, for the constraint set, $\boldsymbol{b_i}$ is a constant vector representing the right-hand side constants and associated with the *i-th* constraint, [Klotz and Newman, 2013].

There are still other common models used in mathematical programming, such as binary linear programming, non-linear programming, stochastic optimization, and others [Lepenioti et al., 2020].

2.2.2 The assignment problem

Historically, there are various classical problems in mathematical programming. The assignment problem is discussed in many mathematical programming and management sciences works. Generally, it addresses the problem of dealing with the question of how to assign n tasks to n agents. The objective function is related to minimizing the total cost of the assignments. However, some examples still involve other situations, such as assigning jobs to machines, tasks to workers, products to machines, students to teachers, clients to sellers, and many others.

In Figure 2.4 we can see a bipartite graph with two sets $\boldsymbol{P}$ and $\boldsymbol{Q}$. Here, our objective is to assign exactly one member of $\boldsymbol{P}$ to each one member of $\boldsymbol{Q}$ with the minimum cost, represented by the edges. Figure 2.4 presents a valid assignment from $\boldsymbol{P}$ to $\boldsymbol{Q}$.

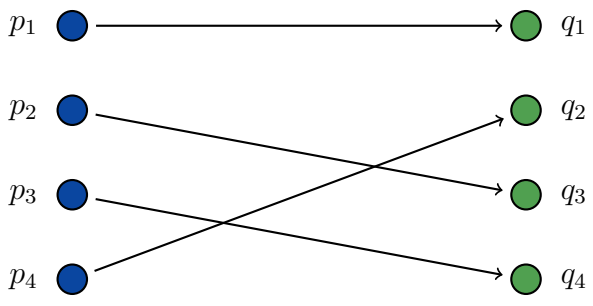

Figure 2.4. One possible example of assignment between $\boldsymbol{P}$ and $\boldsymbol{Q}$. The edges indicating the assignment can be computed using the minimum distance from $\boldsymbol{p}$ to $\boldsymbol{q}$, for example.

The mathematical model for the classic assignment problem may be given as [Pentico, 2007]:

$$\text{minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$\text{subject to:} \begin{cases} \sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \in \{1,2,..,n\}, \\ \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in \{1,2,..,n\}, \\ x_{ij} = 0 \text{ or } 1, \end{cases} \tag{2.11}$$

where $x_{ij} = 1$ indicates if the client/machine i is assigned to the agent/task j, and 0 otherwise. The c_{ij} represents the cost involved in the seller j assigned to the client i. The first constraint ensures that every agent is assigned to only one client and the second constraint ensures that every client is assigned to a agent. It is worth mention that x_{ij} is a binary variable.

The classical linear assignment problem has many variations, such as the lexicographic bottleneck assignment problem, algebraic assignment problem, and sum-k assignment problem.

There are still other more complex assignment problems: the quadratic, cubic, and quartic assignment problem, the quadratic semi-assignment, and the multi-index assignment problem [Burkard et al., 2012].

2.2.3 The packing problem and the Riesz s-energy concept

The packing problems possess a standard structure, essentially, there are two groups of data whose elements define geometric bodies in one or more dimensions; the stock of the so-called large objects and the list of the so-called small items. The packing processes realize patterns being geometric combinations of small items assigned to large objects. The residual pieces, which mean figures are occurring in patterns not belonging to small items, are usually treated as a slight loss. The objective function of most solution models in packing problems is to minimize the wasted material [Faina, 2020].

A general combinatorial optimization problem is formalized as a pair $(\mathcal{C}, \mathcal{F})$, where $\mathcal{C}$ is the finite or possibly countably infinite set of configurations and $\mathcal{F} : \mathcal{C} \to \boldsymbol{R}$ as a cost functional. The functional $\mathcal{F}$ is defined in such a way that the lower the value of $\mathcal{F}$, the better corresponding configuration. The problem is then reduced to find

a configuration for which $\mathcal{F}$ takes its minimum value, satisfying, $\mathcal{F}(\omega_0) = \min_{\omega \in \mathcal{C}} \mathcal{F}(\omega)$. The packing problems are known to be *NP*-hard combinatorial optimization problem [Faina, 2020].

In practical situations, packing problems arise in kinds of goods packaged in cartons for handling, in manufacturing and distribution industries, for example. But, there are other situations as well.

The Riesz s-energy is related to a solution to the best-packing problem. It arises from the problem of distributing N points on the unit sphere S^m in $\mathbb{R}^{m+1}$, influencing potential theory and the distribution of charges. A relevant application of the Riesz s-energy is the discretization of manifolds (e.g., Pareto fronts).

The proposed discrete Riesz s-energy measure the evenness of a set of points in m-dimensional manifolds, [Falcón-Cardona et al., 2020]. Mathematically, given an solution set $\boldsymbol{P} = \{\boldsymbol{p_1}, .., \boldsymbol{p_N}\}$, where $\boldsymbol{p_i} \in \mathbb{R}^m$, the Riesz s-energy is given as follows:

$$E_s(\boldsymbol{P}) = \sum_{\boldsymbol{p} \in \boldsymbol{P}} \sum_{\substack{\boldsymbol{q} \in \boldsymbol{P} \\ \boldsymbol{p} \neq \boldsymbol{q}}} k_s(\boldsymbol{p}, \boldsymbol{q})$$

$$k_s(\boldsymbol{p}, \boldsymbol{q}) = \begin{cases} \| \boldsymbol{p} - \boldsymbol{q} \|^{-s}, & s > 0, \\ -\log \| \boldsymbol{p} - \boldsymbol{q} \|, & s = 0. \end{cases} \tag{2.12}$$

The function k_s is the Riesz s-kernel, and s is a parameter that controls the emphasis on the uniform distribution. As s increases, a more uniform distribution is obtained. In fact, the minimization of E_s is related to the solution of the best-packing problem.

Riesz s-Energy (E_s) has been employed to successfully distribute points uniformly on a sphere or other multi-dimensional surfaces. Recently, the approach has been used to improve the diversity of EMO and EMaO [Falcón-Cardona et al., 2020], as a performance metric for multi-objective optimization and as a reduction method for choosing a set of reference points from a large set [Blank et al., 2021]. Still, in [Blank et al., 2021] an iterative process based on Riesz s-Energy can effectively find an arbitrary number of well-spaced points in higher-dimensional spaces.

2.3 Machine learning tasks

Machine learning is about designing algorithms that automatically extract valuable information from data. Three concepts are at the core of machine learning: data, a model, and the learning process [Deisenroth et al., 2020]. Data is at the heart of machine learning, and its goal is to get general-purpose procedures to obtain relevant patterns from data without domain-specific expertise. Models are typically related to simulating the process that makes the data. Finally, the learning process is the act of automatically finding patterns and structures in data by optimizing the model's parameters.

During the learning process, it is needed to measure the algorithm performance. We want to know how the learning is going, quantify it, and measure it. For example, some measures are used, such as accuracy or inversely error rate. To best measure performance, we are typically interested in assessing data that has not previously been seen by our learning algorithm. At this point emerges the concept of having training and testing datasets, in which the test portion is used as the unforeseen data.

One possible classification of machine learning algorithms is related to the kind of experience they are allowed to have during the learning process. It can be supervised (there is labeled data) or unsupervised (there is no labeled data) [Mitchell, 1997].

In another way, machine learning algorithms can be broadly categorized into a few groups based on the tasks they are designed to solve [Zaki and Meira, 2020]:

- The frequent pattern task extracts informative and valuable patterns in massive and complex datasets. Patterns comprise co-occurring attribute values, such as sequences, which consider explicit precedence relationships between entities. The goal is to discover hidden relationships in data to understand the interactions among the data points and attributes;

- Clustering is one task that seeks to separate data points into natural groups called clusters, such points within a group are very similar, while points between different groups are as dissimilar as possible. There are different clustering paradigms: representative-based, hierarchical, density-based, graph-based, and spectral Clustering. Clustering is an unsupervised learning approach since it does not require a separate training dataset to learn the model parameters;

- The classification is a task to predict the class for a given unlabeled point/case. Formally, a classifier is a function M_c that predicts the class label $\hat{\boldsymbol{y}}$ for a given input case $\boldsymbol{x}$, that is, $\hat{\boldsymbol{y}} = M_c(\boldsymbol{x})$, where $\hat{\boldsymbol{y}} \in \{c_1, c_2, ..., c_k\}$ and each c_i is a class

label. Classification is a supervised learning approach since learning the model requires a set of points with correct class labels, training set. After learning model M_c, we can automatically predict the class for any new point/case;

- Regression is a task to predict the value of a (real-valued) dependent variable $\boldsymbol{Y}$ given a set of d independent variables X_1, X_2 ,.., X_d. The goal is to learn a function M_r such that $\hat{\boldsymbol{y}} = M_r(\boldsymbol{x})$, where $\hat{\boldsymbol{y}}$ is the predicted response value given the input point $\boldsymbol{x}$. Differently to classification, in regression the response variable is real-valued, but in the same way as classification, regression is also a supervised learning approach.

2.3.1 Clustering

Clustering is the task of partitioning data points into similar groups called clusters in a way that different groups are as dissimilar as possible. It is an unsupervised learning approach, and it does not require a separate training dataset to learn model parameters.

There are different types of clustering paradigms. Representative-based clustering is a type that finds mutually exclusive clusters of spherical shape. It uses distance-based methods to calculate points similarity, and as a cluster center, it uses mean or medoid, for example, to represent the cluster center. It is commonly used for small to medium-sized data sets.

Hierarchical clustering is a hierarchical decomposition method that starts from each point in its cluster and successively merges pairs of clusters until the desired number of clusters has been found.

Density-based types can find arbitrarily shaped clusters. Clusters are dense regions of objects in space that are separated by low-density areas.

Grid-based methods use a multi-resolution grid data structure. As a result, these methods present a fast processing time, typically independent of the number of data objects.

Finally, the graph clustering type uses an approach over graph data; given a graph, the goal is to cluster the nodes by using the edges and their weights, representing the similarity between the nodes. Graph clustering is related to divisive hierarchical clustering, as many methods partition the set of nodes to obtain the final clusters using the pairwise similarity matrix between nodes.

Some clustering algorithms are relevant in our work's context.

K-Means is a representative-based algorithm that minimizes the squared distance of points from their respective cluster means (centroids) [Pham et al., 2005]. It performs

clustering interactively, assigning each point to only one cluster and minimizing the sum of squared errors. One of the parameters of K-Means is the number of clusters. Another K-Means feature is its stochastic component in the initialization step (it does not have numerical stability). However, considering the performance, the time complexity is $O(C\ L\ M\ i)$, in which C is the cluster number, L is the number of data points, M the space dimension, and i is the number of iterations.

Mean shift is another clustering algorithm [Comaniciu and Meer, 2002; Yizong Cheng, 1995]. The method assumes that exists some probability density function from which the data is obtained, and it uses the maxima of the density function as clusters of centroids. It does not require prior knowledge of the number of clusters. The number of clusters is obtained automatically by finding the densest regions' centers in the space, the modes. It is an iterative technique, and it estimates the modes of the multivariate distribution underlying the feature space. However, as it approximates the centroids via kernel density estimation techniques, the key parameter is the kernel bandwidth. Concerning the numerical stability, we have a random initialization, as well. But, the performance may vary: if it uses a flat kernel and a ball tree to look for members of each kernel, the complexity is $O(L\ s\ log(s))$ in lower dimensions, and $O(L\ s^2)$ in higher dimensions, with s the number of samples.

Affinity propagation clustering is an algorithm that treats all points as potential cluster centers. The process starts with the similarity matrix ($\boldsymbol{S}$), which is constructed in a pairwise manner using a similarity function, such as the negative Euclidean distance (values near zero indicate similarity otherwise, the degree of dissimilarity between the points) [Wang et al., 2019]. Then, given $\boldsymbol{S}$, the method tries to find 'exemplars' that maximize the net similarity. The algorithm is a graphing approach and can be viewed as a message-passing process through edges with messages exchanged among data points.

Part II

Dominance Move

Chapter 3

MIP-DoM - Dominance Move

3.1 Introduction

In Li and Yao [2017], a new quality measure, called dominance move (DoM) is proposed. Dominance move (DoM) is a binary quality indicator that can be used in multi-objective and many-objective optimization to compare two solution sets. The DoM indicator can differentiate the sets for certain important features, such as *convergence*, *spread*, *uniformity*, and *cardinality*.

DoM measures the minimum 'effort' that one solution set has to make to dominate another set, precisely the sum of the movement needed to make a set dominant. It has the same interpretation of ϵ-indicators, and it can capture some quality aspects of solution sets, such as Pareto convergence and spread. The authors propose an exact algorithm to calculate DoM for the bi-objective case that can be computed at a low computational cost. However, it can not be used or extended to three or more objectives due to the combinatorial calculation nature.

Another attempt to solve the DoM calculation is presented in d. V. Lopes et al. [2020]. DoM is considered an assignment problem. Some experiments in three objective scenarios are presented. However, the assignment-based approach still has some drawbacks preventing its use in solution sets with more than 20 solution candidates. The assignments necessary to calculate the formulation turn it prohibitive when the number of solutions in the non-dominated set increases.

This chapter focuses on the dominance move by applying a mixed-integer linear programming model (MIP) to deal with its calculation. More specifically, it presents the following items: (i) the proposed MIP model for DoM calculation is tested for some objective space dimension and cardinality of the supplied solution sets, (ii) extensive evaluation of the proposed MIP-DoM formulation and approach in a number of common

problem sets having 3, 5, 10, 15, 20, 25 and 30 objective functions, (iii) comparison with optimal DoM solutions in cases where it is possible to achieve the optimal solution, and (iv) raise of questions and details about the model behavior for some test sets, including future research paths to tackle other problems.

This chapter is organized as follows. Section 3.2 introduces DoM definition and its use as an indicator. The mixed-integer linear programming model to approach the DoM calculation is presented in Section 3.3. The MIP-DoM model is introduced with some comments and considerations. Experiments are discussed in Section 3.4. It starts with a bi-objective case presenting the results' correctness and model validity. Following, some common multi-objective problem sets and algorithms are used to compare MIP-DoM with the ϵ-indicator, $IGD+$, and Hypervolume indicators. The Section 3.4, some many-objective experiments are discussed using problem sets with 5, 10, and 15 objectives with till 240 members in the solution sets. In Section 3.5, further extensions are discussed to handle particular idiosyncrasies with some solution sets and improve the quality indicator and its use for other situations. Finally, in Section 3.6, some final considerations are presented.

3.2 DoM as quality indicator

Dominance move (DoM) is an intuitive indicator that is commensurate with all four desirable properties (i.e., *convergence*, *spread*, *uniformity*, and *cardinality*) [Li and Yao, 2017]. The first idea presenting DoM came from the performance comparison indicator (PCI) [Li et al., 2015b]. Examining the PCI proposal is quite similar to DoM in its essential purpose. PCI is a binary quality indicator, and it builds up a reference set using two solution sets, $\boldsymbol{P}$ and $\boldsymbol{Q}$. This reference set is then split up into clusters, using its clustering algorithm, and the indicator estimates the minimum moves of solutions in the approximation sets to dominate these clusters.

Dominance move is a measure for comparing two sets of multi-dimensional points, being classified as a binary indicator. It considers the minimum overall movement of members in one set needed to dominate each and every member of the other set.

The DoM definition is stated as [Li and Yao, 2017]:

Definition 4. *Dominance Move (DoM): Consider that $\boldsymbol{P}$ and $\boldsymbol{Q}$ are two sets of points, with points $\boldsymbol{p}_i$, $i \in \{1, \ldots, |\boldsymbol{P}|\}$ and points $\boldsymbol{q}_j$, $j \in \{1, \ldots, |\boldsymbol{Q}|\}$. The dominance move of $\boldsymbol{P}$ to $\boldsymbol{Q}$, $DoM(\boldsymbol{P}, \boldsymbol{Q})$, is the minimum total distance of moving points of $\boldsymbol{P}$, such that any point in $\boldsymbol{Q}$ is weakly dominated by at least one point in $\boldsymbol{P}$. In fact, the problem aims*

to obtain the moved set $\boldsymbol{P'} = \{\boldsymbol{p}'_1, \boldsymbol{p}'_2, \ldots, \boldsymbol{p}'_{|\boldsymbol{P}|}\}$ *from* $\boldsymbol{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{|\boldsymbol{P}|}\}$ *dominates such that 1)* $\boldsymbol{P'}$ *dominates* $\boldsymbol{Q}$ *and 2) the total move from* $\boldsymbol{P}$ *to* $\boldsymbol{P'}$ *is minimum.*

It is necessary to highlight that our definition is similar in concept to this one. Our definition ensures that every element of $\boldsymbol{Q}$ gets dominated by at least one element of $\boldsymbol{P}$. Since DoM corresponds to the minimum move distance, one of the moved elements of $\boldsymbol{P'}$ can be vanishingly better ($\epsilon \to 0$) in one or more objectives compared to a dominating element of $\boldsymbol{Q}$, thereby providing a vanishingly close DoM value compared to the original DoM.

Note that a solution can not dominate itself, that is, if $\boldsymbol{q} = \boldsymbol{p}$, $\boldsymbol{p}$ does not dominate $\boldsymbol{q}$, and vice versa. But if $p_j = q_j - \epsilon_j$ for a specific j and $\epsilon_j \to 0$, and for all other objectives ($i = 1, 2, \ldots, M$ and $i \neq j$) $p_i = q_i$, then $\boldsymbol{p} \prec \boldsymbol{q}$.

Our DoM definition is stated as:

Definition 5. *Dominance Move (DoM): Consider that* $\boldsymbol{P}$ *and* $\boldsymbol{Q}$ *are two sets of points, with points* $\boldsymbol{p}_i$, $i \in \{1, \ldots, |\boldsymbol{P}|\}$ *and points* $\boldsymbol{q}_j$, $j \in \{1, \ldots, |\boldsymbol{Q}|\}$. *The dominance move of* $\boldsymbol{P}$ *to* $\boldsymbol{Q}$, $DoM(\boldsymbol{P}, \boldsymbol{Q})$, *is the minimum total distance of moving points of* $\boldsymbol{P}$, *such that the moved set* $\boldsymbol{P'} = \{\boldsymbol{p}'_1, \boldsymbol{p}'_2, \ldots, \boldsymbol{p}'_{|\boldsymbol{P}|}\}$ *(with some or all* $\boldsymbol{p}'_i$ *are allowed to be infeasible) from* $\boldsymbol{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{|\boldsymbol{P}|}\}$ *dominates* $\boldsymbol{Q}$ *and that the total move from* $\boldsymbol{P}$ *to* $\boldsymbol{P'}$ *must be minimum.*

The original DoM definition in [Li and Yao, 2017] uses the *weakly* dominance term. There is an issue related to it. The classical Multiple Criteria Decision-Making MCDM literature [Kaisa, 1999; Branke et al., 2008] defines the term *weakly* efficient (or *weakly* Pareto-optimal) solution as follows: A feasible solution $\boldsymbol{p}$ is weakly efficient, if there does not exist any feasible solution $\boldsymbol{q}$ in the search space such that $q_m < p_m$ for all $1 \leq m \leq M$. It means that for a solution $\boldsymbol{q}$ to be weakly efficient, there can be another point $\boldsymbol{p}$ having a few objectives identical to $\boldsymbol{q}$, but there cannot be any point which is strictly better than $\boldsymbol{q}$ in all objectives. There is no definition of a *weakly* dominance term in the MCDM literature, as it is sometimes used in the EMO literature in the following way: Consider two solutions, $\boldsymbol{p}$, $\boldsymbol{q}$. We can state that $\boldsymbol{p}$ *weakly* dominate $\boldsymbol{q}$ if $p_m \leq q_m$ for all for $1 \leq m \leq M$ and $p_m < q_m$ for at least one m. It is denoted as $\boldsymbol{p} \preceq \boldsymbol{q}$. The set $\boldsymbol{P}$ weakly dominates $\boldsymbol{Q}$, i.e. it is denoted as $\boldsymbol{P} \preceq \boldsymbol{Q}$, if every solution $\boldsymbol{q} \in \boldsymbol{Q}$ is weakly dominated by at least one solution $\boldsymbol{p} \in \boldsymbol{P}$. This definition is founded in: [Li et al., 2015b; Zitzler et al., 2003; Li, 2015; Ishibuchi et al., 2015b], for example.

Considering the similarity terms issue, we decide to use the dominance definition discussed in Chapter 2. Our dominance move definition is adequate with the standard Pareto dominance concept and achieves the desired concept proposed:

1. Every element of $\boldsymbol{Q}$ gets dominated by at least one element of $\boldsymbol{P}$.

2. Since DoM corresponds to the minimum move distance, one of the moved elements of $\boldsymbol{P'}$ can be vanishingly better ($\epsilon \to 0$) in one or more objectives compared to a dominating element of $\boldsymbol{Q}$, thereby providing a vanishingly close DoM value compared to the original DoM.

   The formal expression of DoM can be stated as:

$$DoM(\boldsymbol{P}, \boldsymbol{Q}) = \min_{\boldsymbol{P'} \prec \boldsymbol{Q}} \sum_{i=1}^{|\boldsymbol{P}|} d(\boldsymbol{p}_i, \boldsymbol{p}'_i), \tag{3.1}$$

in which $d(\boldsymbol{p}_i, \boldsymbol{p}'_i)$ is not negative and can be the Manhattan distance between $\boldsymbol{p}_i$ to $\boldsymbol{p}'_i$ [Li and Yao, 2017].

DoM reflects how far one solution set $\boldsymbol{P}$ needs to move to dominate another solution set $\boldsymbol{Q}$. Thus, $DoM(\boldsymbol{P}, \boldsymbol{Q})$ is always larger than or equal to zero. A small value suggests that $\boldsymbol{P}$ and $\boldsymbol{Q}$ are close (e.g., one point of $\boldsymbol{P}$ that only needs a little movement to dominate $\boldsymbol{Q}$), and a large value indicates that $\boldsymbol{P}$ performs worst than $\boldsymbol{Q}$ (e.g., some points of $\boldsymbol{P}$ must be moved further away in an attempt to dominate $\boldsymbol{Q}$).

The ϵ-indicators also measure the minimum value added to one solution set to make it dominated by another set. However ϵ-indicators does not try to use all information available. One simple case, proposed in [Li and Yao, 2017], can be easily examplify the situation: consider two 10-objective solutions, such as $\boldsymbol{p_1} = \{0, 0, 0, .., 1\}$ and $\boldsymbol{q_1} = \{1, 1, 1, .., 0\}$. In this case, ϵ-additive($\boldsymbol{p_1}$, $\boldsymbol{q_1}$) = ϵ-additive($\boldsymbol{q_1}$, $\boldsymbol{p_1}$) = 1.

The dominance move indicator is based on the properties of dominance relation among solutions trying to dominate each other, considering all available information (all solutions and objectives). These solutions' efforts scale in a bottom-up manner from the solutions to the set relations. In this sense, the authors claimed that DoM possesses some properties that are enumerated in the following proposition [Li and Yao, 2019]:

Proposition 1. *Consider the solution sets* $\boldsymbol{P}$, $\boldsymbol{Q}$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C} \subset \Theta$ *and the points* $\boldsymbol{p}$, $\boldsymbol{q} \in \Theta$*:*

a) $\boldsymbol{P} = \boldsymbol{Q} \iff DoM(\boldsymbol{P}, \boldsymbol{Q}) = DoM(\boldsymbol{Q}, \boldsymbol{P}) = 0$*;*

b) $\boldsymbol{P} \lhd \boldsymbol{Q}$ *(*$\boldsymbol{P}$ *is better than* $\boldsymbol{Q}$*)* $\iff DoM(\boldsymbol{P}, \boldsymbol{Q}) = 0 \wedge DoM(\boldsymbol{Q}, \boldsymbol{P}) > 0$*;*

c) $DoM(\boldsymbol{P}, \boldsymbol{Q}) \geq DoM(P \cup p, Q)$ *and* $DoM(\boldsymbol{P}, \boldsymbol{Q}) \leq DoM(P, Q \cup q)$*;*

d) Let $\boldsymbol{p} \in \boldsymbol{P}$ *and* $\boldsymbol{q} \in \boldsymbol{Q}$*. If* $\exists \boldsymbol{p'} \in \boldsymbol{P}, \boldsymbol{p'} \prec \boldsymbol{p}$ *then* $DoM(\boldsymbol{P}, \boldsymbol{Q}) = DoM(\boldsymbol{P}/\boldsymbol{p}, \boldsymbol{Q})$*. Also, if* $\exists \boldsymbol{q'} \in \boldsymbol{Q}, \boldsymbol{q'} \prec \boldsymbol{q}$ *or* $\exists \boldsymbol{p'} \in \boldsymbol{P}, \boldsymbol{p'} \prec \boldsymbol{q}$ *then* $DoM(\boldsymbol{P}, \boldsymbol{Q}) = DoM(\boldsymbol{P}, \boldsymbol{Q}/\boldsymbol{q})$

e) If $\boldsymbol{A} \preceq \boldsymbol{B}$*, then* $DoM(\boldsymbol{A}, \boldsymbol{C}) \leq DoM(\boldsymbol{B}, \boldsymbol{C})$ *and* $DoM(\boldsymbol{C}, \boldsymbol{B}) \leq DoM(\boldsymbol{C}, \boldsymbol{A})$*;*

f) $DoM(\boldsymbol{A}, \boldsymbol{B}) + DoM(\boldsymbol{B}, \boldsymbol{C}) \geq DoM(\boldsymbol{A}, \boldsymbol{B} \cup \boldsymbol{C})$ *and* $DoM(\boldsymbol{A}, \boldsymbol{B}) + DoM(\boldsymbol{A}, \boldsymbol{C}) \geq DoM(\boldsymbol{A}, \boldsymbol{B} \cup \boldsymbol{C})$*.*

Proof. Propositions *(a)–(d)* follow directly from the dominance move definition.
(e) By the definition of $DoM(\boldsymbol{B}, \boldsymbol{C})$, for any $\boldsymbol{c} \in \boldsymbol{C}$, there exists one $\boldsymbol{b} \in \boldsymbol{B}$ moving to $\boldsymbol{b'}$ to dominate $\boldsymbol{c}$ in $DoM(\boldsymbol{B}, \boldsymbol{C})$. Since $\boldsymbol{A} \preceq \boldsymbol{B}$, for $\boldsymbol{b}$ there exists one $\boldsymbol{a} \in \boldsymbol{A}$) which dominates $\boldsymbol{b}$. This means that the value of any objective of $\boldsymbol{a}$ is either equal to or smaller than that of $\boldsymbol{b}$. Therefore, the move distance from $\boldsymbol{a}$ to $\boldsymbol{b'}$ is equal to or smaller than that from $\boldsymbol{b}$ to $\boldsymbol{b'}$. Given this and $\boldsymbol{b'} \preceq \boldsymbol{c}$, we can construct a move for $\boldsymbol{A}$ to dominate $\boldsymbol{A}$ whose total distance is equal to or smaller than $DoM(\boldsymbol{B}, \boldsymbol{C})$. In addition, by the definition of the dominance move, this distance is greater than or equal to $DoM(\boldsymbol{A}, \boldsymbol{C})$. Thus $DoM(\boldsymbol{A}, \boldsymbol{C}) \leq DoM(\boldsymbol{B}, \boldsymbol{C})$.
The second inequality is proven analogously.
(f) Let $\boldsymbol{A_B}$ be the new position to which $\boldsymbol{A}$ moves in $DoM(\boldsymbol{A}, \boldsymbol{B})$. Then $\boldsymbol{A_B} \preceq \boldsymbol{B}$. By the proposition *(e)*, $DoM(\boldsymbol{A_B}, \boldsymbol{C}) \leq DoM(\boldsymbol{B}, \boldsymbol{C})$ follows. Let $\boldsymbol{A_{BC}}$ be the position to which $\boldsymbol{A_B}$ moves in $DoM(\boldsymbol{A_B}, \boldsymbol{C})$. Now, we can obtain a path from $\boldsymbol{A}$ to $\boldsymbol{A_B}$ and then from $\boldsymbol{A_B}$ to $\boldsymbol{A_{BC}}$ (for $\boldsymbol{A}$ to dominate $\boldsymbol{B} \cup \boldsymbol{C}$), whose move distance is equal to or smaller than $DoM(\boldsymbol{A}, \boldsymbol{B}) + DoM(\boldsymbol{B}, \boldsymbol{C})$. Likewise, by the dominance move definition, this distance is greater than or equal to $DoM(\boldsymbol{A}, \boldsymbol{B} \cup \boldsymbol{C})$. Thus $DoM(\boldsymbol{A}, \boldsymbol{B}) + DoM(\boldsymbol{B}, \boldsymbol{C}) \geq DoM(\boldsymbol{A}, \boldsymbol{B} \cup \boldsymbol{C})$. □

Proposition *(b)* implies a fundamental binary quality indicator in comparing two solution sets: whenever one solution set $\boldsymbol{P}$ is better than another set $\boldsymbol{Q}$, then $DoM(\boldsymbol{P}, \boldsymbol{Q}) < DoM(\boldsymbol{Q}, \boldsymbol{P})$. Proposition *(e)* brings the relation of two solution sets when they are individually compared with a third set.

The number of possibilities to find $\boldsymbol{P'}$ is numerous. Any combination of some $\boldsymbol{P'}$ can dominate $\boldsymbol{Q}$, considering (3.1). Let $\mathbb{P}^*(\boldsymbol{Q})$ be a set composed of all possible subsets from $\boldsymbol{Q}$, excluding the empty set. For determining each element of $\boldsymbol{P'}$, $\boldsymbol{p'_i}$, it is necessary to associate the corresponding element of $\boldsymbol{P}$ to elements of $\mathbb{P}^*(\boldsymbol{Q})$. The number of all possible associations, η, is the product of $|\boldsymbol{P}|$ and $|\mathbb{P}^*(\boldsymbol{Q})|$ and is given by

$$\begin{aligned} \eta &= |\boldsymbol{P}| \underbrace{\sum_{g=1}^{|\boldsymbol{Q}|} \binom{|\boldsymbol{Q}|}{g}}_{|\mathbb{P}^*(\boldsymbol{Q})|}, \\ &= |\boldsymbol{P}| \left(\binom{|\boldsymbol{Q}|}{1} + \binom{|\boldsymbol{Q}|}{2} + \ldots + \binom{|\boldsymbol{Q}|}{|\boldsymbol{Q}|} \right), \\ &= |\boldsymbol{P}|(2^{|\boldsymbol{Q}|} - 1). \end{aligned} \tag{3.2}$$

Let $\mathcal{Q}_s$ be an element of $\mathbb{P}^*(\boldsymbol{Q})$ for $s = \{1, \ldots, |\mathbb{P}^*(\boldsymbol{Q})|\}$, in which $\mathbb{P}^*(\boldsymbol{Q})$ represents the power set excluding the empty set ($\mathbb{P}^*(\boldsymbol{Q}) \cup \emptyset = \mathbb{P}(\boldsymbol{Q})$). Observe that $\boldsymbol{p}_i$ will be used as a reference point to generate $\boldsymbol{p}'_i$. In this way, $\boldsymbol{p}'_i$ may dominate $\mathcal{Q}_s$

Regardless of this challenge, the original DoM paper also proposes an exact calculation method for the bi-objective case, which is meticulously detailed in Li and Yao [2019]. For the sake of completeness and clarification, an overview of the algorithm is described next. The algorithm employs the concept of *inward neighbor*:

Definition 6. *Inward neighbor: Consider* $\boldsymbol{P}$ *a solution set,* $\boldsymbol{a} \in \boldsymbol{P}$, $\boldsymbol{b} \in \boldsymbol{P}$, *and* $\boldsymbol{b} \neq \boldsymbol{a}$. $n_R(\cdot)$ *is a function which determines* $\boldsymbol{b}$ *as the inward neighbor of* $\boldsymbol{a}$, *denoted as* $\boldsymbol{b} = n_R(\boldsymbol{a})$, *if* $\boldsymbol{b}$ *has the smallest dominance move distance to* $\boldsymbol{a}$, *that is* $\boldsymbol{b} = \arg\min_{\boldsymbol{p} \in \boldsymbol{P} \setminus \boldsymbol{a}} d(\boldsymbol{p}, \boldsymbol{a})$.

Using the *inward neighbor* concept, the DoM algorithm, as presented in Li and Yao [2019], can be outlined as:

Step 1: Remove the dominated points in both $\boldsymbol{P}$ and $\boldsymbol{Q}$, separately. Remove the points of $\boldsymbol{Q}$ that are dominated by at least one point in $\boldsymbol{P}$.

Step 2: Denoting $\boldsymbol{R} = \boldsymbol{P} \cup \boldsymbol{Q}$, each point of $\boldsymbol{Q}$ in $\boldsymbol{R}$ is considered as a subset. For each point of $\boldsymbol{Q}$, find its *inward neighbor* $\boldsymbol{r} = n_R(\boldsymbol{q}_j)$ in $\boldsymbol{R}$. If the point $\boldsymbol{r} \in \boldsymbol{P}$, then merge $\boldsymbol{r}$ into the subset of $\boldsymbol{q_j}$, otherwise $\boldsymbol{r}$ does not belong to the subset, or it is already owned by a subset. At the first case, $\boldsymbol{q}_j$ and $\boldsymbol{r}$ are merged into the subset. At the second case, there is nothing to do.

Step 3: If there exists no point $\boldsymbol{q}_j \in \boldsymbol{Q}$ such that $\boldsymbol{q}_j = n_R(n_R(\boldsymbol{q}_j))$ (i.e., there is a loop between the points) in any subset, then the procedure ends and there is an optimal solution to the case.

Step 4: Otherwise, there is a loop in one or more subsets, then replace these solutions by their ideal solution (formed of the best of each objective in each

solution inside the loop or subset). This leads to a new $\boldsymbol{Q}$ denoted as $\boldsymbol{Q'}$, then find the *inward neighbor* of such ideal point $\boldsymbol{P} \cup \boldsymbol{Q'}$ and group then. Return to step 3 until finish.

The definitions, theorems, and corollaries to prove that this algorithm is correct in the bi-objective case are presented in Li and Yao [2019]. Furthermore, DoM is Pareto dominant compliant, and any prior problem knowledge and pre-defined parameter are not necessary. However, due to the combinatorial nature of the problem, the authors stated that there is no solution for three or more objectives.

3.3 The MIP dominance move calculation approach

3.3.1 The model intuition

The main goal is to create a DoM formulation to deal with problems having three or more objective functions. Our DoM calculation perspective is based on the view that the problem is, in fact, an instance of an assignment problem with two levels and some restrictions. It is considered that to treat the question, we have to find an assignment of $\boldsymbol{P}$ to $\boldsymbol{Q}$ with the restrictions that every $\boldsymbol{q}_j$ must be assigned to exactly one $\boldsymbol{p}_i$ with the minimum distance. Nevertheless, in classic assignment problems, it is not possible that points in $\boldsymbol{P}$ change their own features, such as changing positions to alter their distances. In the proposed DoM calculation, this issue is considered.

The problem can be modeled as a mixed integer linear programming (MIP) approach and using the DoM definition. $\boldsymbol{P}$ and $\boldsymbol{Q}$ are sets of points, with points $\boldsymbol{p}_i$, $i \in \{1, \ldots, |\boldsymbol{P}|\}$ and points $\boldsymbol{q}_j$, $j \in \{1, \ldots, |\boldsymbol{Q}|\}$, respectively. Each $\boldsymbol{p}_i$ is composed of $p_{(i,m)}$ components, $1 \leq m \leq M$ in which M is the number of objective functions. Analogously, each $\boldsymbol{q}_j$ is composed of $q_{(j,m)}$ components, $1 \leq m \leq M$.

Given $\boldsymbol{P}$ and $\boldsymbol{Q}$, $\boldsymbol{P'}$ is a set of points, in which each $\boldsymbol{p}'_i$ is generated from $\boldsymbol{p}_i$ after some updates in one or more objectives to dominate a $\mathcal{Q}_s$ (an element of $\mathbb{P}^*(\boldsymbol{Q})$). It is important to note that if $\boldsymbol{P}$ already dominates $\boldsymbol{Q}$, then $\boldsymbol{p}'_i = \boldsymbol{p}_i$ for all i. Observe that $\boldsymbol{P'}$ must dominate $\boldsymbol{Q}$, resulting in a better distance such as expressed in Equation (3.1).

The proposed MIP model calculates the distance $d(\boldsymbol{p}_i, \boldsymbol{p}'_i)$. In order to calculate this distance, a new point, $\hat{\boldsymbol{p}}_i$, is obtained and there is strict relation between $\hat{\boldsymbol{p}}_i$ and $\boldsymbol{p}'_i$. At the end, the MIP-DoM model calculates the final $\boldsymbol{p}'_i$ using $\hat{\boldsymbol{p}}_i$.

Figure 3.1 illustrates the intuition behind our proposal in a bi-objective problem. The $d(\boldsymbol{p}_i, \boldsymbol{p}'_i)$ is composed of two terms, $zp_{(i,m)}$ and $zpq_{(i,j,m)}$. The summation

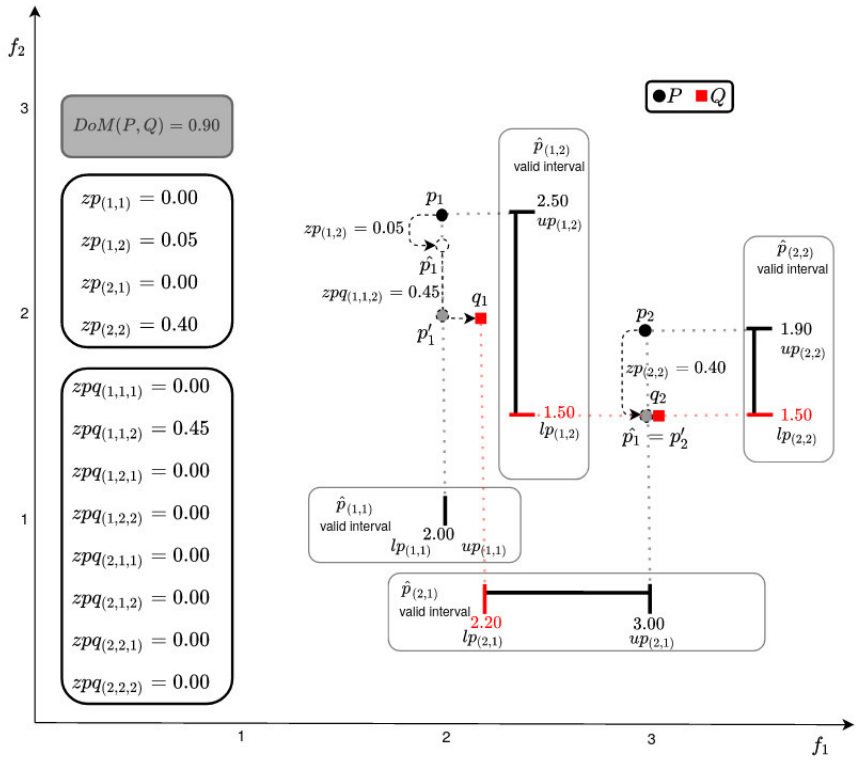

Figure 3.1. An example that shows up the intuition behinds the MIP-DoM approach. Consider two sets, $\boldsymbol{P} = \{(2.0, 2.5), (3.0, 1.9)\}$ and $\boldsymbol{Q} = \{(2.2, 2.0), (3.0, 1.5)\}$. The DoM value indicates the total movement for $\boldsymbol{P}$ to dominate $\boldsymbol{Q}$. A typical vertical movement, $zp_{(1,2)} = 0.05$ represents an improvement in f_2 objective generating $\hat{\boldsymbol{p}}_1$. However, $\hat{\boldsymbol{p}}_1$ does not dominate $\boldsymbol{q}_1$. There is still a movement to be done in f_2 objective, represented by $zpq_{(1,1,2)} = 0.45$, so that $\boldsymbol{p}'_1$ will dominate $\boldsymbol{q}_1$. Observe that point $\boldsymbol{p}_2$ does not dominate $\boldsymbol{q}_2$, so it is vertically moved obtaining $\hat{\boldsymbol{p}}_2$, in an attempt to dominate $\boldsymbol{q}_2$. With this vertical movement, indicated by $zp_{(2,2)} = 0.40$, $\hat{\boldsymbol{p}}_2$ dominates $\boldsymbol{q}_2$, becoming $\boldsymbol{p}'_2$. The DoM value is the total distance using the terms $zp_{(1,2)}, zp_{(2,2)}$, and $zpq_{(1,1,2)}$. Limits on $\boldsymbol{p'}$ movements (lower ($lp_{(i,m)}$) and upper bounds ($up_{(i,m)}$)) are also indicated.

of all $zp_{(i,m)}$, $\forall\ m \in \{1, \ldots, M\}$ results in $d(\boldsymbol{p}_i, \hat{\boldsymbol{p}}_i)$, Analogously, the summation of all $zpq_{(i,j,m)}$, $\forall\ m \in \{1, \ldots, M\}\ \forall j \in \{1, \ldots, |\boldsymbol{Q}|\}$ results in $d(\hat{\boldsymbol{p}}_i, \boldsymbol{p}'_i)$. $\hat{\boldsymbol{p}}_i$, indicated as unfilled circles, represents the point after the first movement. For each $\boldsymbol{p}_i$ and each m-th objective, there are a lower ($lp_{(i,m)}$) and upper ($up_{(i,m)}$) bounds constraining $\hat{\boldsymbol{p}}_i$'s movement. The bounds are generated considering that the $\hat{p}_{(i,m)}$ needs to be smaller or equal to the minimum value in the $\boldsymbol{Q}$ taking the m-th objective into account (lower bound), and cannot be greater than $p_{(i,m)}$ (upper bound). The component $\hat{p}_{(i,m)}$ acts as an extra variable in our MIP model that leverages the calculation. After finding a $\hat{p}_{(i,m)}$ within the corresponding bounds, $zp_{(i,m)}$ represents the extent of movement. In Figure 3.1, there are $\hat{\boldsymbol{p}}_1$ and $\hat{\boldsymbol{p}}_2$, represented as unfilled circles and there are some improvements in f_2 objective generating the $zp_{(1,2)} = 0.05$ and $zp_{(2,2)} = 0.40$.

The $zpq_{(i,j,m)}$ defines the move in m-th objective for the i-th component of $\hat{p}_{(i,m)}$ to generate the $p'_{(i,m)}$. In Figure 3.1, the improvement generated by the $\hat{\boldsymbol{p}}_2$ is good enough to dominate $\boldsymbol{q}_2$. In that case, $zpq_{(2,2,1)}$ and $zpq_{(2,2,2)}$ are equal to zero, considering that $\hat{\boldsymbol{p}}_2 = \boldsymbol{p}'_2$. The same does not happen for $\hat{\boldsymbol{p}}_1$; there is still a distance such that $\hat{\boldsymbol{p}}_1$ must be moved to dominate $\boldsymbol{q}_1$. This distance is indicated by the $zpq_{(1,1,2)} = 0.45$. The final $\boldsymbol{p}'_1$ is obtained using $\hat{\boldsymbol{p}}_1$ and $zpq_{(1,1,2)}$.

In a way to clarify the relationship between $\boldsymbol{zp}$, $\boldsymbol{zpq}$ and $\hat{\boldsymbol{p}}$ variables and their roles, the example in Figure 3.1 is slightly changed. In Figure 3.2 there are two sets, $\boldsymbol{P}$ = {(2.0, 2.5), (3.0, 1.9)} and $\boldsymbol{Q}$ = {(2.2, 2.0), (3.0, 1.5), (1.95,2.45)}. A new solution in $\boldsymbol{Q}$ given by $q_3 = (1.95, 2.45)$ is added, compared with the base case shown in Figure 3.1. Therefore, considering the Figure 3.2, for calculating the total minimum movement for $\boldsymbol{P}$ to dominate $\boldsymbol{Q}$, p_1 moves to dominate q_1 and q_3. The $\boldsymbol{zp}$ variables take the following values: $zp_{(1,2)} = 0.50$, and $zp_{(2,2)} = 0.40$. The $\boldsymbol{zpq}$ variables take the following value: $zpq(1, 3, 1) = 0.05$. Finally, MIP-DoM$(P, Q) = 0.95$.

Based on the Figure 3.2, one first question would be whether it is possible to solve the problem using only the $\boldsymbol{zpq}$ variable. Using these approach, the answer is no. The reasons will be indicated using Figure 3.3.

In Figure 3.3, the total movement value for calculation DoM is a $\boldsymbol{zpq}$ summation. For each member of $\boldsymbol{Q}$, there is a $\boldsymbol{zpq}$ component which represents the move that a member of $\boldsymbol{P'}$ does to dominate a $\boldsymbol{Q}$ member. For example, $\boldsymbol{p'_{1,1}}$ is generated from $\boldsymbol{p_1}$ to dominate $\boldsymbol{q_1}$ with a $zpq_{(1,1,1)} = 0$ and $zpq_{(1,1,2)} = 0.50$. At the same way, the $\boldsymbol{p'_{1,3}}$ is generated to dominate $\boldsymbol{q_3}$ with a $zpq_{(1,3,1)} = 0.05$ and $zpq_{(1,3,2)} = 0.05$. Finally, this happens as well with $\boldsymbol{p'_{2,2}}$ generated from $\boldsymbol{p_2}$ to dominate $\boldsymbol{q_2}$ with $zpq_{(2,2,1)} = 0$ and $zpq_{(2,2,2)} = 0.40$.

Observe that the total $\boldsymbol{zpq}$ summation does not represent the minimum movement value. A decrease in this value is possible if $zpq_{(1,3,2)}$ and $zpq_{(1,1,2)}$ share the same extent of 0.05. Geometrically, it is possible to use $zpq_{(1,1,2)} = 0.50$ which dominates $\boldsymbol{q_1}$, from that point, a new improvement can be done to dominate $\boldsymbol{q_3}$. Considering this point of view, it is necessary to share information that $\boldsymbol{p_1}$ will dominate $\boldsymbol{q_1}$ and $\boldsymbol{q_3}$ at the same time, thus sharing some extent of movement.

The $\boldsymbol{zpq}$ variable can not deal with this situation because it informs explicitly what members of $\boldsymbol{P}$ and $\boldsymbol{Q}$ are used to calculate the movement.

As an attempt to solve the problem raised in the last paragraph, some new variables are introduced in the model: $\boldsymbol{zp}$ and $\hat{\boldsymbol{p}}$. The idea is: whenever there is an opportunity to share some extent of movement, it must be done. The $\hat{\boldsymbol{p}}$ is proposed to perform such task, and $\boldsymbol{zp}$ measure the extent of movement.

A first observation is how the $\boldsymbol{zp}$ movement can be done. In Figure 3.4, the gray

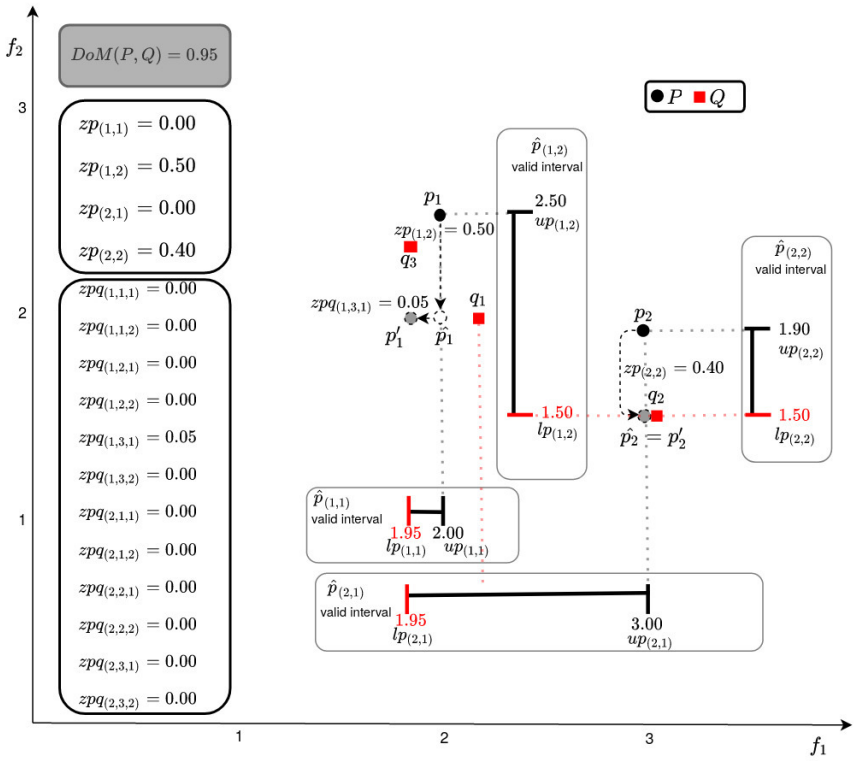

Figure 3.2. A slightly altered example based in Figure 3.1. Consider two sets, $\boldsymbol{P} = \{(2.0, 2.5), (3.0, 1.9)\}$ and $\boldsymbol{Q} = \{(2.2, 2.0), (3.0, 1.5), (1.95,2.45)\}$. The DoM value indicates the total minimum movement for $\boldsymbol{P}$ to dominate $\boldsymbol{Q}$. A typical vertical movement, $zp_{(1,2)} = 0.50$ represents an improvement in f_2 objective generating $\hat{\boldsymbol{p}}_1$. However, $\hat{\boldsymbol{p}}_1$ does dominate $\boldsymbol{q}_1$, but not dominate $\boldsymbol{q}_3$. There is still a movement to be done in f_1 objective, represented by $zpq_{(1,3,1)} = 0.05$, so that $\boldsymbol{p}_1'$ will dominate $\boldsymbol{q}_1$ and $\boldsymbol{q}_3$. Observe that point $\boldsymbol{p}_2$ does not dominate $\boldsymbol{q}_2$, so it is vertically moved obtaining $\hat{\boldsymbol{p}}_2$, in an attempt to dominate $\boldsymbol{q}_2$. With this vertical movement, indicated by $zp_{(2,2)} = 0.40$, $\hat{\boldsymbol{p}}_2$ dominates $\boldsymbol{q}_2$, becoming $\boldsymbol{p}_2'$.

box shows the area in which each $\boldsymbol{zp}$ can take values. For $\boldsymbol{p_1}$, for example, $zp(1,1)$ and $zp(1,2)$ can be extended considering the whole gray area indicated. The same happens to $\boldsymbol{p_2}$, using $zp(2,1)$ and $zp(2,2)$.

Conceptually, the $\boldsymbol{zp}$ will store some movement extent in which some $\boldsymbol{p}$ is doing as indicated in the gray area. In fact, the $\boldsymbol{zp}$ variable is applied to 'reuse' such extent of a movement. For example in Figure 3.2, there is p_1 and $\hat{p}_1$ with $zp(1,2) = 0.50$. If a comparison is made with Figure 3.3, in which only $\boldsymbol{zpq}$ are used, it is possible to observe a reduction of 0.05 considering the moves related to $\boldsymbol{q_1}$ and $\boldsymbol{q_3}$. Therefore, the final solution will be composed by a movement represented in $zp(1,2) = 0.50$ that dominates $\boldsymbol{q_1}$ and a extent $zpq_{(1,3,2)} = 0.05$ that dominates $\boldsymbol{q_3}$.

In this sense, there is a MIP-DoM value composed by: $zp(1,2) = 0.50$, $zpq_{(1,3,2)} = 0.05$ from $\boldsymbol{p_1}$ and $zp(2,2) = 0.40$ from $\boldsymbol{p_2}$.

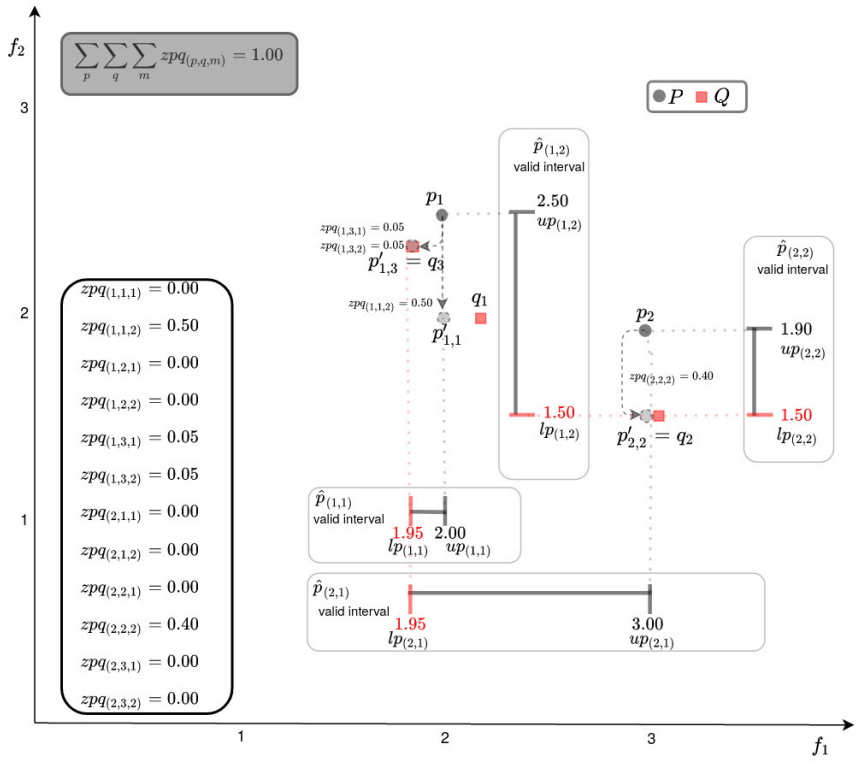

Figure 3.3. A distance value using only the zpq variable. Using only this variable there is zpq value for each dominance from P to Q in each objective. From p_1, the $p'_{1,1}$ and $p'_{1,3}$ are generated in a move to dominate q_1 and q_3, respectively. The same happens in p_2, generating $p'_{2,2}$ to dominate q_2. Observe that the total summation does not represent the minimum summation move.

In general, the motivation for zp and its role in the model can be stated as: It represents a partial movement generating a $\hat{p}$ constrained to a box interval, as indicated in the gray box in Figure 3.4.

Observe that the zp variable does not carry any directly information from Q, in a dominance sense. We argue that it is not possible to solve the problem using only zp. In fact, the zp variable will be directly connected with $\hat{p}$. All values in zp come from the $\hat{p}$'s alternatives.

The $\hat{p}$ values are calculated inside the model. It represents a partial movement generating a $\hat{p}$ and the extent of movements expressed in zp.

The zpq is still related to $\hat{p}$, since there are some constraints relating both variables. In fact, if a p generates $\hat{p}$, the movement is expressed in zp, and, if there is some extent of movement to dominate any Q, it will be expressed in the zpq variable.

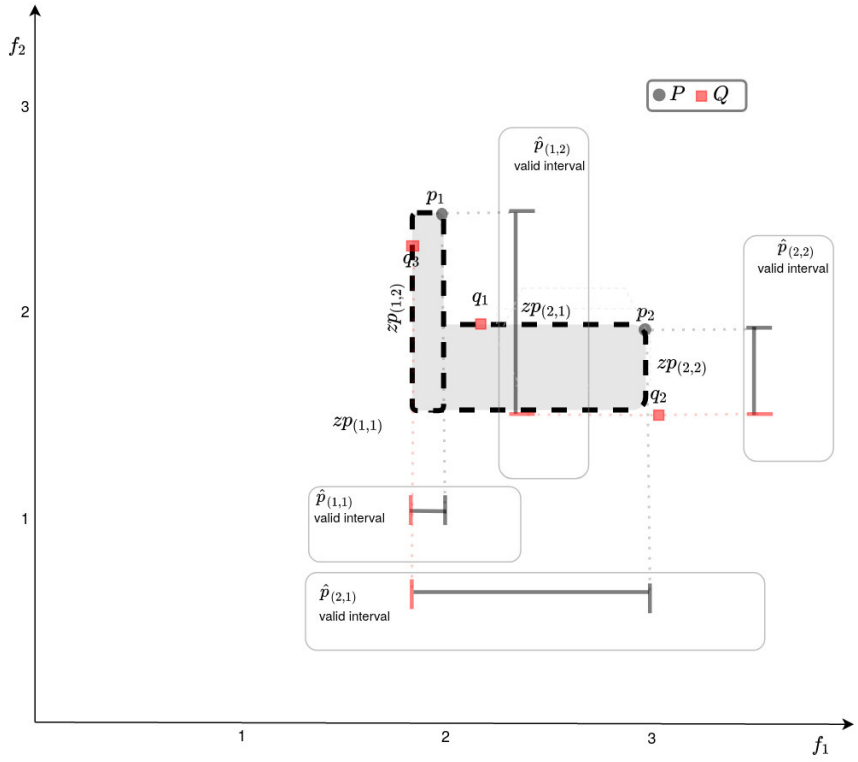

Figure 3.4. The $\boldsymbol{zp}$ possibilities are indicated in the gray boxes. Each $\boldsymbol{zp}$ is linked to its $\boldsymbol{p}$ and the extent of the movement is constrained by the box areas.

3.3.2 The mathematical model

To facilitate the presentation, we first summarize our notation below.

Index:

i: i-th solution from $\boldsymbol{P}$;

j: j-th solution from $\boldsymbol{Q}$;

m: m-th objective in M.

Variables:

$zp_{(i,m)}$: Manhattan distance from $\boldsymbol{p}$ to $\hat{\boldsymbol{p}}$ on the i-th solution on m-th objective;

$zpq_{(i,j,m)}$: Manhattan distance from $\hat{\boldsymbol{p}}_{\boldsymbol{i}}$ to $\boldsymbol{q}_{\boldsymbol{j}}$ in the i-th solution from $\boldsymbol{P}$, j-th solution from $\boldsymbol{Q}$, on m-th objective;

$\hat{p}_{(i,m)}$: an improvement made in $p_{(i,m)}$ on the i-th solution on m-th objective;

$xp_{(i)}$: a binary variable indicating that the i-th solution from $\boldsymbol{P}$ is been used or not;

$xpq_{(i,j)}$: a binary variable indicating that the i-th solution from $\boldsymbol{P}$ is dominating or not the j-th solution from $\boldsymbol{Q}$;

$xpqd_{(i,j,m)}$: binary variable to guarantee that $\hat{p}_{(i,m)}$ will have a value less than $q_{(j,m)}$.

Parameters:

$M_{i,j,m}$: Maximum value between 0 and $p_{(i,m)} - q_{(j,m)}$;

$up_{i,m}$: It assumes the upper bound to $\hat{p}_{(i,m)}$ equal to $p_{(i,m)}$;

$lp_{i,m}$: It assumes the lower bound to $\hat{p}_{(i,m)}$ as Min($p_{(i,m)}$, Min($q_{(1..|\boldsymbol{Q}|,m)}$)).

$$\text{minimize} \quad \sum_{i=1}^{|\boldsymbol{P}|}\sum_{m=1}^{M} zp_{(i,m)} + \sum_{i=1}^{|\boldsymbol{P}|}\sum_{j=1}^{|\boldsymbol{Q}|}\sum_{m=1}^{M} zpq_{(i,j,m)} \tag{3.3}$$

subject to:

$$\begin{cases} zp_{(i,m)} \geq p_{(i,m)} xp_{(i)} - \hat{p}_{(i,m)}, \\ zp_{(i,m)} \leq p_{(i,m)} xp_{(i)}, \end{cases} \quad \forall i, \forall m, \tag{3.4}$$

$$\begin{cases} zpq_{(i,j,m)} \geq \hat{p}_{(i,m)} - q_{(j,m)} - p_{(i,m)}(1 - xpq_{(i,j)}), \\ zpq_{(i,j,m)} \leq \hat{p}_{(i,m)} - q_{(j,m)} + (M_{(i,j,m)} - lp_{(i,m)} - q_{(j,m)}).(1 - xpqd_{(i,j,m)}), \\ zpq_{(i,j,m)} \leq M_{(i,j,m)}(xpqd_{(i,j,m)}), \end{cases} \quad \forall i, \forall j, \forall m, \tag{3.5}$$

$$\left\{ lp_{(i,m)} \leq \hat{p}_{(i,m)} \leq up_{(i,m)}, \forall i, \forall m \right. \tag{3.6}$$

$$\left\{ xp_{(i)} \geq xpq_{(i,j)}, \quad \forall i, \forall j \right. \tag{3.7}$$

$$\begin{cases} xp_{(i)} \leq \sum_{j=1}^{|\boldsymbol{Q}|} xpq_{(i,j)}, & \forall i, \end{cases} \tag{3.8}$$

$$\begin{cases} \sum_{i=1}^{|\boldsymbol{P}|} xpq_{(i,j)} = 1, & \forall j, \end{cases} \tag{3.9}$$

$$\begin{cases} xp_{(i)} \in \{0,1\}, \\ xpq_{(i,j)} \in \{0,1\}, \\ xpqd_{(i,j,m)} \in \{0,1\}, \end{cases} \quad \forall i, \forall j, \forall m, \tag{3.10}$$

where, $i = 1, \ldots, |\boldsymbol{P}|, j = 1, \ldots, |\boldsymbol{Q}|, m = 1, \ldots, M$.

The distances are calculated using an MIP model expressed from Equation (3.3) to (3.10). The model contains continuous **z**-variables (zp and zpq) and binary **x**-variables (discussed later), and a number of constraint sets, which are combined in an

attempt to find the minimum DoM value. The objective function uses **z**-variables. In a practical sense, the inclusion of zp and zpq avoids equality constraints in the model, transforming an equality constraint into an inequality one. There is a constraint set related to $zp_{(i,m)}$, and it is expressed in the set of Equations (3.4). Essentially, it must be greater than or equal to zero and it must be less than $p_{(i,m)}$ and, finally, it must be greater than or equal the difference between $p_{(i,m)}$ and $\hat{p}_{(i,m)}$.

In the same way, the variable $zpq_{(i,j,m)}$ represents the difference that each $\hat{p}_{(i,m)}$ still have to improve to generate the $p'_{(i,m)}$ candidate in an attempt to be less than or equal to some $q_{(j,m)}$. The $zpq_{(i,j,m)}$ can be zero if $\hat{p}_{(i,m)}$ is equal to $p'_{(i,m)}$, or greater than zero if $\hat{p}_{(i,m)}$ is greater than $p'_{(i,m)}$. In fact, $zpq_{(i,j,m)}$ variable can be seen as a penalty value for those solutions that still do not dominate a $\mathcal{Q}_s$.

The constraint set in Equation (3.5) ensures that if $\hat{p}_{(i,m)}$ is not less than or equal to $q_{(j,m)}$, then there is a valid value in $zpq_{(i,j,m)}$. It will receive $\max(0, \hat{p}_{(i,m)} - q_{(j,m)})$, if the difference to be less than $q_{(j,m)}$ or 0, otherwise. A binary variable, $xpq_{(i,j)}$ is used to reach out this model condition. Thus, $xpqd_{(i,j,m)}$ assumes 1 to guarantee the maximum value; otherwise, the solution will be infeasible. Moreover, we have used a linearization technique to obtain the maximum function.

The $\hat{p}_{(i,m)}$ is a continuous variable and the constraint set expressed in Equation (3.6) presents the interval in which $\hat{p}_{(i,m)}$ lies within.

Constraints represented in Equations (3.7) and (3.8) associates the binary variables $xp_{(i)}$ and $xpq_{(i,j)}$ guaranteeing that if $xp_{(i)}$ is 'active', at least one $xpq_{(i,j)}$ would be 'active' as well. Equation (3.7) asserts that a candidate $\hat{\boldsymbol{p}}_i$ coming from $\boldsymbol{p}_i$ will try to dominate $\mathcal{Q}_s$, and in this sense, $xp_{(i)}$ will be equal to one and some $xpq_{(i,j)}$ will be also equal to one.

The constraints which exclusively deal with binary variables can be initially formulated using propositional logic. The model needs to know which $\boldsymbol{p_i}$ is trying to dominate $\boldsymbol{q_j}$. The binary variables $xp_{(i)}$ and $xpq_{(i,j)}$ are used to guarantee this information. We are interested in the unique association between $xp_{(i)}$ and $xpq_{(i,j)}$, and it can be expressed as $xpq_{(i,j)} \leftrightarrow xp_{(i)}$. Observe this is a bi-conditional logical connective, implying $xpq_{(i,j)} \rightarrow xp_{(i)}$ and $xp_{(i)} \rightarrow xpq_{(i,j)}$ are both valid.

The constraint described in Equation (3.7) of our model ensure that if $xp_{(i)}$ is 'active', at least one $xpq_{(i,j)}$ must be 'active' as well ($xpq_{(i,j)} \rightarrow xp_{(i)}$). In propositional logic, this can be written as

$$\bigvee_{j=1}^{|\boldsymbol{Q}|} xpq_{(i,j)} \rightarrow xp_{(i)}.$$

Applying some logic rules, the inequality represented by Equation (3.7) can be derived

as in Equation 3.11.

$$\begin{aligned}
& \bigvee_{j=1}^{|\boldsymbol{Q}|} xpq_{(i,j)} \rightarrow xp_{(i)} && \\
& && \text{(Implication in terms of } \vee\text{)}, \\
\Leftrightarrow \quad & \neg\left(\bigvee_{j=1}^{|\boldsymbol{Q}|} xpq_{(i,j)}\right) \vee xp_{(i)}, && \forall i \\
& && \text{(DeMorgan's Laws)} \\
\Leftrightarrow \quad & \bigwedge_{j=1}^{|\boldsymbol{Q}|} \left(\neg xpq_{(i,j)}\right) \vee xp_{(i)}, && \forall i \\
& && \text{(Distributive Law)} \\
\Leftrightarrow \quad & \bigwedge_{j=1}^{|\boldsymbol{Q}|} \left(\neg xpq_{(i,j)} \vee xp_{(i)}\right), && \forall i \\
\Leftrightarrow \quad & \left(1 - xpq_{(i,j)} + xp_{(i)}\right) \geq 1, && \forall i, \forall j, \\
\Leftrightarrow \quad & xp_{(i)} \geq xpq_{(i,j)}, && \forall i, \forall j. \qquad (3.11)
\end{aligned}$$

Analogously, the conditional proposition $xp_{(i)} \rightarrow xpq_{(i,j)}$ is valid and written in propositional logic as

$$\bigvee_{j=1}^{|\boldsymbol{Q}|} xpq_{(i,j)}.$$

Applying some logic rules, we can obtain the inequality expressed in Equation (3.8) of our model as in Equation (3.12):

$$\begin{aligned}
& xp_{(i)} \rightarrow \bigvee_{j=1}^{|\boldsymbol{Q}|} xpq_{(i,j)} && \\
& && \text{(Implication in terms of } \vee\text{)} \\
\Leftrightarrow \quad & \left(\neg xp_{(i)}\right) \vee \left(\bigvee_{j=1}^{|\boldsymbol{Q}|} xpq_{(i,j)}\right), && \forall i \\
\Leftrightarrow \quad & \left(1 - xp_{(i)}\right) + \sum_{j=1}^{|\boldsymbol{Q}|} xpq_{(i,j)} \geq 1, && \forall i
\end{aligned}$$

$$\Leftrightarrow \quad xp_{(i)} \leq \sum_{j=1}^{|\boldsymbol{Q}|} xpq_{(i,j)}, \qquad \forall i. \tag{3.12}$$

It completes the unique association between $xp_{(i)}$ and $xpq_{(i,j)}$. These two binary variables can give the information about which $\boldsymbol{p_i}$ and $\boldsymbol{q_j}$ are involved in the optimal solution.

Finally, the constraint set in Equation (3.10) presents all the binary variables used in the model. The $xp_{(i)}$ is used to guarantee that if a $\hat{p}_{(i,m)}$ is used in the model, we will know exactly which $p_{(i,m)}$ has generated it. In the same way, $xpq_{(i,j)}$ indicates that $\boldsymbol{p_i}$ generated $\hat{\boldsymbol{p}}_i$ and is trying to dominate a $\boldsymbol{q}_j$. These two binary variables are used to guarantee that at least one $\boldsymbol{p}_i$ will be associated with a $\boldsymbol{q}_j$.

Different MIP solvers can compute the proposed model such as CPLEX [Klotz and Newman, 2013], GUROBI [Gurobi Optimization, 2019], and SCIP [Gamrath et al., 2020], just to name a few. However, there is no guarantee that the problems can be solved optimally in a non-prohibitive computational time. Just to have an idea of the model size regarding the number of variables and constraints, consider the problem having $\boldsymbol{P} = \{(2.0, 2.5), (3.0, 1.9)\}$ and $\boldsymbol{Q} = \{(2.2, 2.0), (3.1, 1.5)\}$, depicted in Figure 3.1. In this case, the model has 16 continuous variables, 22 binary variables, and 60 constraints (including non-negativity constraints). Considering a generic problem, the number of continuous variables, binary variables, and constraints are detailed in the Equations (3.13) to (3.15), respectively:

$$\begin{aligned}
\#\text{ continuous variables} &= (2 + |\boldsymbol{Q}|)(|\boldsymbol{P}|M), && (3.13)\\
\#\text{ binary variables} &= (|\boldsymbol{P}|(1 + |\boldsymbol{Q}|(1 + 2M)), && (3.14)\\
\#\text{ constraints} &= |\boldsymbol{Q}| + |\boldsymbol{P}|(1 + 3M) + \\
&\quad ((|\boldsymbol{P}||\boldsymbol{Q}|)(3 + 4M)). && (3.15)
\end{aligned}$$

As an example, considering $M = 5$, $|\boldsymbol{P}| = |\boldsymbol{Q}| = 200$, there are 202,002 continuous variables, 440,200 binary variables, and 923,400 constraints, making the MIP problem a relatively large-sized optimization problem.

3.4 Experiments

3.4.1 Bi-objective case

The first test aims to show how MIP-DoM calculation addresses the four quality indicator facets: convergence, spread, uniformity, and cardinality [Li and Yao, 2019]. The same experiments proposed in [Li and Yao, 2017] to solve DoM in the bi-objective case are adopted to verify the model.

Figure 3.5 presents the controlled experiments for assessing the model correctness. Each facet of quality indicators, convergence, cardinality, uniformity, and spread, is represented in a 'row' in Figure 3.5. Each row has two plots corresponding to the two examples. The examples are slightly different from one another. In each plot, there are two solution sets, $\boldsymbol{P}$ and $\boldsymbol{Q}$.

Convergence is an important factor to reflect Pareto-dominance compliance of sets. This behavior is shown considering two examples in Figure 3.5, first row: *test = convergence*. In Examples *I* and *II*, the number of points in $\boldsymbol{P}$ and $\boldsymbol{Q}$ are equal, but some points of $\boldsymbol{P}$ dominate some points of $\boldsymbol{Q}$. In plot *I*, $\boldsymbol{P}$ completely dominates $\boldsymbol{Q}$, hence, dominance move from $\boldsymbol{P}$ to $\boldsymbol{Q}$, DoM($\boldsymbol{P}$,$\boldsymbol{Q}$), is zero. In plot *II*, since some points of $\boldsymbol{Q}$ are dominated by $\boldsymbol{P}$, DOM($\boldsymbol{P}$,$\boldsymbol{Q}$) is smaller than ($\boldsymbol{Q}$,$\boldsymbol{P}$). There is still a movement sign (gray arrow) in plot *II* indicating the movement for one set to dominate the other. Observe that $\boldsymbol{q}_4$ needs to move to $\boldsymbol{q}'_4$, thereby dominating $\boldsymbol{P}$. On the same way, $\boldsymbol{p}_1$ needs to move to $\boldsymbol{p}'_1$ to dominate $\boldsymbol{Q}$. Importantly, these are the *minimum* movement needs to achieve a dominance between the two sets.

DoM prefers solutions with a different cardinalities in $\boldsymbol{P}$ and $\boldsymbol{Q}$. In Figure 3.5, *test = cardinality*, graphics *III* and *IV*, it can be observed that the solution sets have the same convergence, and spread. In graphic *III*, $\boldsymbol{Q}$ has one more point than $\boldsymbol{P}$. In graphic *IV*, $\boldsymbol{P}$ has one more point than $\boldsymbol{Q}$. The DoM values of both graphics reflected the cardinality aspect.

Uniformity indicates the preference for evenly distributed points. The solution sets in Figure 3.5, *test = uniformity*, presented this feature. The sets have the same convergence, spread, and cardinality. In graphics *V* and *VI*, the set $\boldsymbol{P}$ is distributed uniformly, and $\boldsymbol{Q}$ has a random distribution. The DoM values show that $\boldsymbol{Q}$ needs a bigger move value (DoM) to dominate $\boldsymbol{P}$. In graphic *VI*, the density of points in set $\boldsymbol{Q}$ increased gradually from bottom to top (considering the f_1). Again, a more uniformly distributed set of points needs a smaller DoM to move to a more non-uniformly distributed set of points.

Finally, DoM must exhibit its preference for solutions having a better spread.

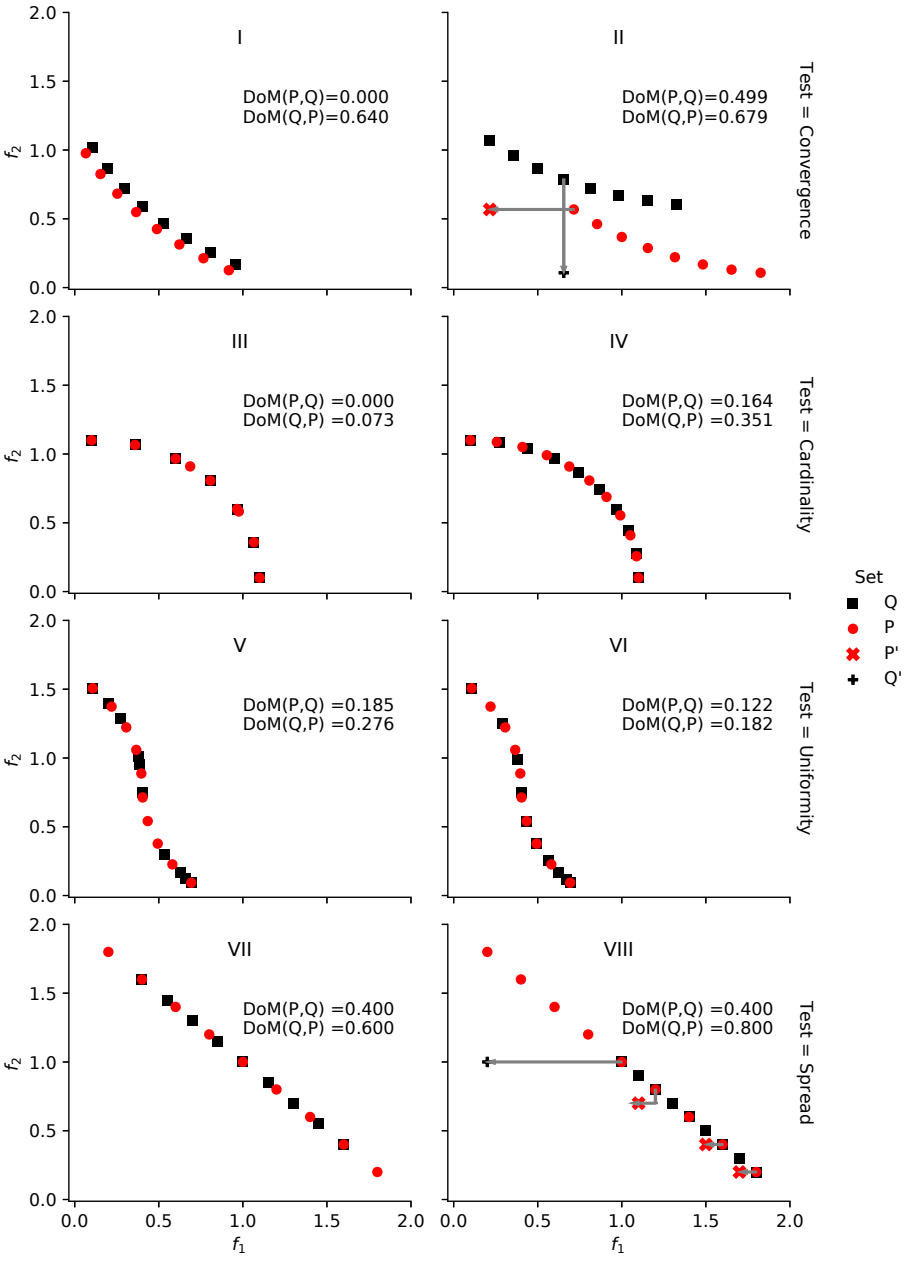

Figure 3.5. Experiments proposed in [Li and Yao, 2017] to assess the four facets of quality indicators: convergence, cardinality, uniformity, and spread.

A set with better extensity (more extreme points) has a smaller dominance move compared to its competitor. In Figure 3.5, *test* = *spread*, considering graphic *VII*, set $\boldsymbol{Q}$ was generated by shrinking $\boldsymbol{P}$ a little (more concentrated in the middle). In graphic *VIII*, set $\boldsymbol{P}$ was distributed uniformly in the range, while $\boldsymbol{Q}$ assumed five right bottom points. The graphic *VIII* still presents the $\boldsymbol{q}_1$ movement generating $\boldsymbol{q}'_1$ to dominate some elements from $\boldsymbol{P}$, which were not yet dominated. On the same way $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ and $\boldsymbol{p}_4$, generate $\boldsymbol{p}'_1$, $\boldsymbol{p}'_2$ and $\boldsymbol{p}'_4$ to dominate $\boldsymbol{Q}$.

The MIP-DoM approach results are the same as the results reported in [Li and Yao, 2017] obtained using the proposed bi-objective algorithm.

3.4.2 Multi- and Many-objective Problems

Following the initial and controlled test problems, it is crucial to test the MIP-DoM model with standard test problem sets. First, a problem set with three objectives is compared with existing indicators, such as the additive ϵ-indicator, hypervolume (HV), IGD+, and visual plots to assess the results. Second, an attempt to solve problems with five, ten, and more objectives are made. In all tests, algorithms such as IBEA, MOEA/D, NSGA-III, NSGA-II, and SPEA2 are used to generate the solution sets. It is essential to highlight that the goal is to assess the proposed MIP-DoM approach's effectiveness and not compare the algorithm's performance, so any other algorithms could have been applied to generate the solution sets.

For each problem set, the population size for the algorithms needs to be initially chosen. We argue that this choice is relevant for the proposed model and is closely related to Equation (3.2). Considering a good approximation set of the Pareto front, in terms of convergence and uniformity, the number of non-dominated solutions grows exponentially concerning the dimension of the objective space. However, some works in many-objective optimization do not strictly follow this concept. In [Tian et al., 2018], for example, the number of objectives M is 3, 5, and 10, and the population size L is 105, 126, and 275, respectively. In [Yang et al., 2019], an efficient hypervolume calculation is provided, and some tests are done with M set to 3, 4, 5, and L between 10 and 200. Likewise, in CEC'2018, a competition on many-objective optimization [Cheng et al., 2018] M was chosen to be 5, 10, and 15, and the maximum population size was set to 240. Generally, in practical situations, the fronts are not so large.

Since Equation (3.2) plays an important role in the proposed model, related to the number of domination moves possible (Equations (3.3) to (3.10)), we have decided to validate DoM using $M = 3$, 5, 10, and 15 and $|\boldsymbol{P}| = |\boldsymbol{Q}| = 50$, 100, 170, and 240 indicating the final Pareto front approximation size.

All experiments are done using *Platypus* [Brockhoff and Tušar, 2019], *PyGMO* [Izzo, 2012], and pymoo [Blank and Deb, 2020a] to generate the problem sets and to calculate the IGD+ and ϵ-indicator. Hypervolume is calculated using the Walking Fish Group based algorithm [While et al., 2012] (considering its worst-case complexity as $O(M \times 2^L)$). The model (Equations (3.3) to (3.10)) is implemented using *Python* and it is solved using *GUROBI* [Gurobi Optimization, 2019] (version 9.0.0 build v9.0.0rc2) running on a Linux 64 bits operational system with 8 CPU's (Intel Xeon E5-2630 v4

2.2GHz) and 32Gb of RAM. Some *GUROBI* solver parameters [Gurobi Optimization, 2019] are altered: the minimum $gap = 10^{-6}\%$, and the $MIPFocus = 3$, which help to improve the best bound during the execution. For all models tested, the optimal solutions are always reached with the established gap.

3.4.2.1 Multi-objective Problems

To test our proposed approach, we select some well-known problem test sets in three dimensions, which come from *DTLZ* and *WFG* families. Some algorithms are used to generate the approximated Pareto front, and using the outcomes, the MIP-DoM indicator is compared to other quality indicators. Embedded with a similar experiment purpose presented in Li et al. [2019a], the NSGA-III and MOEA/D algorithms (Pareto-based and decomposition approach, respectively) are firstly chosen, and, afterwards, IBEA [Zitzler and Künzli, 2004], SPEA2 [Zitzler et al., 2001], and NSGA-II [Bradford et al., 2018]. All the approximation sets are shown in Figure 3.6.

In our experiments, the maximum number of fitness evaluations is set to 10,000, and each algorithm is executed 21 times. Three unary quality indicators, the additive ϵ-indicator, hypervolume (HV), and the inverted generational distance plus (IGD+), are calculated for each run. It is mandatory to have a reference point or a reference set to calculate the indicators, and this task is a challenging one [Ishibuchi et al., 2018], or it is sometimes provided by the user [Yang et al., 2019]. We use the maximum values amongst all algorithm solutions for HV. For additive ϵ-indicator and IGD+, the reference set is a joint Pareto front, which comes from the algorithm's results (it can be viewed in the last column of Figure 3.6).

The MIP-DoM quality measure is a binary indicator. Using MIP-DoM as a unary indicator is still feasible; a straightforward idea merely uses a joint Pareto (created from other algorithm solution sets) or the real Pareto front. The MIP-DoM is then calculated using the joint Pareto as a reference set for each problem. We use this approach to facilitate a comparison with some unary indicators, such as HV and IGD+.

Table 3.1 shows average values of three unary quality indicators, the additive ϵ-indicator, hypervolume (HV), and the inverted generational distance plus (IGD+). MIP-DoM has also been calculated. In this table, the algorithms are sorted taking into account the ascending order of MIP-DoM results. It can be noted from the table that DTLZ1 and DTLZ3 have the HV values inflated by the presence of the dominance resistant solutions [Li and Yao, 2019], which are non-dominated solutions with a poor value in one objective but with good values in others.

For the DTLZ1, the algorithms presenting the best HV, ϵ-indicator, and MIP-

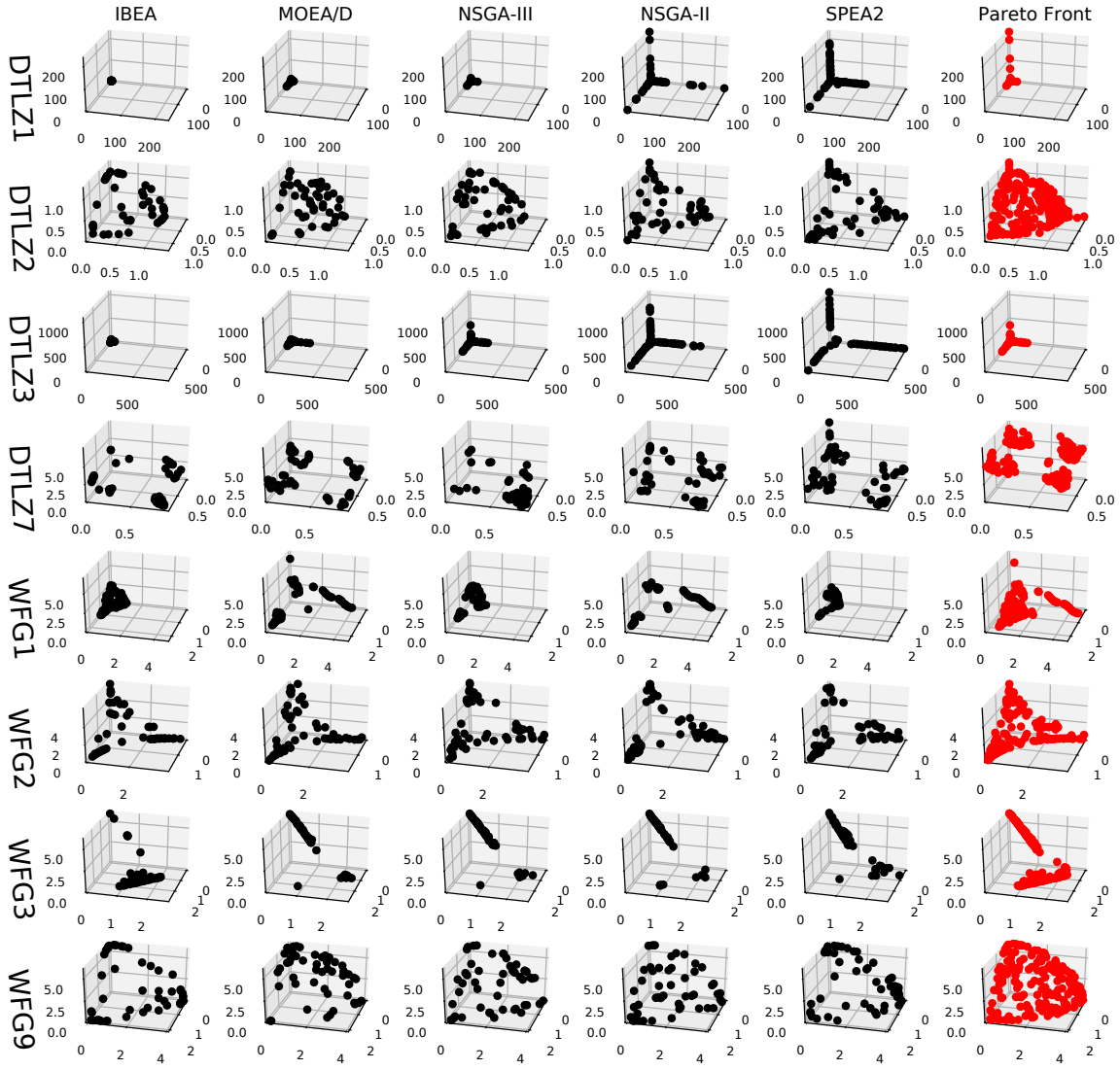

Figure 3.6. Solution sets with $|\boldsymbol{P}| = |\boldsymbol{Q}| = 50$ solutions and three objectives, $M = 3$, generated by IBEA, MOEA/D, NSGA-III, NSGA-II, and SPEA2 algorithms applied to DTLZ1, DTLZ2, DTLZ3, WFG1, WFG2, WFG3 and WFG9 problem sets. The Pareto front is generated by the collecting non-dominated solutions from all algorithms.

DoM values are IBEA and NSGA-III. For the HV indicator, the comparison is difficult due to inflated values. For IGD+, the results indicate NSGA-III and NSGA-II are the best algorithms. For MIP-DoM, the top two algorithms are IBEA and NSGA-III with 0.312 and 0.551, respectively. In the DTLZ2 problem, HV, IGD+, ϵ-indicator, and MIP-DoM indicate MOEA/D as one of the best results. For HV and IGD+, there is a tie between NSGA-III and MOEA/D. DTLZ3 results for ϵ-indicator, HV, and MIP-DoM classify IBEA and NSGA-III as the best algorithms. Again, considering HV, due to some extreme points, the values are next to each other. The best algorithms are

Table 3.1. MIP-DoM($\boldsymbol{P}$,$\boldsymbol{Q}$) values for the *DTLZ* and *WFG* families for comparison among IBEA, MOEA/D, NSGA-III, NSGA-II, and SPEA2 algorithms. It must be noted that $\boldsymbol{P}$ is the solution set generated by the algorithm and $\boldsymbol{Q}$ is the non-dominated solution set from all algorithm's results combined. HV, IGD+, and additive ϵ-indicator are also presented. The algorithms are sorted according to worse performance by MIP-DoM.

Problem	***Algorithms***	**MIP-DoM**	**HV**	**IGD+**	**additive ϵ-indicator**
DTLZ1	*IBEA*	0.312	0.999	0.089	0.001
	NSGA-III	0.551	0.999	0.032	0.001
	MOEA/D	1.630	0.999	0.052	0.009
	NSGA-II	6.422	0.998	0.041	0.027
	SPEA2	11.139	0.984	0.089	0.064
DTLZ2	*MOEA/D*	0.970	0.641	0.065	0.089
	IBEA	1.000	0.632	0.074	0.130
	NSGA-III	1.004	0.641	0.065	0.106
	NSGA-II	1.013	0.607	0.076	0.193
	SPEA2	1.021	0.595	0.082	0.162
DTLZ3	*IBEA*	0.0491	0.999	0.129	0.000
	NSGA-III	18.652	0.999	0.017	0.045
	MOEA/D	25.374	0.998	0.077	0.065
	NSGA-II	51.208	0.984	0.099	0.142
	SPEA2	150.870	0.706	0.342	0.369
DTLZ7	*NSGA-III*	1.402	0.198	0.085	0.175
	IBEA	1.468	0.210	0.064	0.082
	MOEA/D	1.695	0.196	0.073	0.123
	NSGA-II	1.782	0.181	0.071	0.167
	SPEA2	2.184	0.141	0.112	0.185
WFG1	*IBEA*	1.230	0.342	0.192	0.257
	MOEA/D	1.557	0.371	0.105	0.243
	SPEA2	1.659	0.256	0.289	0.425
	NSGA-III	1.822	0.286	0.248	0.326
	NSGA-II	1.944	0.343	0.134	0.272
WFG2	*IBEA*	1.503	0.864	0.054	0.110
	NSGA-III	1.571	0.824	0.062	0.201
	MOEA/D	1.683	0.834	0.057	0.134
	NSGA-II	1.687	0.790	0.075	0.184
	SPEA2	1.859	0.777	0.080	0.216
WFG3	*IBEA*	1.889	0.410	0.060	0.111
	NSGA-III	2.609	0.383	0.069	0.264
	NSGA-II	2.630	0.392	0.060	0.241
	MOEA/D	2.820	0.375	0.083	0.282
	SPEA2	3.178	0.373	0.079	0.211
WFG9	*IBEA*	1.965	0.312	0.085	0.156
	NSGA-III	2.194	0.324	0.074	0.095
	MOEA/D	2.331	0.290	0.089	0.161
	NSGA-II	2.601	0.304	0.074	0.123
	SPEA2	2.759	0.289	0.085	0.171

indicated as IBEA, NSGA-III, and MOEA/D (there is a full agreement between MIP-DoM and HV, considering algorithms' ranking). For IGD+, there is an indication of NSGA-III and MOEA/D as the best ones. Finally, for the DTLZ7 problem set, IBEA is pointed to as the best one for HV, IGD+, and ϵ-indicator. However, NSGA-III is

the best one, according to MIP-DoM, and the second one in HV.

The same experiment is done for the *WFG* family. For WFG1 in Table 3.1, the algorithm with the best value is MOEA/D for HV, IGD+, and ϵ-indicator. For MIP-DoM, the best algorithms are IBEA and MOEA/D. For WFG2, the best values for HV, IGD+, and ϵ-indicator and MIP-DoM indicate IBEA as one of the best algorithms. HV, IGD+, and MIP-DoM rank these three algorithms as the best ones: IBEA, MOEA/D, and NSGA-III. In the same way, for WFG3, IBEA generates the best solution sets for all indicators. Again, there is an agreement amongst them. For IGD+, the best algorithms are IBEA and NSGA-II, presenting a tie between these two algorithms. Finally, for the WFG9 problem set, considering HV, IGD+, and additive ϵ-indicator NSGA-II is the best algorithm. MIP-DoM points out IBEA and NSGA-III as the best ones. For IGD+, there is a tie between NSGA-III and NSGA-II.

3.4.2.2 Correlation with Existing Indicators

Our propose is to introduce the MIP-DoM as an alternate performance indicator. Next, we calculated the Pearson correlation coefficient of the MIP-Dom indicator with HV, IGD+, and additive ϵ-indicator. A Pearson correlation coefficient (ρ) is calculated in Table 3.2 to assess if there is a linear relation among the indicators. We measure the correlation coefficient using the problem sets individually, the problem set families, and the general correlation among all indicators. To assess the statistical significance of the results, a null Pearson correlation hypothesis test with a significance level of 0.05 is carried out. Values which are not statistically significance are indicated as * in Table 3.2. All values are normalized before the calculation of correlation coefficients.

We use the negative HV in the correlation for simplicity since it is the only indicator for which a larger value means better. It is observed that there is a reasonably strong correlation between MIP-DoM and HV, and with additive ϵ-indicator, taking into account all problems. Concerning the DTLZ family, the correlation between MIP-DoM and HV and additive ϵ-indicator is even stronger, 0.90 and 0.82, respectively. For the WFG family, the correlations between MIP-DoM and HV and additive ϵ-indicator are different from the DTLZ family. We argue that WFG1 is responsible for decreasing the correlation values since the correlation between MIP-DoM and HV is high for WFG2 and WFG3.

In general, the correlation between MIP-DoM and HV is high for most problems. This gives us the confidence of its use as an alternate performance indicator for set-based multi-objective optimization algorithms.

Table 3.2. The Pearson correlation coefficient (ρ) to assess the linear relationship between MIP-DoM and other indicators (HV, IGD+, and ϵ-indicator).

	MIP DoM correlation with		
Problem set	***-HV***	***IGD+***	***additive ϵ-indicator***
DTLZ1	0.896	0.347*	0.989
DTLZ2	0.770*	0.769*	0.783*
DTLZ3	0.964	0.887	0.998
DTLZ7	0.942	0.720*	0.502*
Combined DTLZ	**0.895**	**0.681**	**0.818**
WFG1	0.265*	0.019*	0.256*
WFG2	0.860	0.815*	0.626*
WFG3	0.945	0.747*	0.684*
WFG9	0.627*	0.093*	0.226*
Combined WFG	**0.674**	**0.372***	**0.448**
Combined All	**0.784**	**0.526**	**0.633**

* We applied a hypothesis test for the correlation coefficient with a significance level of 0.05. The * shows that the calculated p-value is over 0.05, and it is not possible to reject the null hypothesis that $\rho = 0$.

3.4.2.3 Many-Objective Problems

Next, the goal is to test if the MIP-DoM approach can be applied in many-objective scenarios. Inspired by the issues raised in the multi-objective experiments and the well-known quality indicators' weaknesses, our motivation to apply DoM in many-objective scenarios can be summed up as:

1. HV is hard to exactly calculate in many-objective problems [Guerreiro et al., 2021]. Yet, it is sensitive to extreme points and dominance resistant solutions, such as observed in the last experiment for DTLZ1 and DTLZ3 cases;

2. In many-objective problem sets and real cases, it is hard to obtain a Pareto front or reference set that is evenly distributed such as vital to IGD+, for example; The same happens to HV, in [Ishibuchi et al., 2018], for example, some experiments have shown that a slightly worse point than the nadir point is not always appropriate for problem sets comparison;

3. Considering the DoM definition and its calculation proposal, it is still relevant to note that it try to use all the information available in DoM definition. This is an essential feature to a quality indicator, mainly considering problem sets with high number of objective functions and cardinality.

In this sense, we would like to observe how the MIP-Dom model behaves in such scenarios, and analyze its characteristics. The same problem sets from the previous experiment are used, and two algorithms are chosen to compare and observe the quality indicator features. We choose MOEA/D and NSGA-III to generate the approximation sets. It is relevant to emphasize that any other algorithm could have been selected, however, the rationale behind this choice is due to specificity of these algorithms to deal with many-objective problems. The object of our study here is the quality indicator, and the aim is to assess how it behaves with different many-objective test cases.

In the same manner as the CEC'2018 competition [Cheng et al., 2018] we establish $L = 100$, 170, and 240 and $M = 3$, 5, 10, and 15, respectively.

Table 3.3. Using the solution sets generated by MOEA/D and NSGA-III algorithms, MIP-DoM value for the many-objective experiments for *DTLZ* and *WFG* families. The number of points in the final non-dominated set is set to $L = 100$, 170, and 240, and the number of objectives, $M = 5$, 10, and 15.

		Value		
Problem	***MIP-DoM(P, Q)***	*L=100*	*L=170*	*L=240*
		M=5	*M=10*	*M=15*
DTLZ1	*(MOEA/D, NSGA-III)*	1463.461	560.182	594.894
	(NSGA-III, MOEA/D)	3401.332	6315.371	9944.266
DTLZ2	*(MOEA/D, NSGA-III)*	3.943	1.175	1.065
	(NSGA-III, MOEA/D)	3.804	8.972	16.665
DTLZ3	*(MOEA/D, NSGA-III)*	2839.942	1878.871	2040.851
	(NSGA-III, MOEA/D)	6418.077	15532.120	23735.510
DTLZ7	*(MOEA/D, NSGA-III)*	2.502	5.543	4.745
	(NSGA-III, MOEA/D)	1.148	2.948	2.950
WFG1	*(MOEA/D, NSGA-III)*	0.270	0.279	0.124
	(NSGA-III, MOEA/D)	0.464	0.707	0.775
WFG2	*(MOEA/D, NSGA-III)*	2.001	1.179	1.137
	(NSGA-III, MOEA/D)	1.571	2.423	3.665
WFG3	*(MOEA/D, NSGA-III)*	4.524	4.678	1.955
	(NSGA-III, MOEA/D)	5.170	14.160	22.981
WFG9	*(MOEA/D, NSGA-III)*	3.183	2.177	0.524
	(NSGA-III, MOEA/D)	4.450	13.583	31.296

In Table 3.3 the MIP-DoM values are presented. The algorithm solution sets are compared in pairs and both directions. For DTLZ1 with $L = 100$ and $M = 5$, the MIP-DoM(MOEA/D, NSGA-III) = 1463.461, and the MIP-DoM(NSGA-III, MOEA/D) = 3401.332, indicating that the approximation set generated by MOEA/D is better than that by NSGA-III. For DTLZ2, while MIP-DoM(NSGA-III, MOEA/D) is smaller than MIP-DoM(MOEA/D, NSGA-III) with $M = 5$, but with 10 and 15

objectives, MOEA/D's performance is better. From the table, we observe that MIP-DoM has an agreement about the best algorithm for some problem sets, independent of the parameter M and L. DTLZ1, DTLZ3, WFG1, WFG3, and WFG9, for example, MOEA/D set is better than that in NSGA-III. On the other hand, for DTLZ7, NSGA-III provides lower values. There is a disagreement found in DTLZ2 and WFG2 for $L = 100$ and $M = 5$. Furthermore, for the DTLZ2, for example, the values are similar for both algorithms. However, our goal here is to show that it is possible to use and calculate MIP-DoM regardless of the number of points (L) in the solution set and the number of objective functions (M, and compare two or more EMaO algorithms.

On the other side, the maximum number is 9, 8, and 26 in the experiments for $L = 100$ and $M = 5$, $L = 170$ and $M = 10$, and $L = 240$ and $M = 15$, respectively. This fact denotes that the MIP-DoM value is more suitable for many-objective scenarios.

3.5 Further Studies with MIP-DoM

Having compared the MIP-DoM indicator with popular existing indicators on multi- and many-objective optimization problems, we now analyze and reveal several other properties of the MIP-DoM quality indicator.

3.5.1 Computational Time

Every MIP-DoM computation requires solving a mixed-integer linear programming problem stated in Equations (3.3) to (3.10). In cases in which most points of $\boldsymbol{Q}$ are dominated by some point in $\boldsymbol{P}$ exist, the number of required MIP iterations is small and the computational time to solve the MIP problem is also small. However, in our experiment, the worst-case scenario spends approximately ~ 208 hours, for example, using MOEA/D to solve WFG1 with $L = 240$ and $M = 15$. In this case, the pre-processing method is not feasible, and the number of simplex iterations is very large, as well as the number of explored nodes.

The time spent in seconds by the solver is an important aspect to analyze. Figure 3.7 presents an experiment using DTLZ1 problem, as an example. We generate 50 random approximation set as $\boldsymbol{P}$ and the Pareto front as $\boldsymbol{Q}$ to calculate MIP-DoM($\boldsymbol{P}$,$\boldsymbol{Q}$) (the distance of movement which a random approximation set needs to dominate the Pareto front). Our intention is to assess how the computational time increases with an increase in number of population size for 5, 10, and 15 objective problems. In Figure 3.7, the y axis is in log scale, and the x axis is represented with $L = 50$, 100, 200, 300 and 400 solutions.

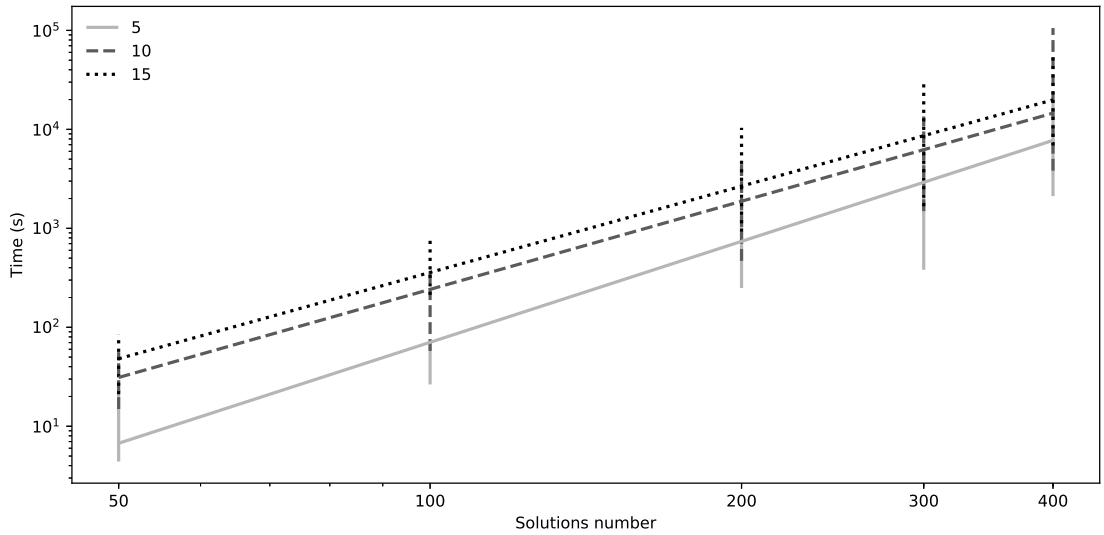

Figure 3.7. A polynomial increase in computational time with population size. Fifty experiments are done with $L = 50$, 100, 200, 300 and 400 solutions using the DTLZ1 problem. The data set is randomly generated, and the objective function is analytically calculated.

In Figure 3.7, considering M = 5, 10 and 15, we can observe a polynomial time behavior for DTLZ1 considering the solutions number ($\approx O(L^{3.311})$, $O(L^{2.888})$ and $O(L^{2.833})$, respectively).

Another aspect to verify is how the time behaves if M is increased for many-objective problems. In Figure 3.8, using the DTLZ1, values of M = 7, 10, 12, 15, 20, 25 and 30 are adopted and we generate random solutions, and calculate its objective functions using its analytical function. The Pareto front is generated, and our experiments measure the time spent to calculate *MIP-DoM(**P**, **Q**)*, with random approximation set as $\boldsymbol{P}$, and the Pareto front as $\boldsymbol{Q}$. For each M, fifty experiments are run with $L = 200$ solutions in the approximation set. In Figure 3.8, both axes are in log scale, representing a polynomial time behavior ($\approx O(M^{0.686})$. Each box-plot graph is a test set with its associated M. The figure reveals that the MIP-DoM indicator can be useful for large number of objectives, as it demands a small increase in computational time for an unit increase in the number of objectives.

A burdensome scenario for MIP-DoM is when there is no a priori dominance between the points in each solution set, and the density of the two solution sets used are high. In areas with high density, there are many options among $\boldsymbol{p}_i$ trying to dominate some $\mathcal{Q}_s$. The branch-and-bound dynamic becomes more challenging in these situations, combining the $\hat{\boldsymbol{p}}$ possibilities, between its lower and upper bound, and the binary variables. This fact must be better investigated and exploited in a future work.

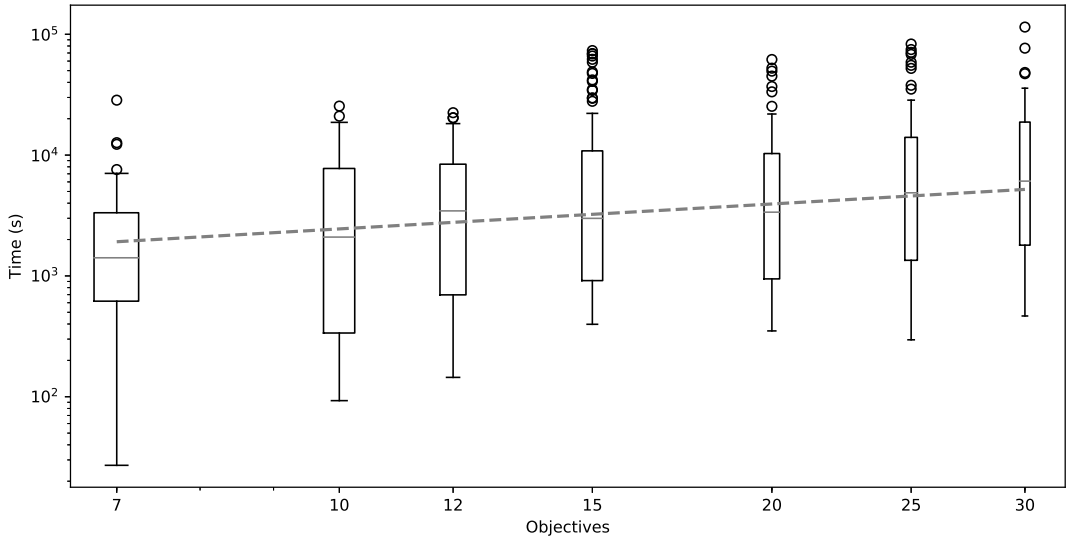

Figure 3.8. A boxplot with the time spent to solve the MIP-DoM model as a function of number of objectives. For each box-plot, fifty experiments are done with $L = 200$ using the DTLZ1 problem set definition. The data set is randomly generated, and the objective function is analytically calculated.

3.5.2 MIP-DoM as a Running Performance Indicator

Popular performance metrics (such as, GD, IGD, IGD+, and also HV in some sense) require the knowledge of true Pareto front so that a reference set can be obtained. This severely restrict their usage in arbitrary problems. But, a recently proposed running performance metric method [Blank and Deb, 2020b] can be used for handling an arbitrary problem. At a generation t, all non-dominated (ND) solutions from all past generations from the initial population is first collected and the resulting ND set is used as the reference set for metric computation. Then, at generation $t + \Delta t$, the whole process can be repeated (Δt can be suitably chosen) and the time-varying plot can be made to indicate how the metric changes with generation.

In Blank and Deb [2020b], IGD+ is used as an example of the running quality indicator. The IGD+ indicator demands a reference set, such as the true Pareto front. We show the use of running metric for two purposes: (i) termination criteria in a pair algorithm comparison and (ii) a detailed algorithm behavior view in the quality indicator perspective. Our goal is to test if MIP-DoM can be used as a running quality indicator, and what are their characteristics. Figure 3.9 presents six curves. We calculate each one using the solution set from each generation as $\boldsymbol{P}$ and, at the generation 10, 20, 30, 40, and 50, solution sets from its generation as $\boldsymbol{Q}$ in the MIP-DoM($\boldsymbol{P}$, $\boldsymbol{Q}$) calculation.The final solid line is computed using the known Pareto front solutions, to demonstrate the running indicator's behavior.

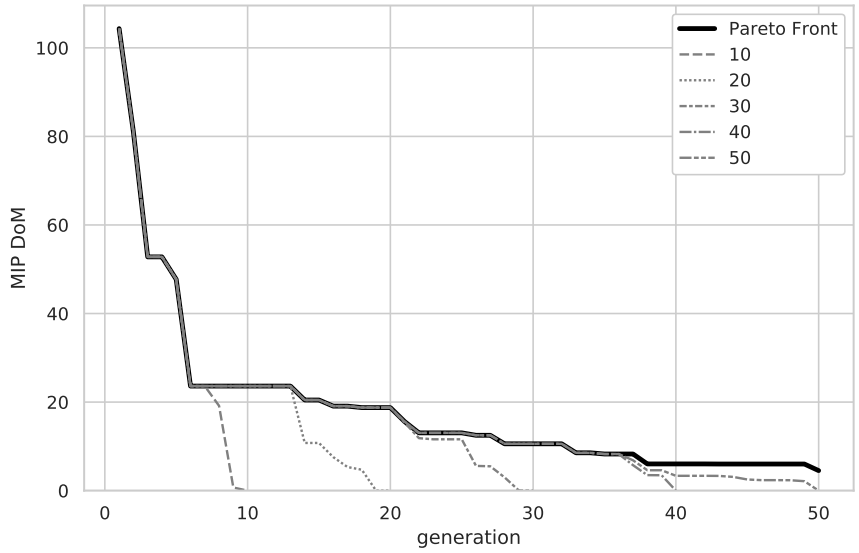

Figure 3.9. MIP-DoM as a running performance quality indicator. The data refers to the DTLZ1 problem set, with two objectives using NSGA-II algorithm with 50 generations. We calculate MIP-DoM using $\boldsymbol{P}$ as each generation solution set and $\boldsymbol{Q}$ as solution sets at generations 10, 20, 30, 40, 50 and the final Pareto front.

Figure 3.9 can be analysed from different manners. First, there is a monotonic decrease in each MIP-DoM curve, indicating that MIP-DoM can correctly indicate the monotonic improvement of IGD+ with generations. It is not surprising that each intermediate running indicator curve drops to zero at the pivot generation at which the indicator was calculated. Since the population at this pivot generation is expected to be best compared to earlier generation populations, MIP-DoM($\boldsymbol{P_t}$, $\boldsymbol{P_t}$) = 0. Second, consider the two consecutive running indicator curves (say at generations 30 and 40). The DoM value calculated at generation 30 using the 40-th generation ND set is higher than that calculated using the 30-th generation ND set. This indicates that the 40-th generation ND set is better than the 30-th generation ND set. Third, notice that for up to about 35 generations, DoM calculated using the Pareto front is almost equal to DoM calculated using ND set at generation 50. By checking with a threshold value of MIP-DoM, a termination of an EMO/EMaO run can be adopted.

Another use of the running MIP-DoM indicator can come in comparing two or more EMO/EMaO algorithms generation by generation. In Figure 3.10, a comparison between NSGA-II and NSGA-III is made. We calculate MIP-DoM(NSGA-II, combined solutions) in a dashed line and MIP-DoM(NSGA-III, combined solutions) in a solid line. The combined solutions represent the non-dominated solutions from NSGA-II and NSGA-III results at generation 50. We use the DTLZ1 with two-objective functions and 30 solutions. Each point in each algorithm curve indicates the MIP-DoM value

at the specific generation calculated using the combined solutions. We can observe a similar monotonic drop behavior. In the beginning, until generation 4, NSGA-II presents a better value than NSGA-III. From generation 8 onward, there is a change, and NSGA-III is better than NSGA-II.

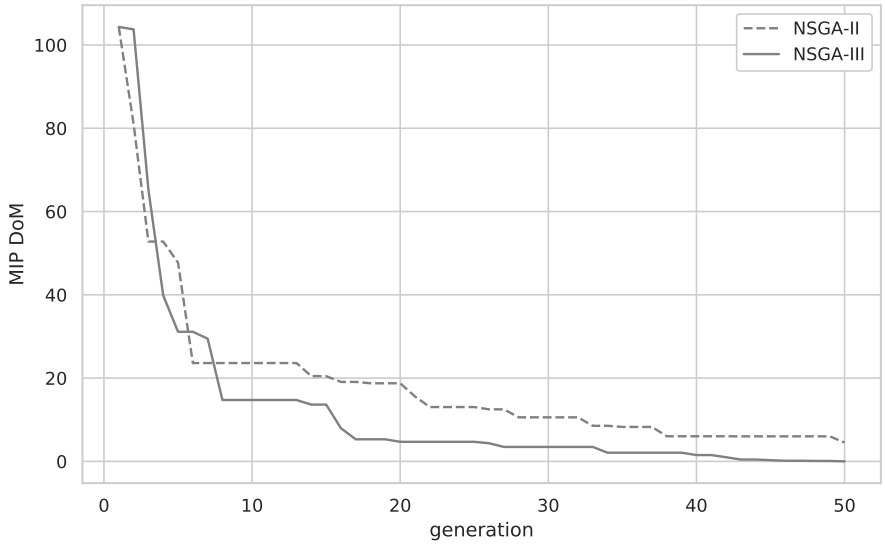

Figure 3.10. An algorithm comparison between NSGA-II and NSGA-III using the MIP-DoM running quality indicator. The data refers to DTLZ1 problem set, with two objective functions and 30 solutions in each set. We calculate MIP-DoM using $\boldsymbol{P}$ as each generation solution set and $\boldsymbol{Q}$ as a joint solution set(non-dominant solutions) from the NSGA-II and NSGA-II 50 generation.

These plots indicate how the MIP-DoM can be employed as a running quality indicator, as termination criteria and as a comparison indicator without the knowledge of the true Pareto solutions.

3.5.3 Parametric Study with Gap in Computing MIP-DoM

In the branch-and-bound formulation, some information can be obtained using the function value of the linear programming relaxation and the function value of the integer program. A valid upper bound on the optimal solution is a proper integer solution with the best value found in the branch-and-bound procedure in a minimization problem. On the opposite side, we also can have a valid lower bound during the branch-and-bound. It comes from the linear programming relaxation, taking the minimum of all current leaf nodes' optimal objective values. The difference between the upper and lower bounds is known as *gap*. In general, when the $gap = 0$, we have reached optimality.

The question raised is whether MIP-DoM is affected by a gap variation, such as $gap = 5\%$ or 10%? To investigate this issue, we repeat the same experiment in the multi-objective section, in which the Pearson correlation coefficient between MIP-DoM and some indicators are calculated. The idea is to observe how the gap will influence the MIP-DoM results and how a change affects the correlation coefficient.

We vary the gap parameter, such as posed in our initial question. In Table 3.4, not only the original results are presented, but also the results with $gap = 5\%$ and 10% are added. We can observe that the gap parameter introduces a small perturbation in the Pearson correlation coefficient. For example, the HV has a change in the correlation coefficient of less than one percent comparing the values at $gap = 0\%$ and 10%. Similar behavior happens to IGD+ and additive ϵ-indicator.

On the other hand, when we analyze the effect in the model's time, we can observe a clear impact in all cases. In Table 3.4, the mean and standard deviation (std deviation) of times for each problem set are presented. The gap has good impacts on the time spent. In some cases, the difference lies in one order of magnitude.

Additionally, there are differences in the time spent amongst the problem sets. At this experiment, it must be noted that the reference set has different sizes (number of solutions in the joint Pareto), and this influences the time spent by the model, as discussed before. For DTLZ1, DTLZ2, DTLZ3 and DTLZ7 are 79, 207, 93 and 154 solutions, respectively. Considering WFG1, WFG2, WFG3, and WFG9, the values are 104, 124, 151, and 179. Even considering just one problem set, the times spent by MIP-DoM still can show relevant variations among the algorithms. In our experiments IBEA and NSGAIII present the highest times, considering the DTLZ2 and WFG3.

The gap information can help in the binary quality indicator context. If one is interested in a provably optimal solution, it is necessary to wait until the gap reaches zero. Usually, in the quality indicator context, optimality is not an issue, and "$\boldsymbol{P}$ is better than $\boldsymbol{Q}$" is the only question to be answered. Moreover, a faster answer would be better. Motivated by these issues, different gaps are analyzed and the time spent to decide whether "$\boldsymbol{P}$ is better than $\boldsymbol{Q}$" is computed in each case. Considering Table 3.3 in the many-objective experiments, some high times are also found, for example, DTLZ7 and WFG1 for MOEA/D with $L = 240$ and $M = 15$. For these cases, and in the context of a binary quality indicator, it is not mandatory to obtain an optimal solution, providing the statement is valid. Taking, for example, DTLZ7 for NSGA-III and MOEA/D with $L = 240$ and $M = 15$, it took $\sim 1,818$ seconds to calculate MIP-DoM(NSGA-III, MOEA/D) = 4.475. Similarly, the solver took $\sim 198,795$ seconds to calculate MIP-DoM(MOEA/D, NSGA-III) = 2.950. However, the solver presented a MIP-DoM valid interval search between 6.188 and 3.101, with $gap = 50\%$ at 98,294

Table 3.4. [Pearson correlation coefficient between MIP-DoM and other indicators. The coefficient is re-calculated with the new MiP-DoM value considering $gap = 5\%$ and 10%]The first part of the table presents the Pearson correlation coefficient between MIP-DoM and other indicators. The coefficient is re-calculated with the new MiP-DoM value considering $gap = 5\%$ and 10%. The second part of the table focuses on the time spent by the model for the given gap. We show the mean and standard deviation for each problem set and gap.

Problems	***MIP DoM*** correlation with									**Time spent** (seconds)					
	-HV			***IGD+***			***additive ϵ-indicator***			mean			std deviation		
	gap			*gap*			*gap*			*gap*			*gap*		
	0%[1]	5%	10%	0%[1]	5%	10%	0%[1]	5%	10%	0%[1]	5%	10%	0%[1]	5%	10%
DTLZ1	0.896	0.896	0.896	0.347*	0.347*	0.347*	0.989	0.989	0.989	2.4e+01	2.5e+01	1.4e+01	1.2e+01	2.9e+01	1.2e+01
DTLZ2	0.770*	0.773*	0.770*	0.769*	0.763*	0.769*	0.783*	0.784*	0.783*	2.8e+05	2.0e+05	1.3e+05	3.9e+05	3.3e+05	2.6e+05
DTLZ3	0.964	0.964	0.963	0.887	0.887	0.886	0.998	0.998	0.998	3.8e+01	2.8e+01	1.6e+01	4.7e+01	2.9e+01	1.8e+01
DTLZ7	0.942	0.942	0.942	0.720*	0.720*	0.720*	0.502*	0.502*	0.502*	1.6e+04	3.1e+03	9.9e+02	2.8e+04	3.3e+03	8.3e+02
DTLZ	**0.895**	**0.894**	**0.895**	**0.681**	**0.679**	**0.681**	**0.818**	**0.818**	**0.818**	**7.5e+04**	**5.1e+04**	**3.3e+04**	**2.2e+05**	**1.7e+05**	**1.3e+05**
WFG1	0.265*	0.250*	0.242*	0.019*	0.003*	0.002*	0.256*	0.242*	0.224*	3.3e+02	1.6e+02	8.7e+01	3.6e+02	1.2e+02	6.1e+01
WFG2	0.860	0.877*	0.877*	0.815*	0.833*	0.833*	0.626*	0.631*	0.631*	3.6e+02	2.0e+02	1.2e+02	3.0e+02	1.3e+02	7.4e+01
WFG3	0.945	0.945	0.946	0.747*	0.747*	0.747*	0.684*	0.684*	0.684*	1.6e+04	2.2e+03	7.4e+02	1.7e+04	1.3e+03	2.8e+02
WFG9	0.627*	0.627*	0.649*	0.093*	0.093*	0.009*	0.226*	0.226*	0.333*	9.9e+02	4.7e+02	2.4e+02	1.1e+03	3.4e+02	1.2e+02
WFG	**0.674**	**0.675**	**0.678**	**0.372***	**0.372***	**0.398***	**0.448**	**0.446**	**0.468**	**4.4e+03**	**7.5e+02**	**3.0e+02**	**1.1e+04**	**1.1e+03**	**3.1e+02**
All	**0.784**	**0.784**	**0.786**	**0.526**	**0.526**	**0.539**	**0.633**	**0.632**	**0.643**	**4.0e+04**	**2.6e+04**	**1.7e+04**	**1.6e+05**	**1.2e+05**	**9.2e+04**

[1] In fact, the 0% means a $gap = 10^{-6}\%$.

* We use a hypothesis test for the correlation coefficient with a significance level of 0.05. The * shows that the calculated p-value is over 0.05, and it is not possible to reject the null hypothesis that $\rho = 0$.

seconds.

3.6 Final observations and notes

The DoM binary quality indicator considers the minimum move of one set to dominate the other set, meaning that every element of the second set is either dominated or identical to at least one element of the first set. The indicator is Pareto compliant and does not demand any parameters or reference sets.

We have presented that MIP-DoM also brings some other advantages as a binary quality indicator. For example, it appeared to have a highly Pearson correlation with HV and also showed a monotonic decrease in its value when the first set ($\boldsymbol{Q}$) was fixed and the second set ($\boldsymbol{P}$) approaches towards the Pareto front.

We have done initial experiments showing that our approach was corrected compared to the original proposal on artificial bi-objective examples. The other results have also made the sensitivity of the DoM indicator on dominance, diversity, cardinality, and other issues related to two non-dominated set comparisons. Furthermore, experiments using multi- and many-objective scenarios showed that the formulation could be used in such cases. We have also presented how the MIP gap information can be used to improve the time spent by the model with a slight change to the DoM value. Finally, we have presented additional use of MIP-DoM as a running performance indicator.

We have proposed the mathematical programming approach to calculate performance quality indicators, and we have provided a detailed MIP formulation of the DoM calculation indicating how the number of variables increases with the number of objectives and population size. The handicap of DoM was its requirement to solve a mixed-integer linear programming problem, making it apparently time-consuming to be applied to problems with many objectives and large population sizes.

Chapter 4

MIP-DoM: a new compact formulation

4.1 Introduction

One MIP-DoM's drawback is related to its computational time. In the last Chapter, some reasons were investigated and discussed. In this Chapter, we are starting to deal with the computational time model issues. As previously discussed, it depends on, for example, I) the number of solutions and II) the number of objectives. Additionally, the model's time is more impacted by the number of solutions and even presenting evidence about a polynomial relation between them; in some cases, the model calculation takes so many hours.

Another issue is related to the model's complexity and interpretability. It comes with the perspective that to solve DoM, we use an assignment model approach. It is a way to solve the question; however, it brings additional complexity to DoM modeling. The number of variables and constraints is significant, affecting the time search for optimal solutions. Therefore, a question emerges in a model critical assessment: Is this possible to model MIP-DoM most straightforwardly?

This Chapter presents a new compact formulation for MIP-DoM, initially proposed by Prof. Dr. Carlos Fonseca. First, a discussion about DoM models' characteristics will be briefly treated. Next, the compact model is introduced and discussed. Subsequently, some experiments comparing the results from the two models are shown, emphasizing how fast the compact one can be in dealing with increments in the number of solutions and objectives. Finally, we investigate some limits of this model using some large problem set instances.

4.2 DoM model's approach

The first idea presenting DoM is related to PCI [Li et al., 2015b]. The question related to DoM calculation is associated with the number of possibilities to find $\boldsymbol{P'}$, which is numerous. Therefore, any combination of some $\boldsymbol{P'}$ must be tested, and some of them can dominate $\boldsymbol{Q}$, regarding Equation (3.1).

A second approach to calculated DoM is based on the observation that the problem is, in fact, a particular case of an assignment problem [d. V. Lopes et al., 2020]. Unfortunately, this approach cannot scale and solve models with more than 20 solutions in each set.

In the last Chapter, the DoM calculation as a mixed-integer linear programming model was presented. But, it is still an extension of the assignment perspective. The MIP model calculates the distance $d(\boldsymbol{p}_i, \boldsymbol{p}'_i)$ inside the model, unlike the model presented in d. V. Lopes et al. [2020]. A new auxiliary variable is introduced, $\hat{\boldsymbol{p}}_i$, and there is a strict relation between $\hat{\boldsymbol{p}}_i$ and $\boldsymbol{p}'_i$. At the end, the MIP-DoM model calculates the final $\boldsymbol{p}'_i$ using $\hat{\boldsymbol{p}}_i$. This point introduces an additional complexity to the model, but allow that the $d(\boldsymbol{p}_i, \boldsymbol{p}'_i)$ can be calculated as an assignment approach.

The MIP model contains continuous variables that represents each objective solution in $\hat{\boldsymbol{p}}_i$. Additionally, there is binary variables that indicates if each $\boldsymbol{p}_i$ is active and generates a $\hat{\boldsymbol{p}}_i$ to dominate some $\boldsymbol{q}$. Additional constraints are used on binary and continuous variables to solve the final problem.

Given the model size and complexity, there is no guarantee that the problems can be solved optimally in a non-prohibitive computational time. Despite all the new ideas brought by the last Chapter, some hard instances show that the computational time and model's complexity are open issues.

4.3 MIP-DoM new compact formulation

Based on a model assessment proposed by Prof. Dr. Carlos Fonseca, this Chapter brings another model to calculate DoM. This model does not come from an assignment abstraction but is based on the inner DoM characteristics to calculate the $d(\boldsymbol{p}_i, \boldsymbol{p}'_i)$ as a Manhattan distance.

Two main variables are used: $z_{(i,m)}$ and $x_{(i,j)}$, in which $i \in \{1, \ldots, |\boldsymbol{P}|\}$, $m \in \{1, \ldots, M\}$, and $j \in \{1, \ldots, |\boldsymbol{Q}|\}$. The first variable represents the extent of movement that must be done using $p_{(i,m)}$ as a starting point to dominate some $q_{(j,m)}$. In this

sense, $z_{(i,m)}$ is a continuous variable. $x_{(i,j)}$ is a binary variable and it controls whether $\boldsymbol{p}_i$ dominates $\boldsymbol{q}_j$ or not. It will assume 1 if $\boldsymbol{p}'_i$ dominates $\boldsymbol{q}_j$, and 0 otherwise.

In this compact model there is no explicitly declaration of $\boldsymbol{p}'_{i,m}$ variable. However, it is implicitly used in the model. In fact, the expression $p'_{(i,m)} = p_{(i,m)} - z_{(i,m)}$ is valid, but not directly used in the model (there is no $p'_{(i,m)}$ continuous variable in the final model).

The model is presented in Equation (4.1) to (4.5).

$$\text{minimize} \quad \sum_{i=1}^{|\boldsymbol{P}|} \sum_{m=1}^{M} z_{(i,m)} \tag{4.1}$$

subject to:

$$z_{(i,m)} \geq 0, \qquad \forall i, \forall m \tag{4.2}$$

$$\Big\{ p_{(i,m)} - z_{(i,m)} \leq q_{(j,m)}) + (1 - x_{(i,j)})\mathbb{U} \qquad \forall i, \forall j, \forall m \tag{4.3}$$

$$\sum_{i=1}^{|\boldsymbol{P}|} x_{(i,j)} \geq 1, \qquad \forall j \tag{4.4}$$

$$\begin{array}{l} x_{(i,j)} \in \{0,1\}, \\ z_{(i,m)} \in \mathbb{R}, \end{array} \quad . \qquad \forall i, \forall j, \forall m \tag{4.5}$$

where, $i = 1, \ldots, |\boldsymbol{P}|, j = 1, \ldots, |\boldsymbol{Q}|, m = 1, \ldots, M$.

The objective function is shown in Equation (4.1). The goal is to minimize the 'effort' that one or more solutions from $\boldsymbol{P}$ must do to dominate all elements in $\boldsymbol{Q}$. The $z_{(i,m)}$ variable represents this move from the i-th element in $\boldsymbol{P}$ on the m-th objective.

Constraint (4.2) indicates that this movement must be positive, because this value will be decreased from each value found in $\boldsymbol{p}_{i,m}$. Constraints in (4.3) indicate the use of $p'_{(i,m)}$ obtained from the difference between $p_{i,m}$ and $z_{(i,m)}$. It must be less than $\boldsymbol{q}_{j,m}$ to assure that $\boldsymbol{p}'_{(i)}$ will dominate $\boldsymbol{q}_j$. In this sense, we have the binary variable $x_{(i,j)}$ that assumes one when $\boldsymbol{p}'_{(i)}$ dominates $\boldsymbol{q}_j$, and zero otherwise. One last detail is about the $\mathbb{U}$ that is a parameter assuming an arbitrary great value.

In 4.4, we guarantee that every $\boldsymbol{q}_j$ must be dominated by at least one $\boldsymbol{p}'_{(i)}$. Finally, in 4.5, we just emphasize the continuous and binary model variables.

As comparison with the former model discussed in last chapter, the Equations 4.6 to 4.8 present the number of variables and constraints of this compact formulation:

$$\#\text{ continuous variables} = M(|\boldsymbol{P}|), \tag{4.6}$$

$$\#\text{ binary variables} = (|\boldsymbol{P}|)(|\boldsymbol{Q}|), \tag{4.7}$$

$$\#\text{ constraints} = M(|\boldsymbol{P}|)(1 + |\boldsymbol{Q}|) + |\boldsymbol{Q}|. \tag{4.8}$$

Taking again the example which has $M = 5$, $|\boldsymbol{P}| = |\boldsymbol{Q}| = 200$, there are 1,000 continuous variables, 40,000 binary variables, and 201,200 constraints in this compact formulation. Compared with its former formulation there is a significant reduction, mainly in terms of the number of continuous variables.

4.4 Computational experiments

The new formulation test is the main issue. Therefore, the same experiments proposed in Chapter 3 have been executed in a way to check the model's results. Initially, some problems sets in the multi-objective scenario were tested. These problems involve the same instances from DTLZ and WFG problem families. The results found are all equal to the former model, considering the final objective value function, or the MIP-DoM value.

Considering the solution set $\boldsymbol{P'}$ generated as the optimal model solution, the former and the new compact formulation present some differences. There are some problems in which the result set is equal, but in others, the models produce different result sets. This comes from the fact that some problems can have multiple optimal solutions. Since the models are different, they treat DoM calculation from different perspectives.

4.4.1 Many-objective problems

Our next test involves many-objective problems. Additionally, we want to verify how the compact formulation tackles the former model's computational time issues. In this sense, we reproduce the same experiment that describes a test case that takes almost 208 hours to be solved.

The experiment proposes a comparison between MOEA/D and NSGA-III (i.e., just as an example), as in Chapter 3. The problem set parameters are set to $L = 100$, 170, and 240 and $M = 3$, 5, 10, and 15, respectively, for many-objective test problems from DTLZ and WFG families. We use the same solver, GUROBI Gurobi Optimization [2019], installed in a workstation with 24 virtual processors and 96 Gb of RAM.

Table 4.1. Solution sets are generated by MOEA/D and NSGA-III algorithms. MIP-DoM value and the time spent in seconds are presented, for the many-objective experiments for *DTLZ* and *WFG* families. The number of points in the final non-dominated set is set to L = 100, 170, and 240, and the number of objectives, M = 5, 10, and 15.

Problem	***MIP-DOM(P,Q)***	**Value**			***Time Spent*** (seconds)					
					Former formulation			**Compact formulation**		
		L=100 M=5	*L=170 M=10*	*L=240 M=15*	*L=100 M=5*	*L=170 M=10*	*L=240 M=15*	*L=100 M=5*	*L=170 M=10*	*L=240 M=15*
DTLZ1	(MOEA/D, NSGA-III)	1498.491	854.421	953.162	206.79	12889.29	418881.64	1.98	25.78	193.73
	(NSGA-III, MOEA/D)	3351.770	6021.131	9688.700	352.02	1967.04	33323.49	3.19	19.75	117.08
DTLZ2	(MOEA/D, NSGA-III)	3.952	1.176	1.065	382.39	240.65	563.79	4.25	4.03	2.97
	(NSGA-III, MOEA/D)	3.795	8.971	16.670	123.86	3150.55	24484.10	8.08	19.25	53.47
DTLZ3	(MOEA/D, NSGA-III)	2902.071	1879.789	2037.103	204.13	427.43	2141.41	2.67	2.92	10.16
	(NSGA-III, MOEA/D)	6377.133	15381.813	23720.072	253.17	3273.67	25957.61	4.46	36.86	778.44
DTLZ7	(MOEA/D, NSGA-III)	6.850	5.169	9.880	545.03	26897.59	42416.19	7.01	172.51	134.44
	(NSGA-III, MOEA/D)	0.000	23.154	58.337	3.14	3499.59	14684.85	0.04	6.02	7.83
WFG1	(MOEA/D, NSGA-III)	0.493	0.052	0.000	233.02	426.71	15.40	3.30	2.60	0.20
	(NSGA-III, MOEA/D)	0.256	0.996	1.764	304.56	10007.41	22170.58	1.83	29.62	120.10
WFG2	(MOEA/D, NSGA-III)	2.255	0.970	0.089	421.62	1800.84	280.47	3.54	15.54	6.21
	(NSGA-III, MOEA/D)	1.327	2.657	5.115	141.63	9026.51	89008.71	0.97	20.68	52.62
WFG3	(MOEA/D, NSGA-III)	4.838	9.519	12.310	8837.06	12837.96	59110.23	69.94	45.34	439.47
	(NSGA-III, MOEA/D)	5.168	10.233	18.568	2561.97	4491.16	23590.15	26.31	14.80	76.82
WFG9	(MOEA/D, NSGA-III)	3.793	2.456	3.799	1057.01	2642.87	43607.81	3.97	12.87	39.71
	(NSGA-III, MOEA/D)	4.020	13.605	31.157	810.48	9935.47	23662.07	5.16	62.47	42.98

In Table 4.1 the results are presented. The DoM value and the time spent by the two formulations are shown. Considering the former formulation, one case related to DTLZ1 takes almost 117 hours to complete. The same problem using the compact formulation takes 194 seconds. There is no case where the compact formulation is worse than the former. In general, based on the experiment, the behavior of the compact formulation is a few hundred times better than the former model.

4.4.2 Computational time

MIP-DoM computation demands a mixed-integer linear programming instance execution. Considering the two formulations, we would like to compare how these formulations behave related to a number of objectives and solutions. A formulation that can use a small number of explored nodes, simplex iterations, or, in summary, a short time computation is desirable. In this sense, an experiment comparing the two formulations can emphasize how the formulations are different using computational time.

The time spent in seconds by the solver is the aspect to assess in our computational time study. Figure 4.1 presents an experiment using DTLZ1 problem, as an example. We use 50 random solution sets as $\boldsymbol{P}$, and the Pareto front $\boldsymbol{Q}$ is analytically

calculated. The formulation as MIP-DoM($\boldsymbol{P}$, $\boldsymbol{Q}$) is computed using these solution sets. The question is to analyze how the formulations behave: an increment in population size impacts the computational time, considering 5, 10, and 15 objective problems. In Figure 4.1 a *Log-log* plot is presented, the y axis is the time spent by the model, and the x axis is represented with $L = 50$, 100, 200, 300 and 400 solutions.

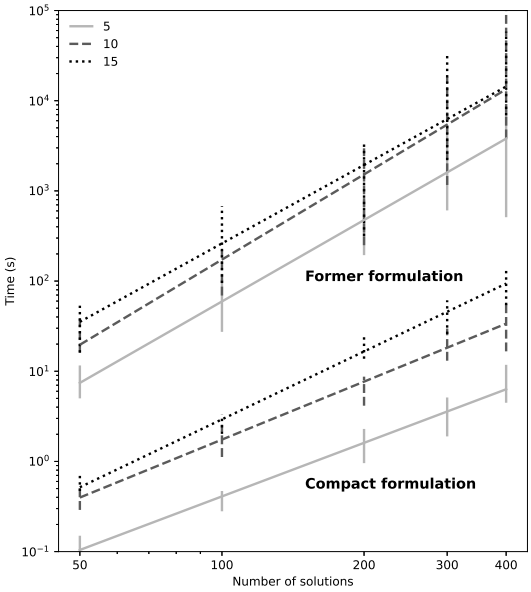

Figure 4.1. Considering the former and compact formulation, an increase in computational time with population size. Fifty experiments are done with $L = 50$, 100, 200, 300, and 400 solutions using the DTLZ1 problem.

In Figure 4.1, considering $M = 5$, 10 and 15, we can observe a polynomial-time behavior for DTLZ1 in both formulations. Former model with $O(L^{2.999})$, $O(L^{3.137})$, and $O(L^{2.888})$, respectively considering M, and the compact model is $O(L^{1.972})$, $O(L^{2.134})$, and $O(L^{2.500})$. One drawback of the former formulation is the increment in the number of solutions. The compact formulation alleviates this limitation. In Figure 4.1, independently of the number of objectives, it is possible to see a significant decrease in the solver time spent. The compact formulation general behavior is still polynomial, but the reduction magnitude is relevant, mainly considering $M = 5$ and 10. In practical situations, this behavior is substantial, and the observed reduction facilitates the use of DoM.

Another computational time perspective proposed in Chapter 3 is related to the model's behavior if M is increased for many-objective problems. In Figure 4.2, the DTLZ1 problem set is used again, but in this case we vary values of $M = 7, 10, 12, 15, 20, 25$ and 30 generating random solutions as $\boldsymbol{P}$, and being $\boldsymbol{Q}$ ana-

lytically calculated. As the same as before, our experiments measure the time spent to calculate MIP-DoM($\boldsymbol{P}$,$\boldsymbol{Q}$), considering the former and compact formulation. For each M , 50 experiments is executed with $L = 200$ solutions in the set. In Figure 4.2, both axes are in log scale, and each box-plot graph is a test set with its associated M.

Figure 4.2 shows some interesting issues. Again, there is an improvement in the time spent comparing the former and the compact formulation. The time spent is $O(L^{0.768})$, indicating that the former model formulation brings a linear relation to time spent and the number of objectives. The compact formulation slope is more inclined, and the complexity is $O(L^{2.012})$, pointing out that there is polynomial relation between the time spent and M.

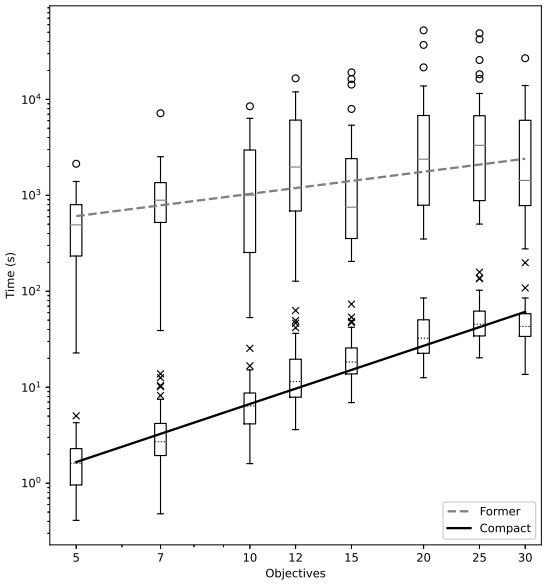

Figure 4.2. A time spent boxplot showing the results from the former and compact formulation. For each boxplot, fifty experiments are done with $L = 200$ using the DTLZ1 problem set definition.

4.4.3 Possible limits for the compact formulation

The compact formulation alleviates the time spent to calculate DoM. However, we would like to explore more practical model limits and mainly advise a user about these limitations. In such a way, we want to test significant cases and observe how the compact model will perform.

The idea is to use different problems with big values for N and M, and assess how the model will behave. We discussed the time complexity for the DTLZ1 problem

in last chapter, but it was generated using random solutions. We want to test some other problems using standard algorithms.

After some tests, we decide to use three different problem sets with $N = 600$ and $M = 10$: DTLZ1, C2-DTLZ2 [Jain and Deb, 2014], and MaF07 [Li et al., 2018] using MOEA/D and NSGA-III algorithms. C2-DTLZ2 and MaF07 are irregular many-objective problems, and besides other features, it brings clusters in their Pareto front. We used the same MIP solver and workstation as described before in these experiments.

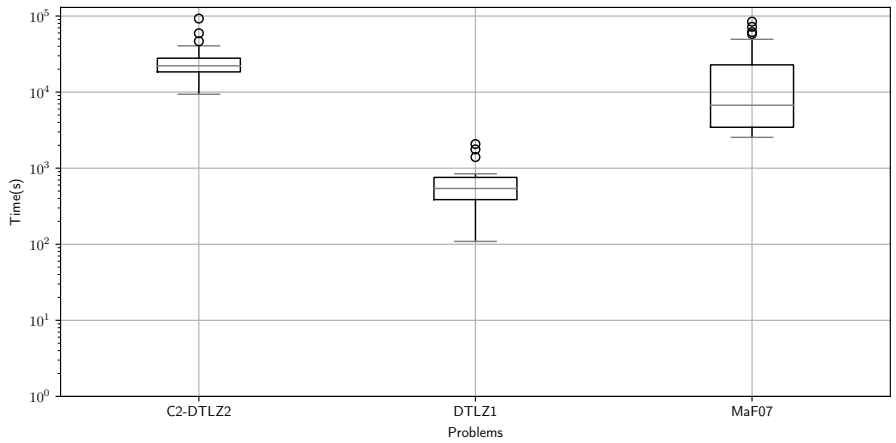

Figure 4.3. Box-plots for compact MIP-DoM computational times for problems C2-DTLZ2, DTLZ1, and MaF07. Data are obtained from 50 runs. The time spent by the model is in seconds using a log scale on y axis.

Figure 4.3 present the box plot with the time executions for each problems. Data are obtained from 50 runs. Small circles indicate some outliers. The maximum values from the small circles are $\approx$ 26 hours, $\approx$ 35 minutes, and $\approx$ 24 hours for C2-DTLZ2, DTLZ1, and MaF07, respectively. It exemplifies a case in practical limits when a user will use MIP-DoM; the time spent in some cases for C2-DTLZ2 and MaF07 is still high. Again, it poses a remaining question: Is it possible to calculate DoM with an approach that can be less time-consuming?

4.5 Concluding remarks

In Chapter 3 we have discussed the first MIP model to compute DoM (based on the assignment problem perspective). However, this formulation was complex and brought a drawback related to the model time spent as the number of solutions in each set increases.

In this Chapter, a new compact formulation has been presented. This formulation is more concise and straightforward. Moreover, compared to the former formulation, the model has shown a reduced number of binary, continuous variables and the number of constraints.

Experiments have assessed the compact formulation behavior. First, a validity comparison with the former model has been made. Afterward, a time spent comparison, in which we verify that the compact formulation improves some orders of magnitude in time spent. It mainly occurred when there were increments in the number of solutions, but there were still improvements concerning the relationship between the time spent and the number of objectives.

However, even with the compact formulation improvement, we have observed a polynomial behavior regarding the number of objectives, while the former model has presented a linear relation. This fact has motivated us to investigate some situations where there could be some practical limits to the compact model. This test has shown us that there were cases where the compact formulation still took hours to compute.

Besides this fact, we believe that the improvement in time spent was relevant, and the new compact formulation for the general MIP-DoM may be easily understood by the EMO/EMaO community. Furthermore, we hope the compact model could help address a broader number of scenarios.

Chapter 5

An approximate MIP-DoM Calculation

5.1 Introduction

In Chapter 4, a compact MIP model to calculate DoM was discussed. It brought an appropriate time spent reduction. Some computation instances take minutes, but others still take hours. This fact is inherent to the number of objectives, cardinality, and some inner characteristics involving the two solution sets, $\boldsymbol{P}$ and $\boldsymbol{Q}$. A question was posed: Is it possible to calculate DoM with an approach that can be less time-consuming? In this chapter, we want to address this question and present an algorithm that is able to calculate DoM using a MIP-DoM model while maintaining a trade-off between time-consuming and accuracy.

For simplicity and to explain your motivation, let us assume that the number of elements in $\boldsymbol{P}$ and $\boldsymbol{Q}$ are equal and given by L. The time complexity function is defined by $T = \alpha L^{\beta}$, in which α and β are two coefficients obtained using a linear regression in a log-log plot with T and L. The primary motivation is to explore some methodology that can reduce T. The main idea is related to a 'divide and conquer' approach to deal with the combinatorial nature of DoM.

The initial hypothesis behind an approximate MIP-DoM computation is to use a preliminary cluster strategy. Based on this, it is possible to establish the plain Proposition 2:

Proposition 2. *Let $\boldsymbol{P}$ and $\boldsymbol{Q}$ be two solution sets with an equal number of elements defined by L, a DoM($\boldsymbol{P}$, $\boldsymbol{Q}$) calculation with a complexity function defined by $T = \alpha L^{\beta}$, in which α and β are two positive coefficients. Assuming C as the number of non-*

overlapping clusters, with the same number of elements, formed by elements from $\boldsymbol{P}$ *and* $\boldsymbol{Q}$*. Then, the new complexity time function can be defined as* $T_C = C\left[\alpha\left(\frac{L}{C}\right)^{\beta}\right]$*. If* $\beta \geq 1$*, then* $T_C \leq T$*.*

Proof. By contraposition, suppose that $T_C > T$. In this way,

$$\begin{aligned} T_C > T \Rightarrow & \alpha\left(\frac{L}{C}\right)^{\beta} C > \alpha L^{\beta} \\ & \left(\frac{L^{\beta}}{C^{\beta}}\right) C > L^{\beta} \\ & \frac{1}{C^{\beta-1}} > 1 \\ & C^{1-\beta} > 1. \end{aligned}$$

Then, we have that $1 - \beta > 0 \Rightarrow \beta < 1$. □

The MIP-DoM calculation for some problem set instances has found $\beta > 1$, considering the former model or the compact model. For example, for 5, 10 and 15-objective in compact formulation in DTLZ1 problems, $\beta \approx 1.972$, 2.134, and 2.500, respectively.

Proposition 2 represents the rationale behind improving the MIP-DoM time spent calculation, or the 'divide' phase of our proposition. A complement to this proposition is related to the 'conquer' phase. It proposes a new MIP-DoM execution using the $\boldsymbol{P'}$ points with some penalization information coming from the 'divide' phase. Consequently, it is essential to observe that $|\boldsymbol{P'}| << |\boldsymbol{P}|$, which alleviates the time spent by this phase and shows that the merge phase does not represent a bottleneck (as will be presented in our test sets).

Some research questions (RQs) and their implementation details regarding the Proposition 2 must be addressed:

1. **RQ1:** How to cluster the solution sets $\boldsymbol{P}$ and $\boldsymbol{Q}$?
2. **RQ2:** How to associate clusters of two $\boldsymbol{P}$ and $\boldsymbol{Q}$?
3. **RQ3:** How to compute an approximate DoM value from individual cluster-wise DOM values?

These questions are addressed in this chapter. **RQ1** is related to clustering the solution sets. We have tested some different algorithms such as affinity propagation [Wang et al., 2019], mean-shift [Yizong Cheng, 1995], and K-means [Pham et al., 2005].

The affinity propagation algorithm presents some context properties such as numerical stability and deterministic behavior. **RQ2** should try to assign each cluster from $\boldsymbol{P}$ to each cluster from $\boldsymbol{Q}$; we solve the problem as an assignment problem. Finally, **RQ3** discusses a two-phase approach in which MIP-DoM approximation maintains the exact MIP-DoM values and properties. Considering the experiments performed, it can generate values with an estimated error of less than 0.40% on average and reduce the time spent by up to 5400 times.

This chapter is organized as follows: Section **5.2** presents some additional concepts to our final approach: our justification of the final choice related to the affinity propagation cluster algorithm, the assignment problem formulation, and the two-phased proposal to the problem. In Section **5.3** our algorithm is presented with a further discussion and some relevant details that deal with the approximate MIP-DoM calculation. Section **5.4** brings some multi- and many-objective experiments to validate the approximate MIP-DoM using affinity propagation with the optimal MIP-DoM value and compare the time spent by the models. The final considerations and comments are presented in Section **5.5**.

5.2 Algorithm background

5.2.1 RQ1: Clustering

We discuss the first research question on how to cluster the solution sets $\boldsymbol{P}$ and $\boldsymbol{Q}$. As discussed in Chapter 2, clustering aims to divide the data points into subsets in such a way that the elements belonging to the same subset are similar to each other.

Taking our context into consideration, we would like to have some clustering method properties:

- *Parameters*: typically, clustering methods involve some parameters as inputs. The difficulty is how to pick settings for those parameters. In this sense, we would like algorithms that have as fewer parameters as possible;

- *Clustering determinism*: some cluster algorithms have a stochastic component (e.g., a random initialization or a sampling approach). Our preference is related to stable methods. We want reproducible results because if we run the same algorithm using the same parameters, we always need to get the same clusters and, consequently, the same final results. Still, if we vary some parameters, we want a change in a somewhat stable, predictable manner. It is a fundamental requisite for our approximation method in a way to reduce its variability;

Algorithm 1: Affinity propagation

Input: $\boldsymbol{S}$, λ, p_r, T_a
Output: $\{c_1, \ldots, c_C\}$

$\boldsymbol{R}$,$\boldsymbol{A} \leftarrow 0, 0$
$t \leftarrow 0$
$\forall k : s[k,k] \leftarrow p_r$
while $t < T_a$ **do**
  // **propagating responsibility**
  $$\forall i,j : \rho[i,j] \leftarrow \begin{cases} (i \neq j), & s[i,j] - \max\limits_{\forall k:k\neq j} (a[i,k] + s[i,k]) \\ (i = j), & s[i,j] - \max\limits_{\forall k:k\neq j} (s[i,k]) \end{cases}$$
  // **propagating availability**
  $$\forall i,j : \alpha[i,j] \leftarrow \begin{cases} (i \neq j), & \min(0, r[j,j] + \sum\limits_{\forall k:k\neq i,j} \max(0, r[k,j])) \\ (i = j), & \sum\limits_{\forall k:k\neq i} \max(0, r[k,j]) \end{cases}$$
  // **updating responsibility**
  $\forall i,j : r[i,j] \leftarrow (1-\lambda)\rho[i,j] + \lambda r[i,j]$
  // **updating availability**
  $\forall i,j : a[i,j] \leftarrow (1-\lambda)\alpha[i,j] + \lambda a[i,j]$
  $t \leftarrow t + 1$
end
$\forall i : c[i] \leftarrow \underset{k}{\operatorname{argmax}}(r[i,k] + a[i,k])$
return $\{c_1, \ldots, c_C\}$

- *Performance*: a clustering algorithm with a computational complexity function better than our Proposition 2 is a suitable choice for the problem. Moreover, we should prefer a clustering algorithm that can scale up to large data sets.

Based on those criteria, we review some clustering methods. Unfortunately, k-Means and Mean-shift do not present the numerical stability feature suitable for the MIP-DoM approximation. Yet, for our calculation approach, this is a fundamental requisite. As an alternative, affinity propagation clustering is a deterministic algorithm (regarding its initial parameters) and is described next.

The clustering algorithm treats all points as potential cluster centers. The process starts with the similarity matrix ($\boldsymbol{S}$), which is constructed in a pairwise manner using a similarity function, such as the negative Euclidean distance (values near zero indicate similarity otherwise, the degree of dissimilarity between the points) [Wang et al., 2019].

Given $\boldsymbol{S}$, the method tries to find 'exemplars' that maximize the net similarity. The algorithm is a graphing approach and can be viewed as a message-passing process through edges with two kinds of messages exchanged among data points. The respon-

sibility matrix $\boldsymbol{R}$ with $r[i, j]$ elements in with $i, j \in \{1, \ldots, L\}$, means how well suited, in an accumulated way, is the point j to be an exemplar to i. The availability matrix $\boldsymbol{A}$ with $a[i, j]$ elements in with $i, j \in \{1, \ldots, L\}$, is a message from data point j to i reflecting how appropriate it would be for data point i to choose data point j as its exemplar.

The whole process is detailed in Algorithm 1. All responsibilities and availabilities are initialized at 0. The λ parameter is a damping factor used to avoid numerical oscillations. High value leads to slow run time but avoids numeric oscillations. There is an interval recommendation to $\lambda \subseteq [0.5, 1.0]$.

Another critical parameter is the preference p_r, which is used to fill up the matrix diagonal and indicates how appropriate the i-th point is to be selected as an 'exemplar'. High preference values will cause the method to find many clusters. Contrarily, low values will generate a small number of clusters. Preference can be set as the median similarities value.

The propagating responsibility between point i and j is indicated by $\rho[i, j]$. The expression in line 7 of Algorithm 1 indicates the update using the similarity value between points i and j minus the best value for a point i. The diagonal matrix values are updated, as well, using the preference minus the second-best similarity value. Similarly, $\alpha[i, j]$ is the propagating availability. Finally, the diagonal matrix is updated, reflecting the accumulated evidence that the point j is suitable as an 'exemplar', based on the positive responsibilities of k other elements.

The messages are updated and kept until convergence is met or the iteration reaches a specific number.

The original algorithm has $O(L^2\ T_a)$ time complexity . However, some variants can improve it [Wang et al., 2019].

We decide to use the affinity propagation clustering algorithm in our approach to approximate MIP-DoM calculation.

5.2.2 RQ2: Assignment

Considering each cluster from $\boldsymbol{P}$ and $\boldsymbol{Q}$ how to assign the subsets to calculate the MIP-DoM for each subset assignment? The classical assignment problem is used in many works in mathematical programming and management sciences, as discussed in Chapter 2.

In Figure 5.1 we can see a bipartite graph with two sets $\boldsymbol{C^P}$ and $\boldsymbol{C^Q}$ which are the clusters centroids/'exemplars' created from $\boldsymbol{P}$ and $\boldsymbol{Q}$, respectively. Our problem leads to an unbalanced version of the assignment problem. Here, our objective is to

assign to all members of $\boldsymbol{C^Q}$ one member from $\boldsymbol{C^P}$ with the minimum cost, represented by the edges. The unbalanced version indicates that a member of $\boldsymbol{C^P}$ could not be used in the final solution. Figure 5.1 presents an example, and a valid assignment from $\boldsymbol{C^P}$ to $\boldsymbol{C^Q}$ representing our case.

The mathematical model for the assignment problem presented in Figure 5.1 can be given by the Equation (5.1). Concerning the cluster number, it is given by $|C^P|$ and $|C^Q|$. The $x_{ij} = 1$ indicates if the i-th member of $\boldsymbol{C^P}$ is assigned to the j-th member of $\boldsymbol{C^Q}$, and 0 otherwise. The c_{ij} represents the distance to j-th assigned to the i-th. The first constraint ensures that every member in $\boldsymbol{C^Q}$ is assigned to only one member of $\boldsymbol{C^P}$.

$$\begin{aligned} &\text{Minimize} \quad \sum_{i=1}^{|\boldsymbol{C^P}|} \sum_{j=1}^{|\boldsymbol{C^Q}|} c_{ij} x_{ij}, \\ &\text{subject to:} \begin{cases} \sum_{i=1}^{|\boldsymbol{C^P}|} x_{ij} = 1 & \forall j \in \{1, 2, \ldots, |\boldsymbol{C^Q}|\}, \\ x_{ij} = 0 \text{ or } 1. \end{cases} \end{aligned} \tag{5.1}$$

It is worth emphasizing that x_{ij} is a binary variable.

The time complexity in the assignment problems can vary due to problem specificity. We use the Hungarian method to solve the linear model detailed in Equation (5.1). It presents the following complexity: the $O(ms + s^2 \log(r))$ in which $m = |\boldsymbol{C^P}||\boldsymbol{C^Q}|$ as the number of edges from $\boldsymbol{P}$ to $\boldsymbol{Q}$, $r = \min(|\boldsymbol{C^P}|, |\boldsymbol{C^Q}|)$, and $s = \max(|\boldsymbol{C^P}|, |\boldsymbol{C^Q}|)$ [Ramshaw and Tarjan, 2012] .

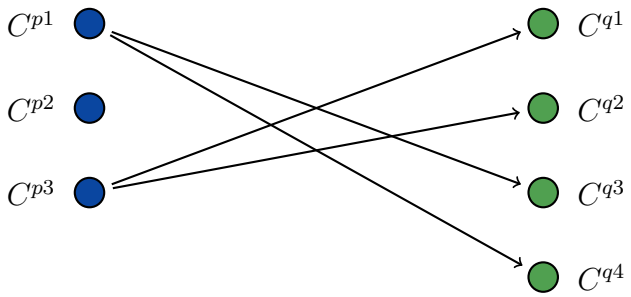

Figure 5.1. One possible example of assignment between $\boldsymbol{C^P}$ and $\boldsymbol{C^Q}$. The edges indicating the assignment can be computed using the minimum distance from members $\boldsymbol{C^P}$ to $\boldsymbol{C^Q}$.

5.2.3 RQ3: A Two-phase Approximate DoM Approach

How to use the MIP-DoM values in a 'divide and conquer' approach and guarantee that it preserves the original MIP-DoM value? Based on the combinatorial nature of DoM

a method that uses clustering followed by an assignment step to calculate DoM values will probably, generate a solution with a little chance of being a good approximation. The hypothesis is to improve the intermediary values from the 'divide' phase turning the final value even better in the 'conquer' phase. We present more details in the next section. In this way, it makes sense to observe that the approximation should always be greater than the original MIP-DoM value.

5.3 Affinity propagation with a two-phase MIP-DoM calculation

The MIP-DoM compact formulation, added with the affinity propagation, and the assignment model are the building blocks of our approximation method. Hereafter, all the referenced methods will be put together to explain the two-phase approach to the problem.

5.3.1 Geometric Intuition

A simple and graphical example can illustrate our approximation proposal. Consider two solution sets $\boldsymbol{P}$ and $\boldsymbol{Q}$ in the bi-objective space, and with four solutions in each set, see Figure 5.2 plot 1. Our intention is to calculate MIP-DoM($\boldsymbol{P}$, $\boldsymbol{Q}$) in a two-phase manner, which mimics a 'divide and conquer' approach.

In Phase 1, the first step is to apply the clustering algorithm in both sets. For that task, affinity propagation cluster is used in $\boldsymbol{P}$ and $\boldsymbol{Q}$, separately. In Figure 5.2, plot 1, it is possible to observe that we have three clusters generated by the method. Let us call $|\boldsymbol{C^P}|$ and $|\boldsymbol{C^Q}|$ the number of clusters generated, in plot 2, $|\boldsymbol{C^P}| = |\boldsymbol{C^Q}|$, but it is not a requirement of our proposal. Lets call $\{\boldsymbol{P^1}, \cdots, \boldsymbol{P}^{|\boldsymbol{C^P}|}\}$ as a set formed with all the subsets created in the cluster procedure. The assertion in which $\boldsymbol{P^1} \bigcup \boldsymbol{P^2} \bigcup \cdots \bigcup \boldsymbol{P}^{|\boldsymbol{C^P}|} = \boldsymbol{P}$ is valid. The similar assertion is true to $\boldsymbol{Q}$.

After the cluster method we have $L_{P^1} + \cdots + L_{P|\boldsymbol{C^P}|} = L_P$ and $L_{Q^1} + \cdots + L_{Q|\boldsymbol{C^Q}|} = L_Q$. In Figure 5.2, phase 1, plot 1, we can observe three clusters from $\boldsymbol{P}$ and $\boldsymbol{Q}$, respectively, $\{\boldsymbol{P^1}, \boldsymbol{P^2}, \boldsymbol{P^3}\}$ and $\{\boldsymbol{Q^1}, \boldsymbol{Q^2}, \boldsymbol{Q^3}\}$. Each solution set member, now is a member belonging to a cluster. Affinity propagation elects a solution as an 'exemplar' to represent the cluster. However, we decide to use the ideal point to represent each cluster: constructed by each objective's best value for all solutions owned by the cluster. The ideal points are depicted in Figure 5.2, phase 1, plot 2.

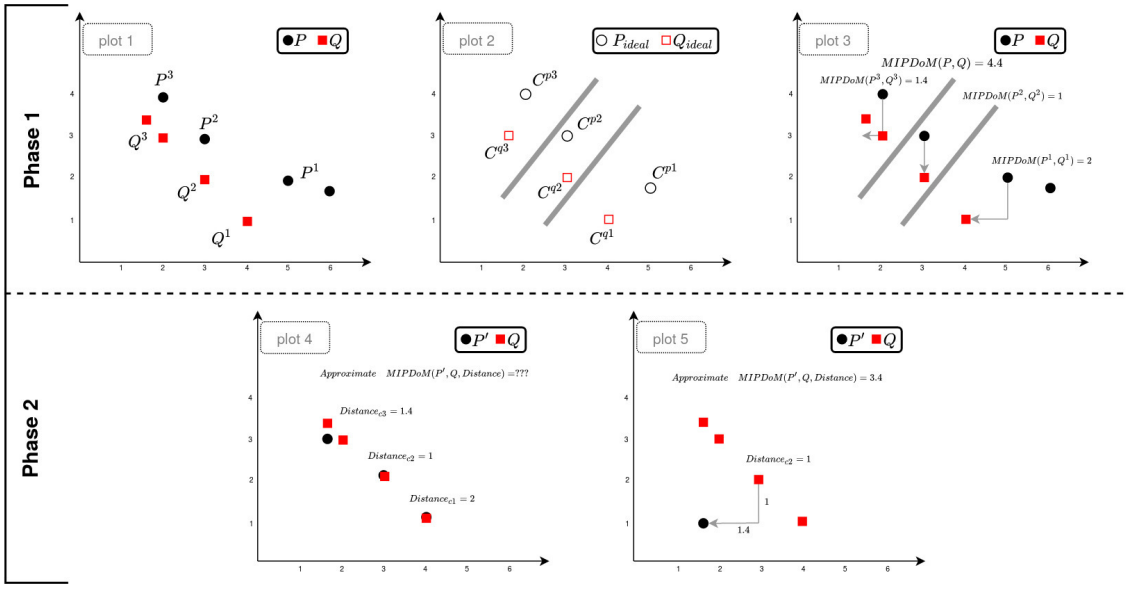

Figure 5.2. A bi-objective example of how the two-phase approximate MIP-DoM using a clustering algorithm works. There are two phases and five plots in order. Plot 1 defines the two solution sets $\boldsymbol{P}$ and $\boldsymbol{Q}$ and the clusters created for each one. In plot 2, using the clusters, the ideal points are generated, and the assignment is done using such ideal points: built by each objective's best value for all solutions owned by each cluster. Plot 3 shows the original MIP-DoM calculated for each cluster. It generates a MIP-DoM value for each cluster. The generated MIP-DoM value is retained, and it is associated with each $\boldsymbol{P'}$ generated. Plot 4, phase two is started, and the approximate MIP-DoM will be calculated, but a new parameter is related to the intermediary MIP-DoM value, which comes from the first phase execution (plot 3). Finally, plot 5 presents the approximate MIP-DoM and the final solution calculated using the distance values.

The next step is how to assign the clusters 'exemplars' using $\boldsymbol{C^P}$ and $\boldsymbol{C^Q}$ sets. Here, the assignment problem does not use the generated 'exemplars'; instead, the ideal point coming from each cluster is used. In plot 2, these points are emphasized. With such points, the assignment is done using the Equation (5.1). The c_{ij} is calculated using the Manhattan distance and represent the distance between the i-th cluster ideal points from $\boldsymbol{P}$ to j-th cluster ideal points from $\boldsymbol{Q}$.

The original MIP-DoM (compact formulation) is calculated for each assigned pair of clusters, generating intermediary MIP-DoM values. The method returns the MIP-DoM value and $\boldsymbol{P'}$ for each assigned cluster pair. In plot 3, Figure 5.2, the values generated by each MIP-DoM execution are shown. In a 'divide and conquer' approach, the first estimate appears in phase 1. Then, considering the problem, MIP-DoM is calculated by summing the parts obtained from assignments. The first value is 4.4.

In plot 4, phase 2 is started, and now the $\boldsymbol{P}$ solution set is substituted by the $\boldsymbol{P'}$ coming from all MIP-DoM cluster executions in phase 1. It is relevant to note that the MIP-DoM offers not only a quality indicator value. The $\boldsymbol{P'}$ is a new solution set

that can or can not dominate $\boldsymbol{Q}$ completely (it can not dominate totally because the cluster approximation makes MIP-DoM 'blind' for the whole solution set). However, if we re-execute MIP-DoM using the new solution set $\boldsymbol{P'}$ and the distance coming from phase 1, it is possible to improve the final solution. An approximate MIP-DoM receives an additional parameter, the intermediary value, indicated by ***Distance*** in plot 4.

Finally, the approximate MIP-DoM is calculated. It is shown in Figure 5.2, phase 2, plot 5, with the final solution (using the ***Distance*** MIP-DoM value and the new $\boldsymbol{P'}$). Considering the MIP-DoM model, a new component is added to the objective function. It works as a penalization strategy using the MIP-DoM execution values as a parameter in this approximated MIP-DoM run (associated with each $\boldsymbol{P'}$ subset comes from the first execution, we have a distance that represents a penalization for the movement already done). The results from the approximated MIP-DoM are the same as before. It generates the new, improved value (now with 3.4) and the new $\boldsymbol{P'}$, which now dominates $\boldsymbol{Q}$ completely.

5.3.2 Proposed algorithm

With the algorithm's intuition explained, it is possible to discuss other details. Additional aspects are presented in Algorithm 2. The $\boldsymbol{P}$ and $\boldsymbol{Q}$ solution sets are the first two input parameters, and the p_r, λ, and T_a are the last ones, and related to affinity propagation clustering.

The first step is related to the similarity matrix calculation: *SIMILARITYMATRIX*. As suggested in the affinity propagation algorithm, we use the negative Euclidean distance. The total number of pairwise comparisons to calculate the pairwise distance matrix is $\lceil L(L-1)/2 \rceil$, and each comparison can be a vector operation with M summations.

Secondly, we create the clusters using the affinity propagation, as detailed in Algorithm 1: *AFFINITYPROPAGATION*. We use the similarity matrix obtained from the last step. As we want the numerical stability, in a try and error approach, we decide the damping factor λ to 0.9. The p_r is another clustering parameter received in the approximation MIP-DoM function, which will be better discussed afterward. The final affinity propagation parameter is the maximum number of iterations. It is used as a default value of $T_a = 200$. The affinity propagation method has the worst time complexity as $O(L^2 T_a)$.

The next step is how to assign the clusters from $\boldsymbol{P}$ to every cluster from $\boldsymbol{Q}$. The *ASSIGNMENT* method has two parameters related to clusters from $\boldsymbol{P}$ and $\boldsymbol{Q}$ which are formed by the elements in each cluster, including the 'exemplars'. The *ASSIGNMENT*

method calculates the ideal points in each cluster. The number of clusters is $|\boldsymbol{C^P}|$ and $|\boldsymbol{C^Q}|$ meaning the number of clusters suggested by the affinity propagation clustering method. Assuming $|\boldsymbol{C^P}| = |\boldsymbol{C^Q}|$, the time complexity for this assignment in the worst case is $O(|\boldsymbol{C^P}|^3)$. It is important to note that $|\boldsymbol{C^P}| << L$, which emphasizes the use of the assignment formulation.

From the assignment step, we have tuples which indicate the clusters from $\boldsymbol{P}$ assigned to every cluster from $\boldsymbol{Q}$. For each tuple in the assignment, it is possible to execute the model to calculate the MIP-DoM value: *MIPDoM* function in line 14 Algorithm 2. D_c is the intermediary MIP-DoM value, and $\boldsymbol{P'_c}$ are the solution sets generated by MIP-DoM. These values and solution sets are retained for the next and final step.

The approximate MIP-DoM step involves the use of $\boldsymbol{D}$ values and $\boldsymbol{P'}$ solution set: *APPROXIMATEMIPDoM* in line 20. The *APPROXIMATEMIPDoM* differ from *MIPDoM* function in line 14, just by the $\boldsymbol{D}$ vector added in the objective function, all the other details remained the same.$\boldsymbol{D}$ is a penalization vector used in the objective function in *APPROXIMATEMIPDoM*

The cardinality of $\boldsymbol{P'}$ is much smaller than $\boldsymbol{P}$, in other words, $L_{P'} \ll L_P$. It is relevant to note that if at least one solution in $\boldsymbol{P'}$ already dominates all $\boldsymbol{Q}$, the approximate MIP-DoM, *APPROXIMATEMIPDoM*, the value will be the same as the MIP-DoM, *MIPDoM*, the value previously executed, using the $\boldsymbol{D}$ values as the final result. The time complexity is decreased due to the $\boldsymbol{P'}$ cardinality.

Concerning all complexity time component functions in the Algorithm 2, we expect that our approximate approach has, in the worst case, a $O(L^2)$ complexity time.

5.4 Experiments

The first group of experiments intends to show how the approximate MIP-DoM behaves compared to MIP-DoM compact formulation. We would like to understand the differences between MIP-DoM values and how fast the approximate can be. The second group of experiments tries to deal with the limits for the compact formulation presented in Chapter 4.

5.4.1 Multi-objective comparisons

To validate our propositions, some multi-objective benchmark problems are selected. We use the same test sets as in Chapter 3 and 4. These problem test sets have three

Algorithm 2: Approximate MIP-DoM with affinity propagation

Input: $\boldsymbol{P}$,$\boldsymbol{Q}$,p_r,λ,T_a
Output: $ApproximateValue$

```
// PHASE 1
// calculating the similarity matrix
S_P ← SIMILARITYMATRIX(P)
S_Q ← SIMILARITYMATRIX(Q)
// generating the clusters from P and Q
P clusters ← AFFINITY PROPAGATION(S_P, λ, p_r, T)
Q clusters ← AFFINITY PROPAGATION(S_Q, λ, p_r, T)
// assigning clusters using the ideal points
Assignment ← ASSIGNMENT(P clusters, Q clusters)
// calculating the MIP-DoM for each assigned cluster
P' ← ∅
D ← ∅
foreach {(p_c, q_c)} ∈ Assignment do
    D_c, P'_c ← MIPDoM(p_c, q_c)
    P' ← P' ⋃ P'_c
    D ← D ⋃ D_c
end
// PHASE 2
// final calculation using the approximate MIP-DoM
ApproximateValue ← APPROXIMATEMIPDoM(P', Q, D)
return ApproximateValue
```

objectives and come from *DTLZ* and *WFG* families. We use DTLZ1, DTLZ2, DTLZ3, and DTLZ7.

Some algorithms are used to generate the approximated Pareto front and use the MIP-DoM indicator outcomes. We use IBEA, NSGA-II, NSGA-III, MOEA/D, and SPEA2 to generate solution sets. The maximum number of fitness evaluations in the experiments is set to 10,000, and each algorithm is executed 21 times.

We intend to compare two main factors: the time spent by the original MIP-DoM ('exact') vs. our approximate approach and the error added at the value calculated by our approximation.

A critical parameter in the affinity propagation MIP-DoM approximation is the preference, p_r, which highly influences the number of clusters. This parameter is input in Algorithm 2. In the initial tests, we observe that the preference value significantly influences the final result; in general, the median value is used. Therefore, we decide to use a brute force strategy to assess the best testing of some percentile. It can represent values near minimum, median, and maximum value of p_r. Therefore, we establish

0.01 meaning the minimum, 0.05, 0.5 as the median, 0.95 and 0.99 percentile as the maximum (this value is calculated using the similarity matrix).

In Table 5.1, the experiment results are presented. This table is ordered by the best algorithm results obtained by each algorithm in each problem set. In this table, the values obtained from the exact MIP-DoM model are labeled as 'Exact' while the values obtained in the best MIP-DoM approximation using affinity propagation, using the best preference value, are labeled as 'Approx.' In the same manner, table shows the time spent by each execution. The time spent in the 'Approx.' column summates all preference percentiles.

Table 5.1. Exact MIP-DoM($\boldsymbol{P}$,$\boldsymbol{Q}$) and Approximate MIP-DoM($\boldsymbol{P}$,$\boldsymbol{Q}$) values for the *DTLZ* and *WFG* families for comparison among some algorithms. Observe that $\boldsymbol{P}$ is the solution set generated by the algorithm, and $\boldsymbol{Q}$ is the non-dominated solution set from all algorithms results combined. The two column's groups detail the MIP-DoM values (Exact vs. Approximate) and time spent in seconds.

Problem	**Algorithms**	**MIP-DoM**		**Time spent (s)**	
		Exact	*Approx.*	*Exact*	*Approx.*
DTLZ1	*IBEA*	0.312	0.312	32.59	**22.24**
	NSGA-III	0.551	0.551	**7.21**	10.87
	MOEA/D	1.630	1.630	**22.25**	27.08
	NSGA-II	6.422	6.422	4.49	**4.45**
	SPEA2	11.139	11.139	5.04	**2.47**
DTLZ2	*MOEA/D*	0.970	0.970	1138.66	**65.77**
	IBEA	1.000	1.000	9980.85	**286.50**
	NSGA-III	1.004	1.004	10875.98	**288.71**
	NSGA-II	1.013	1.013	3950.80	**253.25**
	SPEA2	1.022	1.022	1129.99	**72.73**
DTLZ3	*IBEA*	0.049	0.049	16.72	**5.00**
	NSGA-III	18.652	18.652	14.15	**7.86**
	MOEA/D	25.374	25.374	122.72	**3.94**
	NSGA-II	51.208	51.208	26.78	**5.55**
	SPEA2	150.870	150.870	12.63	**6.56**
DTLZ7	*NSGA-III*	1.402	1.402	3345.07	**364.33**
	IBEA	1.468	1.508	8075.00	**135.76**
	MOEA/D	1.695	1.695	2520.99	**23.90**
	NSGA-II	1.782	1.791	8260.20	**363.03**
	SPEA2	2.184	2.184	2040.71	**21.74**
WFG1	*IBEA*	1.230	1.230	60.27	**22.33**
	MOEA/D	1.557	1.557	158.81	**36.62**
	SPEA2	1.659	1.659	341.12	**78.79**
	NSGA-III	1.822	1.822	950.14	**126.53**
	NSGA-II	1.944	1.944	184.51	**60.14**
WFG2	*IBEA*	1.503	1.503	283.05	**39.53**
	NSGA-III	1.571	1.571	181.02	**50.57**
	MOEA/D	1.683	1.683	328.40	**17.25**
	NSGA-II	1.687	1.687	140.20	**50.57**
	SPEA2	1.859	1.859	879.41	**24.47**
WFG3	*IBEA*	1.889	1.889	1368.77	**269.54**
	NSGA-III	2.609	2.609	2334.96	**855.45**
	NSGA-II	2.630	2.630	3491.16	**440.10**
	MOEA/D	2.820	2.820	2238.82	**1009.81**
	SPEA2	3.178	3.178	2119.42	**425.97**
WFG9	*IBEA*	1.965	2.024	673.26	**23.05**
	NSGA-III	2.194	2.194	348.48	**25.25**
	MOEA/D	2.331	2.331	263.90	**20.80**
	NSGA-II	2.601	2.601	2898.88	**22.89**
	SPEA2	2.759	3.030	748.76	**27.74**

Table 5.2. Some details about the approximation algorithm search process due to the preference parameter. DTLZ2 and WFG3 are further presented, and they are the slowest experiments. Three column groups are presented with the respective percentile preference: MIP-DoM value, number of clusters suggested by the affinity propagation cluster, and the time spent for each case.

Problem	**Algorithms**	**MIP-DoM**	**MIP-DoM Approx. values**					**Number of clusters**					**Time spent**				
		Exact	**Preference percentile**														
			0.01	*0.05*	*0.50*	*0.95*	*0.99*	*0.01*	*0.05*	*0.50*	*0.95*	*0.99*	*0.01*	*0.05*	*0.50*	*0.95*	*0.99*
DTLZ2	*MOEA/D*	**0.970**	1.013	1.011	0.974	**0.970**	**0.970**	4	5	4	10	24	3.27	5.56	4.85	5.48	46.61
	IBEA	**1.000**	**1.000**	**1.000**	**1.000**	1.009	**1.000**	4	4	5	2	23	3.30	3.34	2.56	64.10	216.54
	NSGA-III	**1.004**	**1.004**	**1.004**	**1.004**	**1.004**	**1.004**	4	4	6	10	26	3.14	3.17	3.12	4.23	278.22
	NSGA-II	**1.013**	**1.013**	1.041	**1.013**	**1.013**	**1.013**	3	2	5	9	26	9.02	5.56	4.48	3.57	230.62
	SPEA2	**1.022**	**1.022**	**1.022**	**1.022**	**1.022**	**1.022**	3	3	4	9	24	11.02	10.17	4.26	2.19	55.26
WFG3	*IBEA*	**1.889**	**1.889**	**1.889**	2.072	**1.889**	2.268	3	3	4	1	7	4.51	5.10	0.94	263.65	0.44
	NSGA-III	**2.608**	3.034	3.034	**2.609**	**2.609**	3.099	2	2	1	1	2	233.94	234.84	621.46	603.89	0.05
	NSGA-II	**2.630**	**2.630**	**2.630**	2.888	2.999	3.081	1	1	4	7	2	423.63	428.06	9.78	6.64	0.05
	MOEA/D	**2.820**	3.058	3.058	**2.820**	3.058	3.232	2	2	1	2	3	237.87	237.59	771.83	238.65	0.11
	SPEA2	**3.178**	**3.178**	**3.178**	**3.178**	**3.178**	**3.178**	2	2	1	6	13	215.01	216.93	188.71	2.62	19.63

The difference between the Exact vs. Approximate value in some problem sets did not exist. This is the case for DTLZ1, DTLZ2, DTLZ3, WFG1, WFG2, and WFG3. The approximation value is always greater than the exact values. The first difference appears to DTLZ7 for the IBEA algorithm, the difference is 2.72%, and for NSGA-II the percentual difference is 0.51%. The next one is related to WFG9 for IBEA, and SPEA2, with the values of 3.00%, and 9.82%. Regarding all the problem sets, the experiment average difference is 0.40%.

Regarding the time spent, there is just one problem set in which the 'Approx.' is higher than the Exact time spent: DTLZ1 for NSGA-III and MOEA/D. The first observation is that DTLZ1, in general, does not represent a critical demand/improvement in the time spent considering the exact MIP-DoM. However, the approximation approach maintained a similar time spent behavior compared with the exact MIP-DoM. The differences are subtle, and the values are next to each other.

For all other problem sets, there are improvements in the time spent. The most time-consuming problem sets are DTLZ2, DTLZ7, and WFG3. Regarding this problem sets, there is an improvement range from ~ 2 to ~ 5105 times better, which happens to WFG3 and DTLZ2 for MOEA/D.

As an approximation approach, the method is influenced by parameters. In the approximate MIP-DoM, this parameter comes from the affinity propagation algorithm: preference as p_r. It is relevant to assess the behavior of this parameter in the experiments.

A brute force strategy is used with some preference percentiles search space. Table 5.2 tries to uncover the method behavior due to the preference parameter. The two time-consuming problem sets of each family are exposed in more detail: DTLZ2 and WFG3.

Table 5.2 has three main columns exposing the MIP-DoM values, the number of clusters, and the time spent. All these columns are detailed by the respective percentile preference. The first observation related to preference is the non-monotonic behavior; MIP-DoM values, the number of clusters, or the time spent; all of them present it. For example, in WFG3 for NSGA-III, the percentile 0.01 and 0.05 shows the same MIP-DoM value 3.034, it decreases with percentile 0.50, but it is increased again with percentile 0.99.

Another interesting observation is that sometimes different percentile generated the same number of clusters and produces the same MIP-DoM value. In fact, in these cases, the same clusters were generated, resulting in the same final solution. This kind of 'memory' is one of the improvements in the iterative call of the Algorithm 2. In other words, the $AFFINITYPROPAGATION$ is called before, and if the method

generates the same clusters in which the results are known, there is no necessity to call the Algorithm 2 again. This is a relevant fact; for example, let us consider the WFG3 for the IBEA algorithm, the total spent time presented in Table 5.1 do not compute the percentile 0.05 , as it has generated the same clusters for percentile 0.01, as showed in Table 5.2.

In Table 5.2, it is possible to observe that in some cases, the number of clusters is 1. It happens for WFG3 in all algorithms but in different percentile preferences. It means that the MIP-DoM will solve the exact problem with the same number of solutions in each set. It happens for the DTLZ1 as well, for example. The exact MIP-DoM computation must be avoided and treated. An early stopping procedure was adopted in the approximate MIP-DoM, specifically in Algorithm 2 line 17: if there is no improvement during 180 seconds, the MIP-DoM terminates and returns the current solution. The early stopping is an additional parameter, and it must be specified for each case.

One last interesting observation comes from the SPEA2 in both problem sets, and in DTLZ2 for NSGA-III algorithm. The exact value is always obtained, and the percentile variance is innocuous considering the MIP-DoM value.

5.4.2 Extending limits in many-objective scenarios

In the last chapter, we discussed possible limits for the proposed compact MIP-DoM formulation. It was observed that some instances, even using the compact formulation, can spend some hours to compute. This behavior can make the use of MIP-DoM unfeasible in practical situations. Using the MIP-DoM approximation, we intend to provide an alternative for this issue.

Again, as in Chapter 4, we use C2-DTLZ2, DTLZ1, and MaF07 problem sets. We want to observe the MIP-DoM value differences and the total time the methods spend, comparing the approximation method with the compact formulation. Experiments using 50 runs each were executed with C2-DTLZ2, DTLZ1, and MaF07 problem instances, $N = 600$, and $M = 10$ were set. The solution sets were generated with MOEA/D and NSGA-III algorithms.

Figure 5.3 present three different plots for C2-DTLZ2 problem set. Starting left to right, the first plot shows the MIP-DoM values obtained from 50 runs considering the approximation method, labeled as 'Approx.', and the compact formulation, labeled as 'Exact'. We can see that there are 3 points in which the values obtained by the approximation method are indicated as outliers. Disregard these 3 cases, all the other values are near each other. These outliers compromise the Box plot view due to the

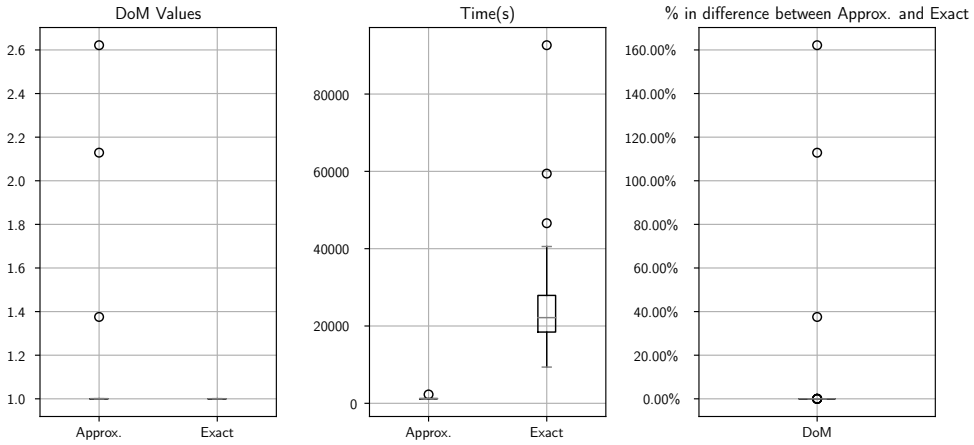

Figure 5.3. A comparison between the approximate method and the compact formulation using the C2-DTLZ2 problem set with 50 runs, N=600 and M=10. Plots with comparisons for DoM values and time spent by the methods. The last box plot shows a percentage difference between the methods.

plot scale; in Figure 5.4 we extract the outliers to give a better view. We can observe that the 'Approx.' median value is higher than the MIP-DoM 'Exact' median, which is an expected behavior.

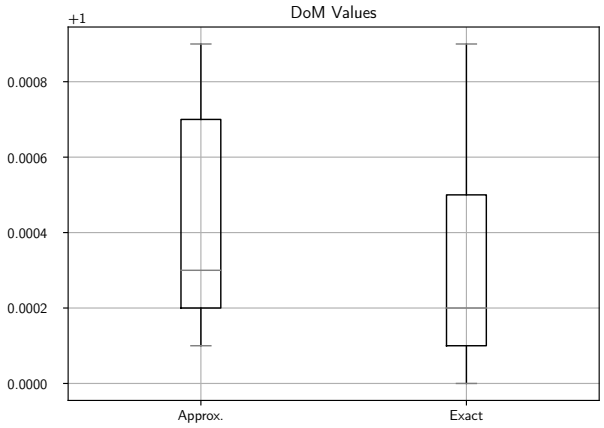

Figure 5.4. A comparison between the approximate method and the compact formulation using the C2-DTLZ2 problem set with 50 runs, N=600 and M=10. Plots with comparisons for DoM values **without outliers**.

The second plot in Figure 5.3 shows the time spent by the two methods, and we can observe 3 extreme cases in the compact model. They are the same cases pointed out in the first plot on the 'Approx.' Box-plot. It indicates that the compact model calculated these 3 cases, and the approximation method just did part of the job, and

it could not get the final and correct MIP-DoM value. However, the improvement in time spent comparing the approximation method with the compact formulation is significant. The median values are ~ 1200 and ~ 22000 seconds, respectively.

Finally, the last plot in Figure 5.3 shows the distribution of the difference between the MIP-DoM values from the exact to the approximation method. In all experiments, we observed that the approximation method's value is always greater than or equal to the value calculated by the compact formulation. This plot exhibits that the median difference is roughly zero percent, disregarding 3 previously mentioned cases.

Following, we test the DTLZ1 problem set. Figure 5.5 show the 3 plots again. Considering the DoM Values, there were no discrepant cases at this time, and we can observe an alignment between the Box-plots, taking the minimum, first quartile, median, third quartile, and maximum quite similar. Considering the second plot, detailing the time spent, we can observe a significant reduction. It is possible to observe two outliers in'Approx.' case, they are next to the value of 1250 seconds, and they have an equivalent behavior on the compact formulation, but with the time spent greater than 1750 seconds. The last plot shows the same information as the first plot, so there is a small percentage difference between MIP-DoM values from compact formulation to approximate method.

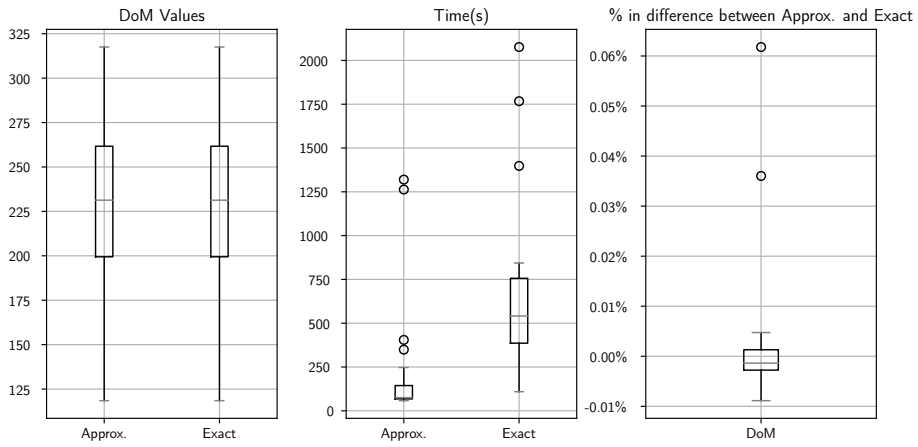

Figure 5.5. A comparison between the approximate method and the compact formulation using the DTLZ1 problem set with 50 runs, N=600 and M=10. Plots with comparisons for DoM values and time spent by the methods. The last box plot shows a percentage difference between the methods.

The final experiment uses the MaF07 problem set. It is depicted in Figure 5.6, on the first plot is possible to see that the median from 'Approx.' method is greater than the 'Exact' method. As said before, it is expected behavior. The time spent

on the compact formulation presents some outliers, the worst of them reaching more than 80,000 seconds. The median time spent is ~ 308 for the 'Approx.' method and ~ 6705 for the 'Exact' compact formulation. Considering the MIP-DoM percentage difference, the median is less than 0.78%. However, there are still some cases in which the 'Approx.' method could not generate good approximations. Considering 50 runs, there were 4 cases in which the percentual difference was more than five percent. It must be taken into account when using the approximate method.

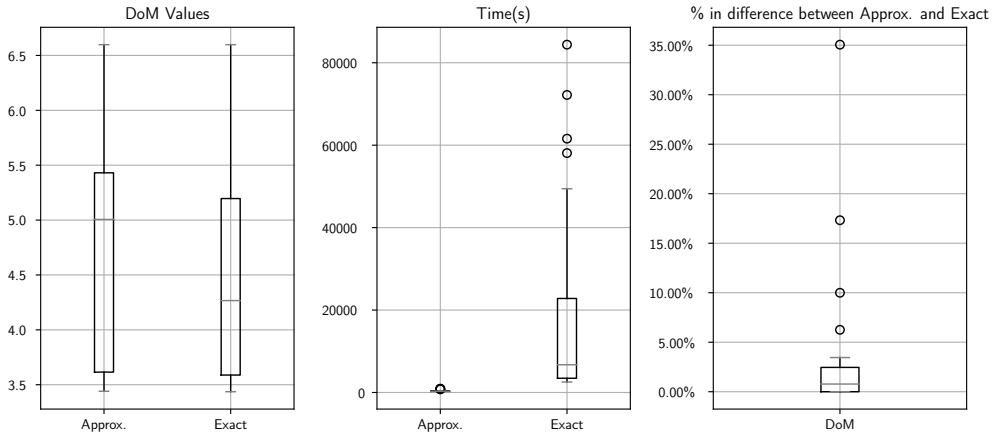

Figure 5.6. A comparison between the approximate method and the compact formulation using the MaF07 problem set with 50 runs, N=600 and M=10.Plots with comparisons for DoM values and time spent by the methods. The last box plot shows a percentage difference between the methods.

5.5 Concluding remarks

This chapter has analyzed and proposed an approximate MIP-DoM calculation using the affinity propagation cluster algorithm. The results from our experiments have indicated that the proposed Approximative MIP-DoM is computationally faster than the exact MIP-DoM compact formulation model. In addition, there were some improvements in time spent with ranges from ~ 2 to $\sim 5,000$, favoring the Approximate MIP-DoM. Moreover, the Approximate MIP-DoM provided accurate estimates for the exact MIP-DoM with a tiny average approximation error.

We also empirically presented the effects of the preference parameter. Our results demonstrated that this percentile parameter had played an essential role in using a brute force approach in the approximation method. In the future, a better way should perform different strategies to find a better preference, such as the irace approach

López-Ibáñez et al. [2016]. Nevertheless, it is still possible to predict these values based on inner solution set features. Similarly, a parametric study could further show the influence of λ and T_a, if any, in the approximation quality result.

Finally, we have explored some limits of the compact formulation. The approximate method could alleviate the hard instances, bringing an interesting trade-off between the error rate and the time spent by the model. The general recommendation was to run a set of trials, consider the experiment's design, and use the median value to avoid some possible outliers. It is also possible to improve the Hungarian method used in the assignment step.

This MIP-DoM approximation approach has drastically reduced the computational cost of DoM. Therefore, we believe this approximate computation method can enhance DoM's applications in more EMO/EMaO use cases.

Part III

Multi-stage reference-vector-based framework to identify efficient fronts reliably

Chapter 6

A Multi-Stage Reference-vector-based Framework

6.1 Introduction

Evolutionary algorithms are regularly being used to find a set of trade-off Pareto-optimal solutions for multi- and many-objective optimization problems. The use of reference direction or reference points is not new [Zhou et al., 2009; Deb and Jain, 2014]. Using a reference point allows the algorithm to focus the search on more 'preferred' regions, potentially saving considerable computational resources. However, even with the help of such information, there are some known shortcomings of EMO and EMaO algorithms, as discussed in Chapter 2. Such drawbacks have been recognized in the literature and still stay as open issues that require to be solved with systematic approaches.

First, since EMO and EMaO algorithms are stochastic in nature, a single application may not always produce a reproducible well-distributed and well-converged set of non-dominated (ND) solutions over multiple applications. EMO researchers often use multiple runs, each starting with a different initial population and present a median or average performing result. This is achieved by presenting the ND set obtained by the median or mean-performing hypervolume (HV) or inverse generational distance (IGD) run Ishibuchi et al. [2015a], or by using a more sophisticated attainment surface method Fonseca et al. [2005].

Second, EMO and EMaO algorithms are expected to produce a prescribed number of ND solutions, usually dictated by the chosen population size N (and/or number of reference vectors $\boldsymbol{R}$) Deb and Jain [2014]; Zhang and Li [2007], although the number of

solutions needed to adequately represent a ND set may not be known a priori. But every EMO or EMaO run may not produce the exact number of ND points as desired. This can be mainly due to two reasons. First, the stochasticity involved in an algorithm's operators may have failed to find an ND solution for every specified reference vector (RV). Second and more likely, since the exact location and shape of the efficient set are not known before a run, all specified reference vectors may not associate with an efficient solution, thereby causing less than $|\boldsymbol{R}|$ ND solutions to be found at the end of the run. These issues make a comparison between two sets of ND solutions difficult. For example, a reliable comparison using the HV metric or any uniformity measure expects both sets to have an identical number of ND solutions. Moreover, since a RV-based EMO or EMaO algorithm is designed to process $|\boldsymbol{R}|$ (usually equal to the population size N) reference vectors, fewer ND solutions in a population cause a waste of overall computational effort and memory.

Third, a set of ND solutions may indicate certain gaps in the apparent ND front discovered by an EMO or EMaO algorithm. A gap in the ND front may artificially arise from discovering fewer points in a certain area of the efficient front or a gap may truly exist in the efficient front. Usually, a gap-confirming method [Pellicer et al., 2020] with a focused EMO [Deb and Sundar, 2006] or focused EMaO [Vesikar et al., 2018] is employed to confirm the true existence of a gap. However, if the true or artificial existence of gaps can be confirmed and repaired during the optimization process, and not as a post-optimal process, the resulting EMO/EMaO application is expected to be more efficient and reliable.

Thus, although these known shortcomings are addressed with certain other additional execution of specific tasks, it would be desired if the EMO/EMaO algorithms are able to find a more reliable and reproducible ND set having exactly the desired number of a uniformly distributed set of points on the entire efficient set with a clear indication of true gaps and holes, if any.

In the recent past, EMO researchers have emphasized making a balance between convergence and diversity in finding ND solutions and proposed multi-phase EMO and EMaO algorithms. Balanced NSGA-III (B-NSGA-III) Seada et al. [2019] approach identified one of the three possible stages – extreme point identification phase, uniform diversity enforcement phase, and local improvement phase. A recent study Tian et al. [2021a] has followed three different distinct phases – convergence phase, diversity preservation phase, and convergence-diversity phase. These approaches focus on identifying gaps and holes within their obtained ND front either explicitly or through their innovative diversity evaluation performance indicators. But, none of these methods has focused on finding a pre-specified number of ND solutions, which are evaluated during

the optimization process to possess the convergence and diversity properties without any artificial holes or gaps as a collection of final solutions with a given number of solution evaluations.

This chapter attempts to address these issues by proposing a three-stage framework to improve the performance of existing evolutionary multi and many-objective optimization algorithms.

The first stage attempts to find a set of ND solutions roughly on the entire efficient set by using a standard EMO or EMaO algorithm. Based on the achieved distribution of solutions, the second stage focuses on to fill the apparent gaps and holes not covered by the first stage. In the third stage, solutions from both stages are combined, and a final EMO/EMaO algorithm is run in two iterations with a well-estimated population size to arrive at the exact desired number of ND solutions having a better convergence and uniformity. Later stages use populations obtained at the previous stages to maintain continuity and also to reduce overall computational complexity. Moreover, repeated emphases at critical regions of the search space ensure that the obtained solutions are reliably close to the true efficient front.

In the remainder of this chapter, we briefly review of the main focus in EMaO algorithms and motivate the need for a multi-stage framework involving EMaO algorithms for reliably finding a specific number of ND solutions in Section 6.2. The multi-stage framework is presented in detail next in Section 6.3 with the help of a constrained test problem and a unconstrained practical problem. Results on a number of three to 10-objective problems are presented in Section 6.4. A parametric study of a single hyper-parameter of the proposed framework is presented in Section 6.5. Finally, conclusions of this proposal are summarized in Section 6.6.

6.2 Shortcomings in identifying efficient fronts reliably

Evolutionary algorithms have a clear niche in finding multiple efficient solutions for multi- and many-objective optimization problems in a single run [Deb, 2001; Coello et al., 2002; Shukla and Deb, 2005] compared to their traditional counterparts which mostly work by finding a single efficient solution in a generational manner [Kaisa, 1999; Steuer, 1986]. In EMO and EMaO algorithms, usually an emphasis for non-dominated solutions to achieve convergence and an emphasis for diverse objective solutions to achieve a uniform distribution are established either hierarchically or simultaneously. In EMaO algorithms, a set of pre-specified normalized reference vectors (RVs) are

used to help achieve a better distribution of solutions. Despite their ever-increasing popularity in both developmental and application studies, there are a few shortcomings which we highlight here, handling of which becomes the main motivation of this multi-stage framework.

6.2.1 Reliable Convergence and Uniform Distribution of Solutions

EMO and EMaO algorithms start with an initial population and use stochastic search operators to iteratively improve two aspects – convergence towards the efficient frontier and maintain a uniform distribution of solutions in the objective space. Convergence aspect is achieved mostly by providing more selection pressure for non-dominated solutions. Diversity of solutions is achieved by various means, including a crowding measure estimating the number of solutions in the neighborhood. Original MOEA/D [Zhang and Li, 2007] proposed the diversity maintenance through a pre-defined uniform set of reference vectors in the objective space originating from the ideal point and preferring solutions that are close to RVs and also close to the ideal point. This revolutionary idea has been particularly found to work well for many-objective problems.

Some recent studies also suggested non-uniform distribution of reference vectors based on Pareto-front shapes [Ma et al., 2020], [Ishibuchi et al., 2016], [Giagkiozis et al., 2013], and [Jain and Deb, 2014]

While two aspects 'convergence first, diversity second' are undoubtedly the main emphasis of any EMO and EMaO algorithms, their order and extent of use within an algorithm are not yet settled. Some algorithms (such as, NSGA-II [Deb et al., 2002], NSGA-III [Deb and Jain, 2014], SPEA2 [Zitzler et al., 2001] and others) use a hierarchical 'convergence first, diversity second' strategy, whereas recent studies have raised questions about this strategy. Too much focus on convergence early on may lead to a loss of diversity in population members, thereby making a diversity enhancement difficult in certain problems. It is clear that a 'diversity first, convergence second' strategy may not be that useful, as although a well-distributed set of solutions is possible to achieve early on the search space far away from the efficient front, it may be difficult to maintain the diversity all along towards the convergence phase of the algorithm at the latter part of the run. However, a number of studies have proposed the need to strike a balance between convergence and diversity from start to finish during a run.

In chapter 2 some multi-stage or multi-phase algorithms was presented, such as C-TAEA [Li et al., 2019b], MOEA/D-AWA [Qi et al., 2014], B-NSGA-III [Seada

et al., 2019], CMOEA-MS [Tian et al., 2021b], CLIA [Ge et al., 2019], RVRL-EA [Ma et al., 2021], and MSEA [Tian et al., 2021a]. These approaches generally divide the optimization process into a certain number of stages and apply different selection strategies in these stages. In this way, the diversity performance of the multi-stage evolutionary algorithm is improved. Some algorithms deal with constraint problems; others are integrated only with a decomposition-based algorithm, and all of them are defined as new EMO or/and EMaO algorithm.

These existing multi-stage EMaO algorithms attempt to stress convergence, diversity, or both, whenever needed during the evolutionary process, but lack significant actions towards reinforcing the true convergence and diversity of evolving solutions. In the single-objective evolutionary algorithm literature, restarts are often employed to help the algorithm to escape from premature convergence and also to reinforce convergence to true optimal solutions. However, due to multi-faceted goals in multi-objective optimization, restarts must have different goals. From a prematurely stuck existing ND solution set, how would a restart be initiated so that convergence, diversity, or both can be achieved in the next phase, depending on what is lacking in the stuck set of population members? Such a restart strategy should not only be disruptive to the previous stuck phase and, if done properly, can address various aspects of multi-objective problem solving in a systematic manner and help produce a more reliable set of ND solutions. Our proposed multi-stage framework is a step toward this effort.

6.2.2 Fewer than Expected Number of Solutions

In an EMO or EMaO, we usually target a set (say N) of well-distributed and well-converged ND solutions. The parameter N is user-supplied. If a run produces less than N ND solutions at the end, there are at least two difficulties. First, it does not fulfill the stated requirement and second, it may be difficult to compare or statistically come up with a single performance measure for two or more competing such sets having different number of points. No existing method guarantees this aspect and most often, solutions from a big archive are chosen to find exactly N solutions. This may not produce the best possible distribution achievable with N ND points and beats the algorithmic spirit of search for N well-distributed solutions.

6.2.3 Gaps in the Obtained Non-dominated Front

Some algorithms were proposed to identify apparent gaps in obtained efficient set and employ additional tasks to fill the gaps. In Pellicer et al. [2020], a three-step algorithm

was proposed to deal with identifying and filling gaps in obtained ND fronts. In the first step, a well-distributed and converged solution set is obtained (using an EMO/EMaO algorithm). The second step tries to identify whether there are real gaps (based on inner problem characteristics) or not. Two indicators are proposed to help in this step. The third step attempts to fill the gap based on the outcome of the second step. This study only attempted to find and fill gaps, but did not focus on finding a pre-defined number of ND points, nor did the study make the steps use points from previous steps to make the overall approach more computationally efficient.

Based on the above discussions, we set our goal of arriving at a multi-stage framework involving an EMO/EMaO algorithm having the following properties:

1. A coherent and seamless three-stage algorithmic framework with a possibility of skipping second or third stages completely based on the number of well-distributed ND solutions achieved,

2. A multi-stage framework that attempts to find as close to the desired number of ND points and having as uniformly distributed as possible on the entire feasible efficient set,

3. A multi-stage framework that repeats its search among critical parts of the search space to ensure a reliable set of well-converged and well-distributed ND points, and

4. A multi-stage framework which is pragmatic in using existing seed solutions, implementable with any existing reference vector based EMO/EMaO algorithm, executed for a budget of solution evaluations, and controlled with a single well-tuned parameter.

6.3 Proposed Multi-Stage Framework

Our multi-stage framework improves the performance of a specific reference vector based EMO/EMaO algorithm (such as, NSGA-III Deb and Jain [2014] or MOEA/D Zhang and Li [2007]) to find a prescribed number of repeatable, well-converged and well-distributed set of efficient solutions (N). The chosen RV-based EMaO algorithm uses a set of reference vectors ($\boldsymbol{R}$) usually equal to the population size N, an initial seed population $\boldsymbol{P}^{\boldsymbol{seed}}$ of any size, and the preset number of solution evaluations (SEs) T_S, and produces a set of non-dominated solution set $\boldsymbol{S}$ of size N. Note also that the size of the supplied initial population $\boldsymbol{P}^{\boldsymbol{seed}}$ need not be same as N or $|\boldsymbol{R}|$. The initial

Algorithm 3: MuSt-EMaO Framework.

Input: EMaO (optimization algorithm), N (desired number of solutions), $T_S = (T_1, T_2, T_3)$ (number of solution evaluations), $\boldsymbol{P^{seed}}$ (initial population)

Output: $\boldsymbol{S}$ (a reliable and well-distributed ND solution set)

1 // **Stage 1**
2 $\boldsymbol{R_1} \leftarrow$ GenerateReferenceVectors(N);
3 $\boldsymbol{S_1} \leftarrow$ EMaO($\boldsymbol{R_1}$, $\boldsymbol{P^{seed}}$, T_1);
4 $[\boldsymbol{S_{1c}^{T1}}, \boldsymbol{R_{1\bar{c}}}] \leftarrow$ Classify($\boldsymbol{R_1}$, $\boldsymbol{S_1}$);
5 **if** $|\boldsymbol{S_{1c}^{T1}}| < N$ **then**
6 // **Stage 2**
7 $\boldsymbol{S_2} \leftarrow$ EMaO($\boldsymbol{R_{1\bar{c}}}$, $\boldsymbol{S_1}$, T_2);
8 $\boldsymbol{S_{12}} \leftarrow$ NonDominance($\boldsymbol{S_{1c}^{T1}} \cup \boldsymbol{S_2}$);
9 $[\boldsymbol{S_{3c}^0}, \boldsymbol{R_{12\bar{c}}}] \leftarrow$ Classify($\boldsymbol{R_1}$, $\boldsymbol{S_{12}}$);
10 **else**
11 $\boldsymbol{S_2} = \emptyset$, $\boldsymbol{S_{3c}^0} = \boldsymbol{S_{1c}^{T1}}$;
12 $T_3 \leftarrow T_2 + T_3$;
13 **end**
14 // **Stage 3**
15 $\boldsymbol{S_3^0} \leftarrow \boldsymbol{S_1} \cup \boldsymbol{S_2}$;
16 $N_3 = N$;
17 **for** *i = 1:2* **do**
18 $N_3 \leftarrow \text{int}\Big(N(N_3/|\boldsymbol{S_{3c}^{i-1}}|)\Big)$;
19 $\boldsymbol{R_3^i} \leftarrow$ GenerateReferenceVectors(N_3);
20 $\boldsymbol{S_3^i} \leftarrow$ EMaO($\boldsymbol{R_3^i}$, $\boldsymbol{S_3^{i-1}}$, $T_3/2$);
21 $\boldsymbol{S_3^i} \leftarrow$ NonDominance($\boldsymbol{S_3^{i-1}} \cup \boldsymbol{S_3^i}$);
22 $[\boldsymbol{S_{3c}^i}, \boldsymbol{R_{3\bar{c}}^i}] \leftarrow$ Classify($\boldsymbol{R_3^i}$, $\boldsymbol{S_3^i}$);
23 **end**
24 **if** $|\boldsymbol{S_{3c}^2}| < N$ **then**
25 $\boldsymbol{S_{3c}^2} \leftarrow$ NonDominance($\boldsymbol{S_{3c}^1} \cup \boldsymbol{S_{3c}^2}$);
26 **end**
27 $\boldsymbol{S} \leftarrow$ Reduce($\boldsymbol{S_{3c}^2}$, N);

population generation operator creates exactly $N = |\boldsymbol{R}|$ population members at the first generation by the EMaO algorithm's operators from $\boldsymbol{P^{seed}}$. The proposed MuSt-EMaO uses three stages, dividing T_S SEs among them satisfying $T_S = T_1 + T_2 + T_3$, where T_s is the number of SEs for the s-th stage. The pseudo-code of the proposed MuSt-EMaO is presented in Algorithm 3.

Stage 1 of MuSt-EMaO framework makes the first attempt to generate an evenly distributed ND solution set using an existing reference vector based approach (the baseline EMaO algorithm). We apply the Riesz s-energy method discussed in Blank et al. [2021] to first create exactly N reference lines ($\boldsymbol{R_1}$) by *GenerateReferenceVectors* operator in Line 2 and execute the EMaO algorithm for a maximum of T_1 SEs (Line 3).

EMaO starts by creating an initial population of size N from the supplied $\boldsymbol{P^{seed}}$ of any size. Applying the baseline EMaO algorithm, we obtain the first non-dominated solution set ($\boldsymbol{S_1}$ of size N_1 ($\leq N$)). The next task in Stage 1 is to classify the reference vector set $\boldsymbol{R_1}$ into two classes: (i) active reference vector (ARV) set and inactive reference vector (IRV) set by using the proposed *Classify* operator. This operator first associates every member of $\boldsymbol{S_1}$ with a reference line based on the shortest normalized Euclidean distance (identical to the d_2 operator in MOEA/D [Zhang and Li, 2007] or NSGA-III [Deb and Jain, 2014]). Then, all reference vectors for which there is an associated member from set $\boldsymbol{S_1}$ are saved in the ARV set. Let us say that the solution set $\boldsymbol{S_{1c}^{T1}}$ ($\subset \boldsymbol{S_1}$) is associated with the ARV set $\boldsymbol{R_{1c}}$. The remaining ND solutions of $\boldsymbol{S_1}$ are saved as $\boldsymbol{S_{1\bar{c}}} = \boldsymbol{S_1} \setminus \boldsymbol{S_{1c}^{T1}}$ and the corresponding reference vectors are saved in the IRV set as $\boldsymbol{R_{1\bar{c}}} = \boldsymbol{R_1} \setminus \boldsymbol{R_{1c}}$. Note that the IRV set indicates the region on the unit simplex for which no ND solutions are found in Stage 1.

If the cardinality of $\boldsymbol{S_{1c}^{T1}}$ is less than the number of desired solutions (N), then the MuSt-EMaO framework goes to Stage 2 (Line 5), in which a separate EMaO is run to find ND solutions for IRVs from the Stage 1 using $R_{1\bar{c}}$. Solutions from Stages 1 and 2 are then combined, *NonDominance* operator finds the ND set in Line 8, and the resulting ND set is classified to find associated points for the original R_1 RVs, generating $\boldsymbol{S_{3c}^{0}}$.

On the other hand, if Stage 1 is able to find an associated point for each R_1 RV, Stage 2 is omitted, and remaining T_2 and T_3 SEs are allocated to Stage 3.

Next, the MuSt-EMaO framework moves to Stage 3, irrespective of whether the total number of ND points obtained from combined Stages 1 and 2 is still less than N, or equal to N. In case of former, Stage 3 creates new ND points by increasing the number of RVs beyond N, so that N points are found by Stage 3 EMaO execution, and in case of latter, Stage 3 uses the remaining SEs (T_3) with N RVs to improve the obtained solutions. The first task in Stage 3 is to estimate the number of required reference lines for this purpose. We suggest linearly scaling the population size by making the argument that if N reference vectors have produced $\boldsymbol{S_{3c}^{0}}$ ND points in the combined Stages 1 and 2, how many reference vectors are needed to produce N ND points? We calculate this estimate as N_3 in Line 18. The first iteration of Stage 3 applies the same EMaO algorithm with a combined population $\boldsymbol{S_1}$ and $\boldsymbol{S_2}$ of all ND solutions found in Stages 1 and 2 as an initial population. The EMaO algorithm is run for N_3 number of reference vectors (set $\boldsymbol{R_3^1}$ created in Line 19) for half of the allocated SEs for Stage 3. This produces a new ND set $\boldsymbol{S_3^1}$ in Line 20. The associated ND points are saved in set $\boldsymbol{S_{3c}^{1}}$. In order to not leave out any previously obtained better distributed ND points, $\boldsymbol{S_{3c}^{1}}$ is combined with $\boldsymbol{S_{3c}^{0}}$, and a better set $\boldsymbol{S_3^1}$ of size of $|\boldsymbol{R_3^1}|$

is chosen in Lines 21-22. Since at least N RVs are used in Stage 3, it is likely that two iterations will find N or more ND points at the end of Stage 3.

There is still a remote possibility that we are not able to have more than or equal N points. Line 24-25 treats this case. Solutions from $\boldsymbol{S^1_{3c}}$ and $\boldsymbol{S^2_{3c}}$ are combined, the *NonDominance* operator is used, generating a new $\boldsymbol{S^2_{3c}}$.

Finally, $\boldsymbol{S^2_{3c}}$ set is then reduced to have exactly N final points in Line 27.

6.3.1 An Illustration

We illustrate the working of the proposed MuSt-EMaO framework with NSGA-III through a simulation on three-objective C2-DTLZ2 constrained problem. Figure 6.1(a) shows the active reference point set on the unit simplex by '+' that have an associated ND point from the final population of NSGA-III algorithm run with $N = |\boldsymbol{R}| = 100$ reference points and other standard parameter settings [Deb and Jain, 2014]. The respective ND objective vectors are shown with circles in Figure 6.1(b). A total budget of solution evaluations of $T_S = 20,000$ is divided as $T_1 = T_2 = 5,000$ and $T_3 = 10,000$. It is clear that each ARV has at least one ND solution assigned to it. For this problem, Stage 1 is able to find $|\boldsymbol{S^{T1}_{1c}}| = 65$ ND solutions. Each RV in the IRV set (35 RVs represented with a '×') does not have an ND solution associated with it in Stage 1. This can be a failure on the part of the NSGA-III algorithm or the C2-DTLZ2 problem does not possess any true efficient point associated with the IRVs. This is intended to be verified at Stage 2 by executing NSGA-III again with a special focus in attempting to find efficient solutions corresponding to the IRV set only. This makes the MuSt-EMaO framework more reliable than a single algorithmic run with a focus on all RVs on the entire unit simplex.

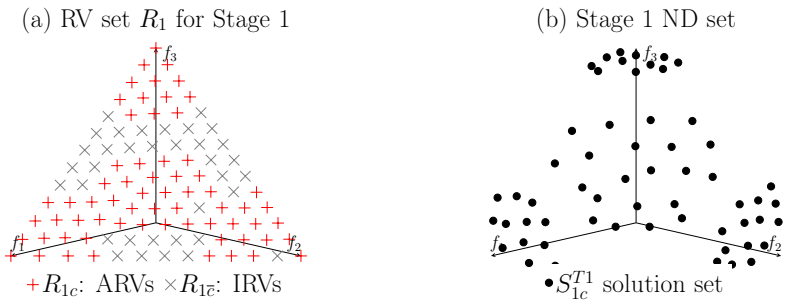

Figure 6.1. Stage 1 of MuSt-EMaO framework is illustrated on C2-DTLZ2 problem. Of $|\boldsymbol{R_1}| = 100$ RVs, 65 ND points are obtained with $T_1 = 5,000$ SEs in Stage 1.

In Stage 2, the same EMaO algorithm is applied with the Stage 1 ND set $\boldsymbol{S_1}$ as the initial population, but only with the inactive reference vector set $\boldsymbol{R_{1\bar{c}}}$. This allows to focus the search on the part of the efficient front left out at Stage 1 and finds a

new ND set $\boldsymbol{S_2}$ in Line 7. In the illustrative problem, $|\boldsymbol{R_{1\bar{c}}}| = 35$. Stage 2 NSGA-III execution finds $|\boldsymbol{S_2}| = 11$ new ND points that were not found in Stage 1, as shown in Figure 6.2. The non-dominated solutions of the ND sets of Stages 1 and 2 ($\boldsymbol{S_{1c}^{T1}}$ and $\boldsymbol{S_2}$) are identified and stored in $\boldsymbol{S_{12}}$ in Line 8. 65 and 11 (or $|\boldsymbol{S_{12}}| = 76$) ND solutions are shown in Figure 6.2(b) with solid and open circles, respectively. The respective ARV points (for Stage 2 solutions are shown in Figure 6.2(a) with a '+' symbol. This ND set is then classified using the original entire reference set $\boldsymbol{R_1}$ to identify the associated points $\boldsymbol{S_{3c}^0}$ and the IRV set $\boldsymbol{R_{12\bar{c}}}$ in Line 9. An explicit focus in undiscovered region provides the reliability of the proposed framework, as depicted in the figure. The total number of ND points found after Stages 1 and 2 are $|\boldsymbol{S_{3c}^0}| = 76$ for the C2-DTLZ2 example.

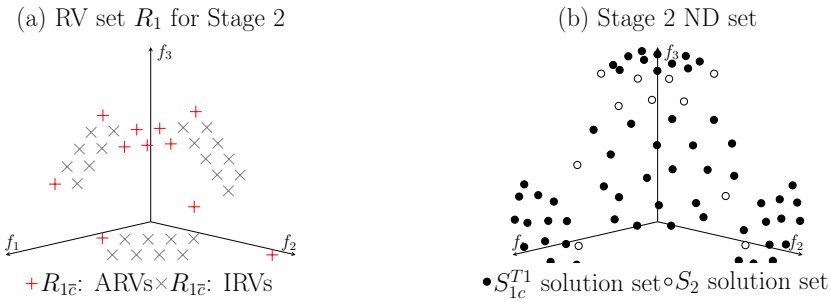

Figure 6.2. Stage 2 of MuSt-EMaO framework is illustrated on C2-DTLZ2 problem. 11 new ND points are found in Stage 2, making a total of 76 points (out of 100 original RVs) obtained with $T_1 + T_2 = 10,000$ SEs.

The first iteration of Stage 3 estimates that $N_3 = int\,(100(100/76)) = 131$ reference vectors are needed. Figures 6.3(a) and (b) show that 96 ND solutions are found. These points are denser but are all within the feasible region of the efficient set. However, the number of obtained points is close, but still not equal to the desired number $N = 100$.

The above linear estimation procedure for an increased number of reference vectors $\boldsymbol{R_3^1}$ does not require any complicated information on the region where more RVs are required to be added, as proposed in the adaptive NSGA-III procedure Jain and Deb [2014]. More RVs are simply placed on the entire unit simplex. In the second iteration, the required number of RVs is estimated to be $N_3 = int\,(100(131/96)) = 136$. Figures 6.3(c) and (d) show the obtained 100 ND points (equal to the desired number of $N = 100$ points), which are slightly more packed compared to iteration 1 of Stage 3. It is clear that without the iterative process, it would have been difficult to predict exactly how many RVs were required to be chosen the desired number of efficient points.

One can argue that instead of going through two iterations within Stage 3, one

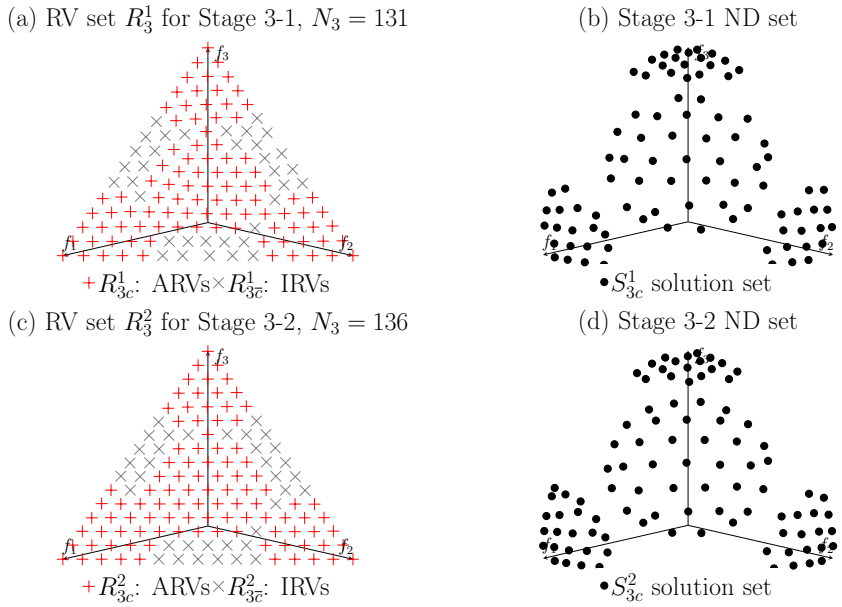

Figure 6.3. Two iterations of Stage 3 of MuSt-EMaO framework are illustrated on the C2-DTLZ2 problem. 20 more ND points are found on Stage 3, Iteration 1 ((a) and (b)), making a total of 96 points, and, finally, 4 more points are found on Stage 3, Iteration 2 ((c) and (d)), making 100 points at the end. Both iterations of Stage 3 use $T_3 = 10,000$ SEs.

can double or triple the number of RVs and make a single iteration in Stage 3 and then use the reduce operator to adjust the ND points to N. This alternate approach is not viable in a few ways. First, as argued above, it is difficult to know beforehand how many RVs are needed to produce N well-distributed ND points. Second, there is waste of computational effort in dealing with unnecessarily large RVs. Our two-iteration approach in Stage 3 makes a good compromise among the naive larger-than-required RV approach and many meticulous iterations to slowly reach the desired number of RVs.

Note that the proposed MuSt-EMaO framework does not demand too many critical parameters. The N is the desired number of ND solutions and is not a parameter. The total SEs, T_S, is also not a parameter, but two of the three SEs: T_1, T_2, and T_3 are the only required parameters to be set. In this sense, we intend to have only one parameter: γ is the proportion of T_S to be set for T_3: $T_3 = \gamma T_S$. Therefore, $T_1 = T_2 = \frac{1-\gamma}{2} T_S$. Since that Stage 3 needs to find denser points, we suggest that Stage 3 should take the lion's share of T_S. We perform a parametric study in Section 6.5 to suggest a suitable value of γ.

6.4 Results

In this section, first, a set of computational tests is designed aiming to assess the performance of the proposed multi-stage approach with NSGA-III [Jain and Deb, 2014] as an EMaO algorithm. In this case, well-known quality indicators are employed, and the MuSt-NSGA-III is compared with the baseline NSGA-III approach on a set of benchmark problems. Thereafter, the general applicability of the proposed MuSt-EMaO approach is tested on other EMaO algorithms. In this case, our goal is to investigate if the performance of different reference-based EMaO algorithms can be improved by the multi-stage approach proposed in this study.

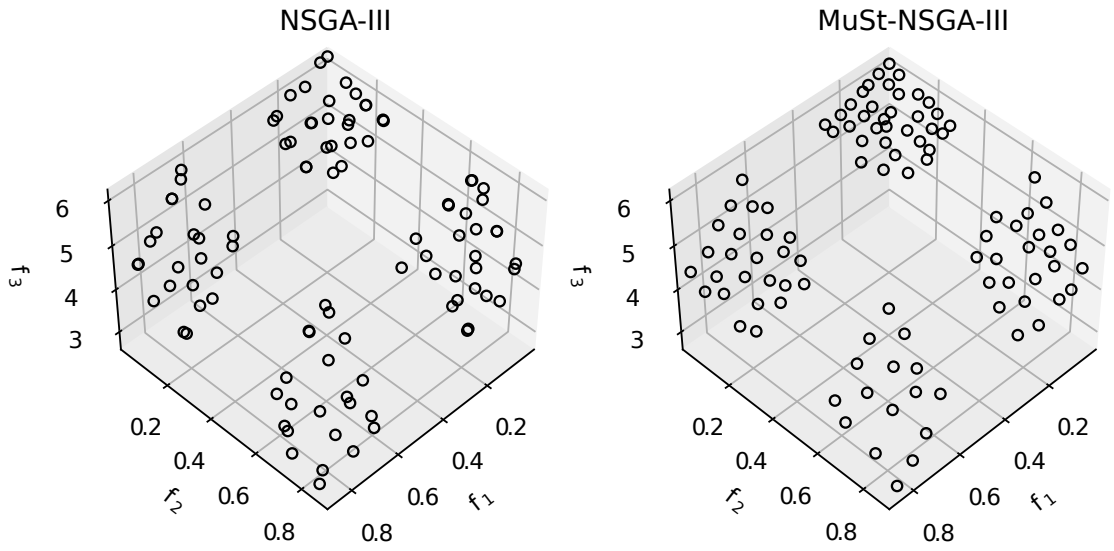

Figure 6.4. NSGA-III and MuSt-NSGA-III results for the median HV run for MaF07 problem show a better distribution obtained by the latter with identical SEs.

6.4.1 Proof-of-Principle Results of MuSt-NSGA-III Approach

First, we integrate the proposed MuSt-EMaO framework with NSGA-III and attempt to solve two-objective and three-objective problems to gain insights into the working principle of the proposed multi-stage framework.

To demonstrate the distribution of final ND points obtained from NSGA-III and MuSt-NSGA-III algorithms, we plot the efficient points from the median performing runs for the three-objective MaF07 problem in Figure 6.4. In all cases, N is set to 100. Both algorithms are implemented using the pymoo framework Blank and Deb [2020a].

In all tests, both algorithms are executed with $T_S = 20,000$ for all problems, For MuSt-NSGA-III, we use $T_1 = T_2 = 5,000$, so that $\gamma = 1/2$. Other NSGA-III parameters are set to their standard settings: $p_c = 1$, $p_m = 1/n$, $\eta_c = 30$, and $\eta_m = 20$. It is clear that MuSt-NSGA-III is able to find a better distributed set of efficient points compared to NSGA-III. This visual comparison is aided with quantifiable quality indicators in Table 6.1.

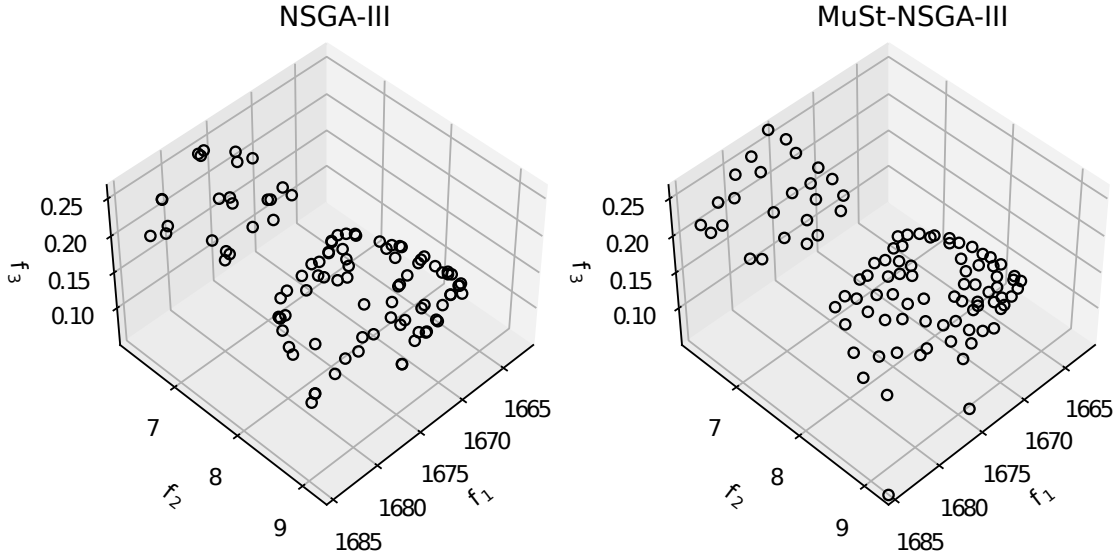

Figure 6.5. NSGA-III and MuSt-NSGA-III sample results for the crashworthiness problem. 50 runs are executed and the ND points from the median HV run are plotted for each case.

There are exactly 100 ND points spread almost uniformly on the four clusters of the efficient front. Despite many reference vectors in the original set $\boldsymbol{R_1}$ not conforming to any efficient solution, our MuSt-NSGA-III is able to find more active reference vectors in Stage 3 and provide a well-distributed efficient point set with the same number of SEs ($T_S = 20,000$) as the baseline algorithm.

Another example is the crashworthiness problem depicted in Figure 6.5. The non-dominated solutions from the median HV over the 50 runs are shown. It is clear from the figure that MuSt-NSGA-III procedure is able to find a better distribution with more solutions. Similar behaviour is also observed for other problems, supporting the superiority of the proposed multi-stage approach from the five uniformity indicators used here.

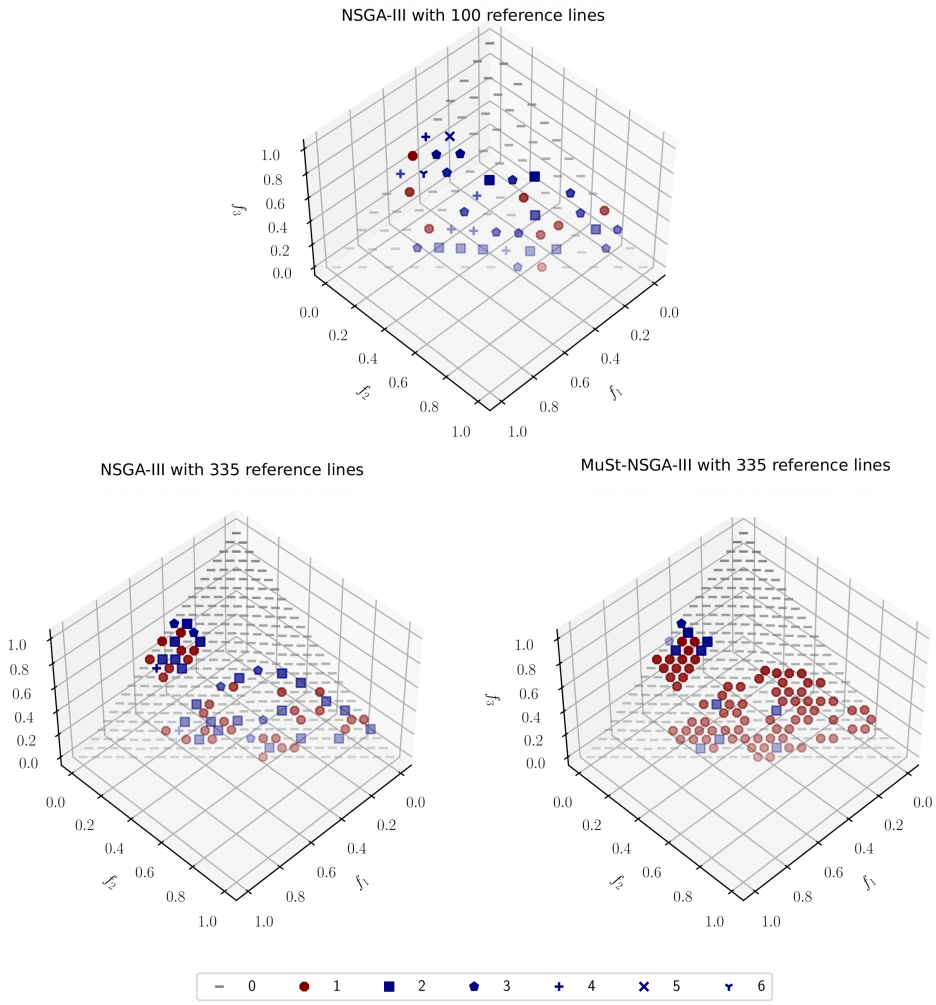

Figure 6.6. Active RVs from NSGA-III on unit simplex with 100 and 335 reference points, respectively, at the top and bottom-left. Associated RVs for MuSt-NSGA-III ND solutions are shown at the bottom-right, which uses the same 335 reference points on the entire unit simplex. Each symbol on the plots indicates the number of solutions assigned to each RV. MuSt-NSGA-III finds more active RVs

Clearly, with 100 RVs on the entire unit simplex in NSGA-III, only 38 RVs are found to have associated population members. No matter how many generations are executed, this number will not change much, but the objective vectors may come closer to the RVs making the overall distribution better with generations. However, NSGA-III does not put any specific emphasis in arranging the distribution of other ND solutions, which are not the closest associated RV solutions. Thus, locations of these additional ND solutions are not expected to make the overall distribution uniform. The top plot in Figure 6.6 marks each of the active reference points on the unit simplex. First, notice that there are many inactive reference points on the simplex, shown with a '−'. Second, notice from the shape of the markers indicating the number of ND population

members that are associated with each ARV is that some reference points have more associated population members and some are not. All 100 ND points obtained by NSGA-III are also shown in left plot in Figure 6.5, clearly depicting the non-uniform density of points. This is expected to happen to problems in which not all RVs of the unit simplex are expected to have an associated point.

However, in Iteration 2 of Stage 3, 335 RVs were estimated and used in the median HV run. The left plot in Figure 6.6 shows all 335 reference points on the unit simplex and associated reference points are marked with different markers according to the number of neighboring ND population members associated with them. Now, 59 of 335 reference points are active. However, when the active reference points are marked with MuSt-NSGA-III ND points, there are 87 reference points with one population member, 9 reference points with two population members and two reference points with 3 population members found. This makes a total of 100 points having a much closer to uniform distribution than NSGA-III points. Since our density adjustments are made on the entire unit simplex uniformly by increasing the number of reference points, the uniformity in distributed points is also expected to be better on the efficient frontier.

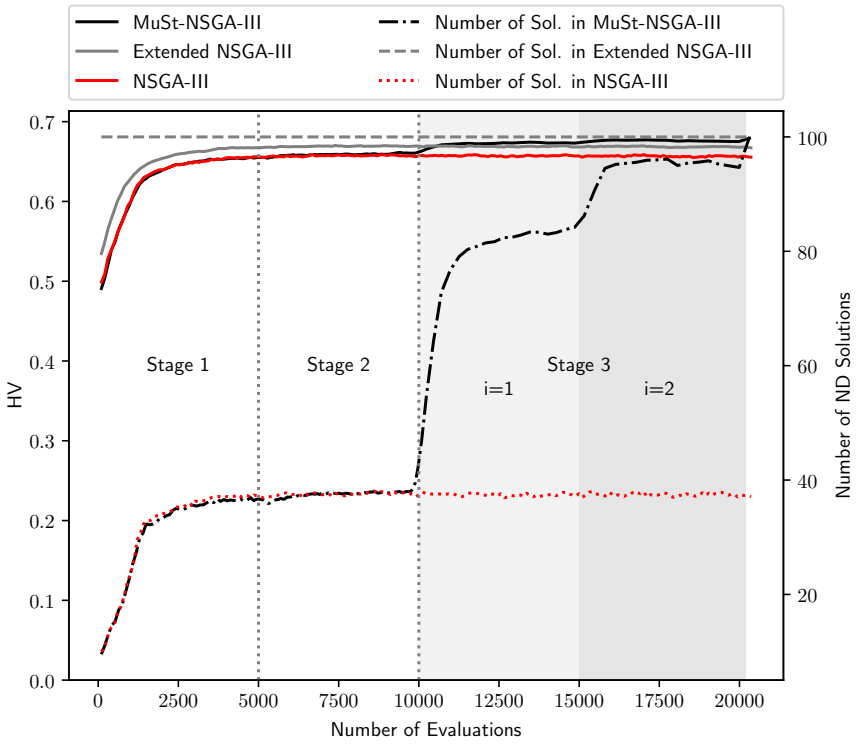

Figure 6.7. Average hypervolume (solid lines) and the number of active RVs (dotted lines) obtained using NSGA-III (calculated from two sets, closest for each ARV and all ND) and MuSt-NSGA-III for the crashworthiness problem.

A convergence behavior through HV can give us another perspective of how the three stages of our proposed framework work compared to the baseline algorithm. Figure 6.7 shows the average HV variation over a number of SEs from 50 runs for three different cases of HV computation executed after every generation of each algorithm with (i) the closest MuSt-NSGA-III ND population member to each associated RV, marked as 'MuSt-NSGA-III', (ii) all NSGA-III ND population members, marked as 'Extended NSGA-III', and (iii) the closest NSGA-III ND population member to each associated RV, marked as 'NSGA-III'. The number of such population members are also shown with values shown on the right vertical axis. For MuSt-NSGA-III, the final population from the previous stages and the current population are combined to find associated active points, and HV is computed from them. Thus, a solution evaluation-wise HV variation is expected to be mostly monotonic, as can be seen in Figure 6.7. It can also be seen that very early on, NSGA-III is able to find 100 ND population members (shown by the gray dashed line). NSGA-III and MuSt-NSGA-III find a similar number of active RVs until about 10,000 SEs (until the end of Stage 2). This is not surprising because both these algorithms employ the same baseline NSGA-III until then, and Stage 1 is able to capture most of the active RVs, and Stage 2 does not add many new ARVs. However, due to the increase in total RVs using our proposed estimation procedure, the number of ARVs drastically increase in Iteration 1 of Stage 3 and some more in Iteration 2. Stage 1 was saturated with 38 ARVs (our of 100 RVs), and without these iteration-wise increases in RVs, as in original NSGA-III shown with dotted line, the algorithm would not have increased the number of ARVs close to the desired number ($N = 100$). It is also clear that Iteration 2 of Stage 3 is not able to produce exactly 100 ARVs, but Line 24 to 27 of Algorithm 3 allows a combination of both iteration's points together to produce exactly 100 solutions at the end.

The HV variations reveal that NSGA-III and MuSt-NSGA-III perform almost in the same manner during Stages 1 and 2. However, MuSt-NSGA-III improves the HV by adding more ND points in both iterations of Stage 3, whereas NSGA-III's HV does not improve due to lack of denser active ND solutions in the population. But, if all ND solutions are used to compute HV, marked with 'Extended NSGA-III', a slight increase in HV from the HV computed with single active ND point, marked with 'NSGA-III' is observed. The HV for 'MuSt-NSGA-III' is better due to a more uniformly distributed set of increasingly denser points obtained by our multi-stage framework.

The above deeper analysis of the behavior of the proposed MuSt-NSGA-III compared to the original NSGA-III makes a better understanding of the working principle of the proposed approach. Furthermore, it provides us with confidence to apply the approach to solve other problems and integrate with other EMaO algorithms.

6.4.2 MuSt-NSGA-III Applied to Multi-objective Test Problems

Next, we apply MuSt-NSGA-III to a number of multi-objective test problems. We use some of the performance quality indicators discussed in Chapter 2:

- *Hypervolume* performance quality indicator with nadir point as the reference point. Higher the HV value, better is the set in terms of convergence and diversity;
- *MIP-DoM* as a binary quality indicator. Smaller the value, better is the approximation set;
- *k-th nearest neighbor* as a naive uniformity of ND solutions measure. We use the mean and standard deviation of the ($k = \lfloor\sqrt{M-1}\rfloor$) values of ND solutions. A large value of mean and a small value of standard deviation are desired;
- *Spacing* (SP) measures how uniform the ND solutions are according to the standard deviation of Euclidean distance from their nearest neighbors. Smaller spacing values indicate better distribution;
- *Uniform Distribution* (UD) is a niching-based unary quality indicator that measures the standard deviation in niche count values of solutions. Larger the metric, more uniform is the distribution. As discussed in Chapter 2 there is a user-specified parameter, it is the σ_{share}. In all simulations of this study, we have set σ_{share} as 20% of the maximum nearest neighbor distance of reference points on the unit simplex;
- Finally, we use the *Evenness* (ξ) as an indicator to measure gaps, if any, by a coefficient of variance measure of two critical neighbor distances of all ND points. Smaller the value, the better is the evenness.

Using the above quality indicators, the computational experiments are designed to assess the performance of the MuSt-NSGA-III for comparison to its baseline version in 12 multi-objective problems. ZDT3 and MaF07 present disjoint efficient sets having natural gaps in their respective efficient fronts. Crashworthiness and car side-impact problems are practical problems and involve non-linearity and non-convexity features of the efficient front. Other problems from the MaF benchmark suite (designed for CEC'2018 MaOP competition) are selected: MaF01, MaF10, MaF11, and MaF12, which present linear, non-separable, biased, convex/disconnected, and concave features. A few DAS-CMOP family problems having multiple disjointed clusters in their efficient

fronts are also selected. For DAS-CMOP7 to DAS-CMOP9 problems, the difficulty triplets are set to (0.5,0.5,0.5).

Table 6.1 presents the average quality indicator values obtained by NSGA-III and MuSt-NSGA-III over 50 runs of both algorithms on 12 problems. In all cases, N is set to 100. In all tests, MuSt-NSGA-III is executed with $T_S = 20,000$ for all problems, except for DAS problems, for which we use $T_S = 100,000$. For all problems, $\gamma = 1/2$ is chosen. The Wilcoxon signed-rank test with a significance level of 0.05 is applied to assess the statistical significance of the obtained values. Data in italics means that the p-value is more than 0.05, and it is not possible to reject the null hypothesis that the distribution of the differences between NSGA-III and MuSt-NSGA-III is symmetric at about zero. Data in bold indicates the best value between the approaches.

It is important to observe that in some problems, the base algorithm is not able to generate N solutions as pre-specified, mainly due to the non-existence of an efficient solution for every chosen RV. To make a fair comparison with our multi-stage framework, and we use an extended solution sets which are ND solutions but are not necessarily the closest point to each RV. This makes the total number of ND points equal to $N = 100$, which is also set in MuSt-NSGA-III.

Table 6.1. Experimental results of baseline NSGA-III with extended ND solution set and MuSt-NSGA-III. Seven performance indicators are used to statistically compare both algorithms over 50 runs. Symbols ↑ and ↓ indicate better metric values for large and small values, respectively.

Problem	*Algorithm*	*M*	*N*	*HV*		*MIP-DoM* ↓	*k-neighbors distance*		*SP* ↓	*UD* ↑	ξ ↓	*#RVs* Stage 3, i=2
				mean ↑	*std dev* ↓		*mean* ↑	*std dev* ↓				
ZDT3	NSGA-III	2	100	*0.4806*	*0.0038*	**0.1086**	0.0071	0.0048	0.0068	0.6983	*1.6641*	-
	MuSt-NSGA-III		100	***0.4816***	***0.0006***	0.1594	**0.0083**	**0.0028**	**0.0040**	**0.9634**	***1.6300***	177
DTLZ2	NSGA-III	3	100	***0.7917***	***0.0000***	***0.7201***	***0.1049***	***0.0034***	***0.0052***	***1.0000***	***0.0858***	-
	MuSt-NSGA-III		100	*0.7916*	*0.0002*	*0.7253*	*0.1031*	*0.0037*	*0.0061*	*0.9975*	*0.0902*	100
MaF01	NSGA-III	3	100	0.0146	0.0002	1.7711	0.0175	0.0147	0.0227	0.5329	0.8066	-
	MuSt-NSGA-III		100	**0.0162**	**0.0001**	**1.7393**	**0.0439**	**0.0073**	**0.0102**	**0.9748**	**0.2392**	343
MaF07	NSGA-III	3	100	0.2503	0.0150	*1.4809*	0.0254	0.0179	0.0279	0.6400	0.9236	-
	MuSt-NSGA-III		100	**0.2551**	**0.0092**	***1.4508***	**0.0498**	**0.0133**	**0.0190**	**0.8781**	**0.6991**	222
MaF10	NSGA-III	3	100	0.5153	0.0893	***1.0939***	0.0290	0.0171	0.0266	0.5922	*0.8987*	-
	MuSt-NSGA-III		100	**0.6177**	**0.0289**	*1.2600*	**0.0500**	**0.0141**	**0.0218**	**0.9359**	***0.7319***	187
MaF11	NSGA-III	3	100	*0.7273*	*0.0635*	*1.5877*	*0.0855*	0.0198	0.0282	0.9030	0.2285	-
	MuSt-NSGA-III		100	***0.7371***	***0.0614***	***1.5133***	***0.0868***	**0.0146**	**0.0204**	**0.9868**	**0.2098**	106
MaF12	NSGA-III	3	100	*0.6640*	***0.0019***	***2.2017***	0.0837	0.0192	0.0272	0.9049	0.2311	-
	MuSt-NSGA-III		100	***0.6646***	*0.0020*	*2.2286*	**0.0858**	**0.0141**	**0.0194**	**0.9951**	**0.2119**	102
Carside impact	NSGA-III	3	100	0.2288	**0.0005**	***1.9857***	0.0506	0.0334	0.0498	0.6919	0.4876	-
	MuSt-NSGA-III		100	**0.2303**	0.0009	*2.0307*	**0.0733**	**0.0118**	**0.0163**	**0.9671**	**0.2117**	143
Crashworthiness	NSGA-III	3	100	0.1101	0.0027	1.4844	0.0195	0.0166	0.0260	0.5440	0.9333	-
	MuSt-NSGA-III		100	**0.1139**	**0.0019**	**1.0682**	**0.0403**	**0.0112**	**0.0169**	**0.7991**	**0.6205**	325
DAS-CMOP7	NSGA-III	3	100	***0.7331***	*0.0074*	*0.9922*	*0.0778*	0.0244	0.0359	0.8402	0.3194	-
	MuSt-NSGA-III		100	*0.7330*	***0.0060***	***0.9918***	***0.0794***	**0.0158**	**0.0237**	**1.0000**	**0.2881**	109
DAS-CMOP8	NSGA-III	3	100	0.7148	0.0065	***0.9917***	***0.0837***	*0.0199*	0.0292	0.9017	***0.2955***	-
	MuSt-NSGA-III		100	**0.7156**	**0.0027**	*0.9963*	*0.0834*	***0.0161***	**0.0239**	**0.9975**	*0.2974*	111
DAS-CMOP9	NSGA-III	3	100	*0.2744*	*0.0513*	***0.4712***	0.0139	*0.0135*	0.0209	*0.4625*	0.9744	-
	MuSt-NSGA-III		100	***0.2912***	***0.0509***	*0.5330*	**0.0194**	***0.0122***	**0.0183**	***0.4766***	**0.8128**	396

The Wilcoxon signed-rank test with a significance level of 0.05 was applied to perform a statistical comparison between the approaches. Thus, the data in italics means that it is not possible to detect differences between NSGA-III and MuSt-NSGA-III. Data in bold indicates the best value between the approaches.

Analyzing Table 6.1, some important features of the proposed multi-stage approach can be highlighted. First, we argue that the MuSt-EMaO maintains the inner characteristics of the baseline reference-based algorithm. If the problem allows finding all N solutions at Stage 1, like in ZDT3 and DTLZ2 problems, MuSt-NSGA-III and the baseline NSGA-III perform precisely the same way. In three-objective DTLZ2, there is no hole or clusters in the efficient front. Then, only Stage 1 of the multi-stage approach is executed for the entire T_S SEs. Hence, both baseline NSGA-III and MuSt-NSGA-III are identical algorithms. The table shows that all quality indicators present no statistical difference among the results. This degeneracy of our proposed algorithm to the baseline algorithm was a desired feature.

Second, observing the hypervolume values, it is possible to observe that the MuSt-NSGA-III always produce a larger hypervolume value than its baseline version, except for DAS-CMOP7. However, the statistical test indicates that there is no statistical difference between the algorithms on this problem. However, MuSt-NSGA-III is statistically better on MAF01, MaF07, car side-impact, crashworthiness, DAS-CMOP8 and DAS-COMP9 problems. For the MIP-DoM metric, except in ZDT3, MuSt-NSGA-III is statistically better or equivalent to NSGA-III on all 11 problems. All other indicators capture the reliability and uniformity of the ND solution set. MuSt-NSGA-III clearly performs better than the baseline algorithm considering all the indicators. The only exceptions are the k-nearest neighbor distance mean and Evenness indicators for DAS-CMOP8 problem; however, the tests indicate no statistical difference between their values.

Third, the standard deviation on HV shows that, in most problems, the variation of HV over 50 independent runs has a much smaller standard deviation compared to NSGA-III, meaning that the performance of MuSt-NSGA-III runs produce almost similar set of 100 ND points. The coefficient of variation (standard deviation of HV divided by the mean HV) varies between 0.0000 and 0.1748 with a median of 0.0072, while NSGA-III values vary between 0.0001 to 0.187 with a median of 0.0119.

Fourth, the last column of the table presents the number of RVs needed in Iteration 2 of Stage 3 of MuSt-NSGA-III to obtain $N = 100$ ND solutions. As it can be seen, about four times more RVs than the required ND solutions are needed. For the DTLZ5 problem shown in the Appendix A, about 19 times more RVs are needed. These numbers are difficult to predict at the beginning and our proposed multi-stage approach adaptively estimates a requisite number of RVs needed to find the desired number of ND points.

6.4.3 Multi-stage Framework Applied to Other EMaO Algorithms

The computational results so far were achieved using the NSGA-III as the baseline reference-vector based algorithm. However, the multi-stage approach can also be integrated with any other reference-based EMaO as long as a requirement is met. Since in the multi-stage approach, the initial population from Stage 1 must be used in Stage 2, and in Stage 3, the baseline algorithm must allow the user to set an initial population instead of always a random one. Three additional reference-based algorithms are included in this study: MOEA/D [Zhang and Li, 2007], C-TAEA [Li et al., 2019b], and CLIA [Ge et al., 2019]. The algorithms have been implemented using pymoo and PlatEMO frameworks, [Blank and Deb, 2020a; Tian et al., 2017].

Table 6.2. MaF01 problem set with $M = 3$. The experiment involves 50 trials, and each algorithm is run with $T_S = 20,000$. Some selected quality indicators are tabulated for our discussion here.

Algorithm	***N***	***HV*** ↑	***MIP-DoM*** ↓	***k-neighbors distance***		***SP*** ↓	***UD*** ↑	ξ ↓
				mean ↑	***std dev*** ↓			
NSGA-III	100	0.0146	1.7711	0.0175	0.0147	0.0227	0.5329	0.8066
MuSt-NSGA-III	100	**0.0162**	**1.7393**	**0.0439**	**0.0073**	**0.0102**	**0.9748**	**0.2392**
MOEA/D	97	0.0129	1.7946	0.0261	0.0269	0.0409	0.4709	0.8589
MuSt-MOEA/D	100	**0.0161**	**1.7473**	**0.0424**	**0.0085**	**0.0122**	**0.9282**	**0.2808**
C-TAEA	100	0.0153	***1.7449***	0.0289	0.0130	0.0205	0.5552	0.5800
MuSt-C-TAEA	100	**0.0162**	*1.7770*	**0.0450**	**0.0079**	**0.0108**	**0.9646**	**0.2233**
CLIA	100	***0.0164***	1.8025	0.0345	0.0128	0.0198	0.7148	0.4338
MuSt-CLIA	100	***0.0164***	**1.7332**	**0.0450**	**0.0085**	**0.0110**	**0.9687**	**0.2318**
CLIA	100	*0.0163*	1.8537	0.0344	0.0128	0.0199	0.7123	0.4323
MuSt-CC	100	***0.0164***	**1.6558**	**0.0438**	**0.0087**	**0.0120**	**0.9798**	**0.2466**

The performance quality indicators – HV, k-neighbors distance mean, k-neighbors distance standard deviation, Spacing, UD, and Evenness – are used as quality indicators. The Wilcoxon signed-rank test with a significance level of 0.05 was employed to assess the statistical significance of the values. Data in italics means the p-value is more than 0.05, and it is not possible to detect difference between the values. Data in bold indicates the best value between the multi-stage approach and its baseline.

Table 6.2 presents seven performance indicators obtained for five EMaO algorithms with and without the multi-stage approach on three-objective MaF01 problem

(similar to the inverted DTLZ1 problem). As described in Chapter 2, Section 2.1, CLIA [Ge et al., 2019] is an algorithm which combines two interactive processes: Cascade Clustering (CC) and Reference Point Incremental Learning. CC uses reference vectors to create and sort the solutions, while the SVM model adjusts these references throughout the evolutionary process. Therefore, we want to apply CLIA as a baseline algorithm in MuSt-EMaO in two different contexts.

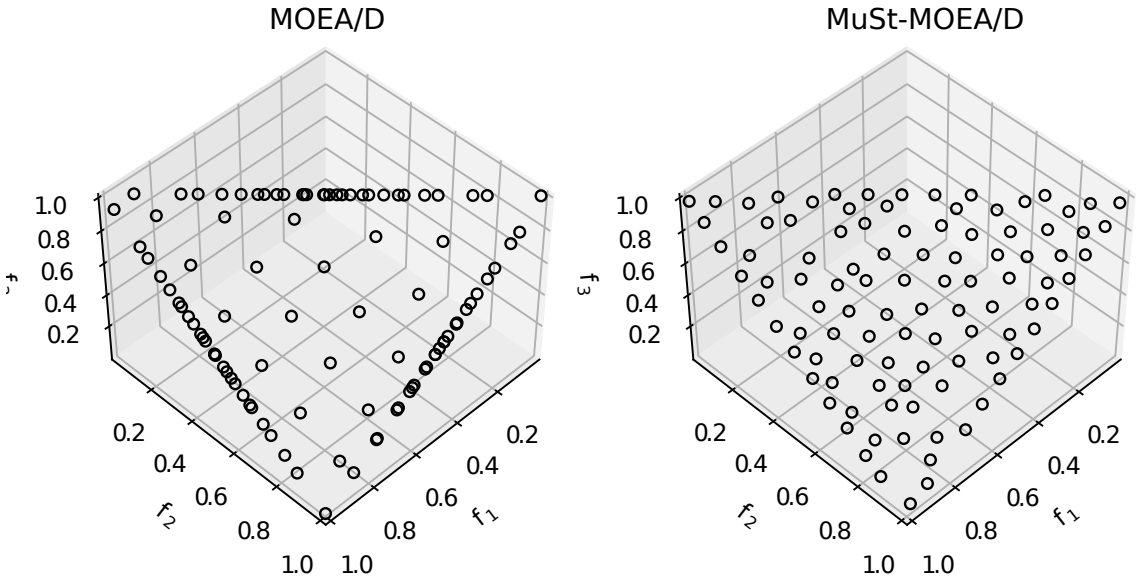

Figure 6.8. MOEA/D and MuSt-MOEA/D ND solutions from the median HV run of 50 runs for MaF01 problem.

Initially, CLIA is applied as the reference-based algorithm in MuSt-EMaO just like other EMaO algorithms (NSGA-III, MOEA/D, and C-TAEA). Thereafter, we integrate our multi-stage framework coupled with the Cascade Clustering (CC) of CLIA. The rationale behind this idea is that although the SVM model in the incremental learning phase improves the uniformity of adjusting the reference vectors, it presents a quadratic computational cost related to the number of objectives. We argue that the proposed multi-stage approach can improve the reliability and uniformity, when coupled with CC. In this context, to assess the effectiveness of the reference vector adjustments, we turn off the incremental learning approach from CLIA and integrate our MuSt-EMaO framework with the Cascade Clustering process (MuSt-CC). Interestingly, in all performance quality indicators, the respective multi-stage version has obtained statistically better or equivalent performance.

To show the distribution of obtained points, we plot MOEA/D obtained ND points in the left plot of Figure 6.8. These points come from the median HV run. The ND points from MuSt-MOEA/D are shown in the right plot. Clearly, the multi-stage procedure is able to produce a much better distribution of points on the MaF01 problem.

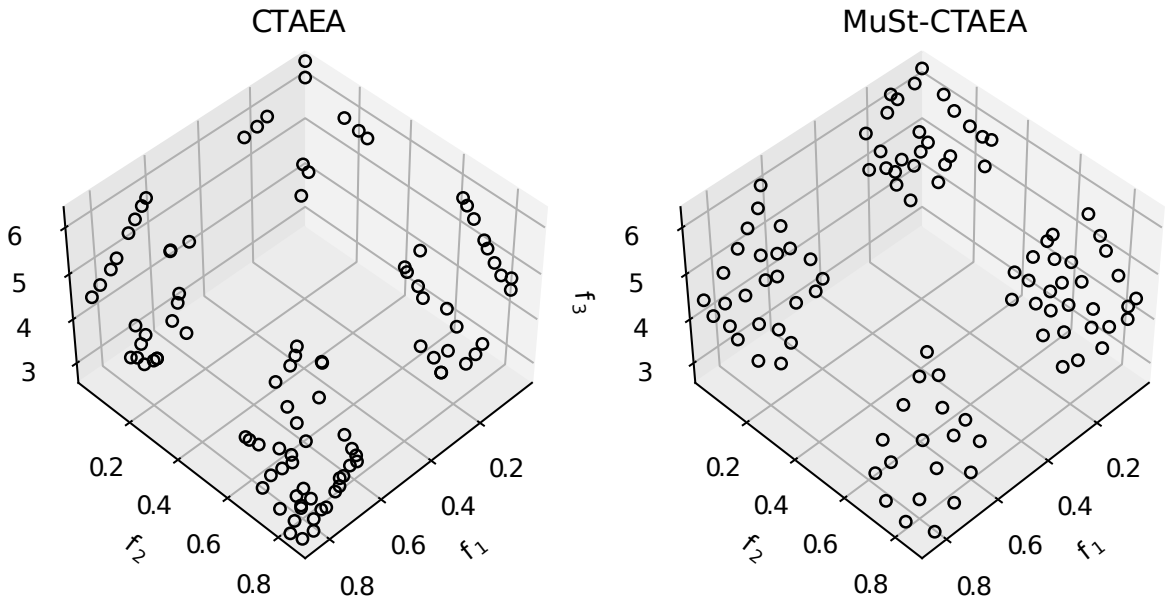

Figure 6.9. C-TAEA and MuSt-C-TAEA ND solutions from the median HV run of 50 runs for MaF07 problem set.

Note from Table 6.2 that MOEA/D could not find 100 ND points. We allow a 5% less number of ND points that desired to compute the performance indicators. Any further discrepancy in the number of ND solutions will not make the comparison between two vastly unequal sets fair. The experiment is extended for two more problems: MaF07 and MaF11 in Tables 6.3, and 6.4, respectively.

Table 6.3. MaF07 problem set with $M = 3$. The experiment involves 50 trials, and each algorithm has set with $T_S = 20,000$. Some quality indicators are presented. MOEA/D algorithm is not able to generate 100 solutions; hence, we do not compute and compare the quality indicators with the multi-stage approach.

Algorithm	***N***	***HV*** ↑	***MIP-DoM*** ↓	***k-neighbors distance***		***SP*** ↓	***UD*** ↑	ξ ↓
				mean ↑	***std dev*** ↓			
NSGA-III	100	0.2503	*1.4809*	0.0254	0.0179	0.0279	0.6400	0.9236
MuSt-NSGA-III	100	**0.2551**	***1.4508***	**0.0498**	**0.0133**	**0.0190**	**0.8781**	**0.6991**
MOEA/D	85	-	-	-	-	-	-	-
MuSt-MOEA/D	100	0.2205	1.1091	0.0374	0.0148	0.0229	0.7366	0.7080
C-TAEA	100	0.2603	1.3762	0.0272	0.0224	0.0335	0.4616	1.1244
MuSt-C-TAEA	100	**0.2702**	**1.3036**	**0.0467**	**0.0143**	**0.0211**	**0.8916**	**0.7503**
CLIA	100	*0.2076*	*1.5099*	*0.0314*	0.0221	0.0341	0.4909	0.9842
MuSt-CLIA	100	***0.2218***	***1.3957***	***0.0441***	**0.0110**	**0.0164**	**0.8425**	**0.7239**
CLIA	100	*0.2066*	1.6067	0.0310	0.0225	0.0347	0.5065	0.9870
MuSt-CC	100	***0.2162***	**1.2410**	**0.0436**	**0.0110**	**0.0163**	**0.8264**	**0.7390**

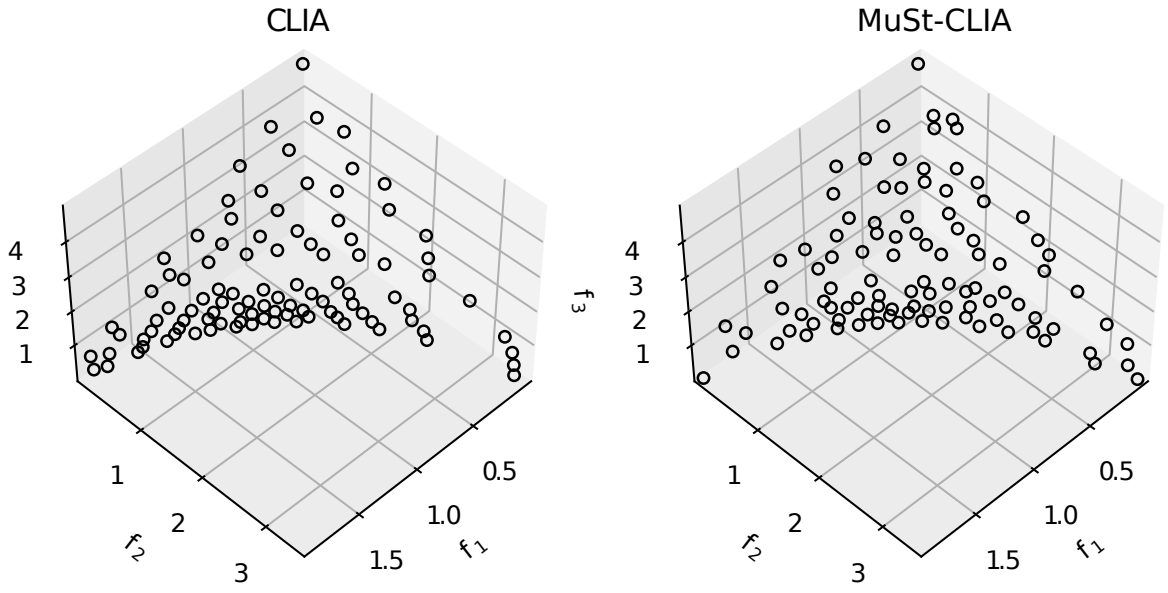

Figure 6.10. CLIA and MuSt-CLIA ND solutions from the median HV run of 50 runs for MaF11.

It is important to note that MOEA/D could only find 85 ND solutions with

100 RVs on the entire unit simplex for MaF07 (presented in Table 6.3) and MaF11 problems. The final population of MOEA/D does not have any more ND points to increase the number of ND points to the desired number (100, in our case). Thus, we do not compute the performance indicators to compare with its multi-stage version.

Analyzing the results from MaF01 problem, presented in Table 6.2, a similar pattern observed in Table 6.1 also appears here. There is a significant statistical difference favoring the multi-stage approach when the uniformity indicators are analyzed. For MuSt-CLIA and MuSt-CC, regarding the HV indicator, there is no statistical difference between the baseline algorithm and the multi-stage approach.

Table 6.4. MaF11 problem set with $M = 3$. The experiment involves 50 trials, and each algorithm has set with $T_S = 20,000$. Some quality indicators are presented. MOEA/D algorithm is not able to generate 100 solutions; hence, we do not compute and compare the quality indicators with the multi-stage approach.

Algorithm	***N***	***HV*** ↑	***MIP-DoM*** ↓	***k-neighbors distance***		***SP*** ↓	***UD*** ↑	ξ ↓
				mean ↑	***std dev*** ↓			
NSGA-III	100	*0.7273*	*1.5877*	*0.0855*	0.0198	0.0282	0.9030	0.2285
MuSt-NSGA-III	100	***0.7371***	***1.5733***	***0.0868***	**0.0146**	**0.0204**	**0.9868**	**0.2098**
MOEA/D	85	-	-	-	-	-	-	-
MuSt-MOEA/D	100	0.3593	0.7533	0.0535	0.0103	0.0162	0.6854	0.5377
C-TAEA	100	0.5782	***1.3290***	0.0608	0.0297	0.0446	0.7901	0.6010
MuSt-C-TAEA	100	**0.7500**	*1.3526*	**0.0675**	**0.0174**	**0.0264**	**0.9696**	**0.4102**
CLIA	100	0.7389	***1.5326***	0.0727	0.0264	0.0408	*0.9729*	0.4795
MuSt-CLIA	100	**0.7406**	*1.5397*	**0.0737**	**0.0195**	**0.0303**	***0.9803***	**0.3794**
CLIA	100	0.7398	***1.5356***	*0.0723*	0.0258	0.0399	*0.9819*	0.4827
MuSt-CC	100	**0.7421**	*1.5423*	***0.0743***	**0.0188**	**0.0293**	***0.9893***	**0.3746**

Figure 6.9 shows the ND points from C-TAEA and MuSt-C-TAEA. A better distribution of ND points can be observed with the multi-stage procedure.

In Table 6.4, the results on MaF11 problem are presented. In this case, there is a statistical difference in CLIA comparisons related to HV. However, considering the k-neighbor distance mean, the pair-wise values are very close, and there is no statistical difference. In Figure 6.10, it is possible to observe the CLIA and MuSt-CLIA solution sets.

In the Appendix A, more results on other problems, such as, ZDT3, C2-DTLZ2, MW14, some more MaF problems, and DAS-CMOP problems, are provided.

6.4.4 MuSt-NSGA-III Applied to Many-objective Problems

Next, to analyze the effect of the number of objective functions on the proposed approach, MaF07 and C2-DTLZ2 are considered. For both problems, the number of objective functions is set to $M = 3$, 5, 8, and 10 using $N = 100$, 200, 400, and 600, and $T_S = 20,000$, $40,000$, $80,000$, and $120,000$, respectively. Only NSGA-III and MuSt-NSGA-III are used in this study. Table 6.5 shows the average of performance indicators over 50 runs on MaF07 problem.

Table 6.5. Number of objective functions is set to $M = 3$, 5, 8, and 10, with $N = 100$, 200, 400, and 600, and, $T_S = 20,000$, $40,000$, $80,000$, and $120,000$, respectively, for MaF07 problem. Average of quality indicators for 50 runs are presented.

Algorithm	***M***	***N***	***HV*** ↑	***MIP-DoM*** ↓	***k-neighbors distance***		***SP*** ↓	***UD*** ↑	ξ ↓
					mean ↑	***std dev*** ↓			
NSGA-III	3	100	0.2503	*1.4809*	0.0254	0.0179	0.0279	0.6400	0.9236
MuSt-NSGA-III		100	**0.2551**	***1.4508***	**0.0498**	**0.0133**	**0.0190**	**0.8781**	**0.6991**
NSGA-III	5	200	0.2284	*3.3726*	0.0970	0.0606	0.0760	0.6224	0.7188
MuSt-NSGA-III		200	**0.2300**	***3.0803***	**0.1323**	**0.0392**	**0.0585**	**0.9212**	**0.4425**
NSGA-III	8	400	***0.0528***	***5.4310***	0.1489	0.0700	0.1234	0.6698	0.6098
MuSt-NSGA-III		400	*0.0526*	*5.5817*	**0.1731**	**0.0518**	**0.0864**	**0.8119**	**0.4571**
NSGA-III	10	600	***0.0323***	***5.5446***	0.1982	0.0662	0.1382	0.6721	0.5454
MuSt-NSGA-III		600	*0.0319*	*5.6760*	**0.2167**	**0.0359**	**0.0672**	**0.9500**	**0.3827**

Considering HV and MIP-DoM, the results of MuSt-NSGA-III for $M = 8$ and 10 are not statistical different from NSGA-III. All other results bring evidences favoring the multi-stage procedure. In Figure 6.11, a comparison of ND solutions for eight-objective MaF07 problem from NSGA-III and MuSt-NSGA-III are shown on parallel coordinate plots (PCPs). Visually, it is possible to observe a better regularity in the position of the lines for MuSt-NSGA-III, particularly for objective functions f_3, f_7, and f_8.

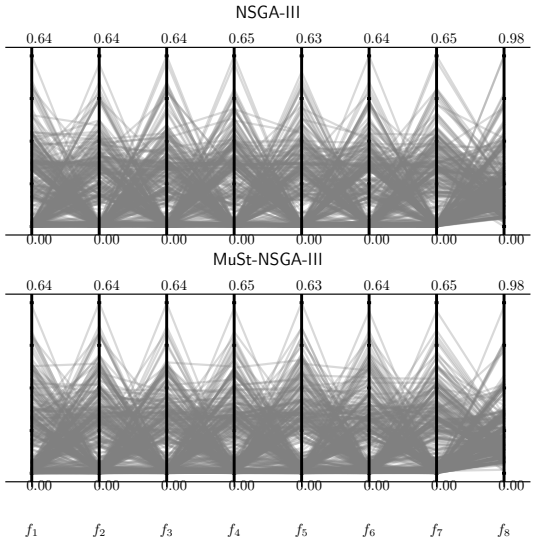

Figure 6.11. Parallel coordinate plots for NSGA-III and MuSt-NSGA-III for the MaF07 problem with $M = 8$. 50 runs are made and the presented results come from the median HV run.

Table 6.6 shows the results for C2-DTLZ2. There is no statistical difference for HV, MIP-DoM, and k-neighbors distance mean indicators, as the number of objectives increase. All other indicators show a more uniform distribution for MuSt-NSGA-III, as observed for the MaF07 problem.

Table 6.6. Number of objectives is set to $M = 3$, 5, 8, and 10, $N = 100$, 200, 400, and 600, and, $T_S = 20,000$, $40,000$, $80,000$, and $120,000$, respectively, for C2-DTLZ2 problem. Average of quality indicators for 50 runs are presented.

Algorithm	***M***	***N***	***HV*** ↑	***MIP-DoM*** ↓	***k-neighbors distance***		***SP*** ↓	***UD*** ↑	ξ ↓
					mean ↑	***std dev*** ↓			
NSGA-III	*3*	100	0.7379	0.9331	0.0573	0.0365	0.0540	0.6965	0.4780
MuSt-NSGA-III		100	**0.7476**	**0.9221**	**0.0710**	**0.0140**	**0.0199**	**1.0000**	**0.3506**
NSGA-III	*5*	200	*0.9603*	*1.0005*	0.1647	0.0459	0.0812	*0.7818*	0.4874
MuSt-NSGA-III		200	***0.9607***	***1.0004***	**0.1695**	**0.0268**	**0.0340**	***1.0000***	**0.4197**
NSGA-III	*8*	400	***0.9812***	***1.0007***	*0.2036*	0.0732	0.1199	0.7741	0.5038
MuSt-NSGA-III		400	*0.9811*	*1.0013*	***0.2042***	**0.0464**	**0.0563**	**0.9969**	**0.4798**
NSGA-III	*10*	600	***0.9612***	***1.0002***	*0.2255*	0.0765	0.1247	0.7921	0.4951
MuSt-NSGA-III		600	***0.9612***	*1.0007*	***0.2314***	**0.0698**	**0.0815**	**0.0849**	**0.4437**

6.5 Parametric Studies

Algorithm 3 shows that four input are needed to execute the MuSt-EMaO framework. Of them, the EMaO algorithm, desired number of ND solutions, and the overall number of SEs are supplied by the user and are not algorithm parameters which can be set in any way by the developer. The above section has shown how the proposed MuSt-EMaO framework can be integrated with a number of EMaO algorithms to improve their performance in terms of finding a well-distributed set of a pre-specified number of desired ND point reliably. The use of three stages adaptively determines if the first stage will continue until all pre-specified number of solution evaluations are completed, or if the algorithm has to move to Stage 2 to ensure if any active RVs is left out, or if the algorithm needs to produce denser points to come up the desired number of ND points. The $\boldsymbol{P}^{\boldsymbol{seed}}$ is the initial supplied population, which is again an optional entity, supplied by the user. If the user does not have any specific initial seed population to supply, the algorithm can create a random population to start Stage 1. The proposed multi-stage procedure requires only one parameter: γ, which determines the proportion of total SEs to be dedicated to Stage 3, while the remaining SEs are equally distributed between Stages 1 and 2. Therefore, the number of desired ND points (N) is truly not a parameter for the algorithm but is the expected number that the user needs to provide. In this section, first, we show that the MuSt-NSGA-III works well with N values other than 100.

6.5.1 Effect of Number of Desired Solutions

The above experiments have attempted to find $N = 100$ ND solutions. Next, we investigate the effect of number of desired solutions (N) on the performance of our proposed multi-stage procedure. Table 6.7 presents the results on the MaF07 problem for three different N values: 50, 100 and 200. We use a linearly proportionate SEs: 10,000, 20,000 and 40,000 for the three scenarios, respectively. We have also set $\gamma = 0.5$ in this study. It is clear from the table that MuSt-NSGA-III is able to produce a more uniformly distributed set of efficient solutions compared to NSGA-III for the same number of respective SEs over 50 runs. In most cases, MuSt-NSGA-III is statistically superior to NSGA-III.

Next, we consider the crashworthiness problem and show the results for $N =$ 50, 100 and 200 as well in Table 6.8. For this problem, MuSt-NSGA-III performs statistically superior to NSGA-III over 50 independent runs, clearly indicating the advantage of the multi-stage procedure.

Table 6.7. Results using MaF07. Number of solutions are set to $N = 50$, 100, 200 and $M = 3$, using $T_S = 10,000$, 20,000, and 40,000, respectively. Average of quality indicators for 50 runs are presented.

Algorithm	*N*	*HV* ↑	*MIP-DoM* ↓	*k-neighbors distance*		*SP* ↓	*UD* ↑	ξ ↓
				mean ↑	*std dev* ↓			
NSGA-III	*50*	*0.2416*	1.1875	0.0446	0.0367	0.0557	0.6489	0.7807
MuSt-NSGA-III		***0.2421***	**1.0841**	**0.0692**	**0.0229**	**0.0337**	**0.7988**	**0.5837**
NSGA-III	*100*	0.2503	*1.4809*	0.0254	0.0179	0.0279	0.6400	0.9236
MuSt-NSGA-III		**0.2551**	***1.4508***	**0.0498**	**0.0133**	**0.0190**	**0.8781**	**0.6991**
NSGA-III	*200*	0.2546	*1.5469*	0.0170	0.0118	0.0183	0.5922	1.1038
MuSt-NSGA-III		**0.2565**	***1.4665***	**0.0335**	**0.0078**	**0.0113**	**0.8990**	**0.8840**

Table 6.8. Results using Crashworthiness. Number of solutions are set to $N = 50$, 100, 200 and $M = 3$, using $T_S = 10,000$, 20,000, and 40,000, respectively. Average of quality indicators for 50 runs are presented.

Algorithm	*N*	*HV* ↑	*MIP-DoM* ↓	*k-neighbors distance*		*SP* ↓	*UD* ↑	ξ ↓
				mean ↑	*std dev* ↓			
NSGA-III	50	0.1091	1.4982	0.0275	0.0246	0.0388	0.5625	0.9016
MuSt-NSGA-III		**0.1132**	**1.3401**	**0.0592**	**0.0187**	**0.0283**	**0.7846**	**0.5409**
NSGA-III	100	0.1101	1.4844	0.0195	0.0166	0.0260	0.5440	0.9333
MuSt-NSGA-III		**0.1139**	**1.0682**	**0.0403**	**0.0112**	**0.0169**	**0.7991**	**0.6205**
NSGA-III	200	0.1135	1.4164	0.0138	0.0110	0.0172	0.5268	1.0229
MuSt-NSGA-III		**0.1162**	**0.9642**	**0.0281**	**0.0074**	**0.0108**	**0.8226**	**0.7266**

6.5.2 Effect of the Solution Evaluation Budget on Each Stage

Next, we perform a parametric study with the γ parameter, that decides the individual SEs for each stage. We assume that $T_1 = T_2$, providing two equal chances for the EMaO algorithm to reliably find a set of ND solutions by first focusing on the entire efficient set in Stage 1 and then focusing on the IRVs in Stage 2, that could not be covered in Stage 2. Using T_S as the total number of solution evaluations, we define $T_3 = \gamma T_S$ and $T_1 = T_2 = \frac{1-\gamma}{2} T_S$. Based on this setting, three different configuration tests are designed with $\gamma = 1/3$, $1/2$, and $2/3$. Therefore, for example, considering 10,000 total solution evaluations, a configuration set with $\gamma = 1/2$ will set $T_1 = 2,500$, $T_2 = 2,500$, and $T_3 = 5,000$. Note that Stage 3 involves two iterations.

Our goal is to determine the statistically best setting considering all quality in-

dicators and all problems used in this study. However, the usual multiple pairwise comparisons, such as the ones done in the Wilcoxon test, suffers from the multiplicative effect to control the family-wise error rate [García et al., 2010]. We adopt two non-parametric statistical tests that are able to make multiple comparisons. The first test, the Friedman test, ranks the algorithm's performance for each problem and computes the average performance among problems, and uses these rankings to test the null hypothesis: the equivalence of median metric value for different γ values versus the alternative hypothesis that two or more medians are different. The second test, the Quade test [García et al., 2010], considers the difference among the problems. It calculates the range of the problems as the maximum differences between the samples and ranks the k problem ranges. Then, these ranks are assigned to the problems and represent a weight to the ranks obtained by the Friedman method. Finally, it uses the same null and alternative hypotheses as in the Friedman test.

Table 6.9 shows the Friedman test results using the HV values in the top part of the table. The mean HV value and the ranking of each γ is shown for each problem in brackets. The mean HV-based ranking is tabulated in the last row. While $\gamma = 1/2$ performs the best, the other two values are also statistically equivalent. Similar tables are also produced for all other indicators. The complete Friedman table for each quality indicator is shown in the Appendix A. Here, we show only the final ranking for the other indicators.

The numbers indicate that it is not possible to reject the null hypothesis, concluding that there is an equivalence among all three γ values for the hypervolume metric. We conduct the same test using the Quade ranking test, and even considering the different problem difficulties, it is again impossible to reject the null hypothesis. Therefore, the experiments present no statistical difference related to this quality indicator. The same phenomenon happens to k-neighbors distance mean, UD, and Evenness, in which it is not possible to observe statistical differences among different γ values.

Contrarily, k-neighbors distance std-dev and spacing have produced different results. Results indicate statistically significant differences among γ values, preferring $\gamma = 1/2$ and $2/3$.

The main drawback of the Friedman and Quade tests is that they can only detect significant differences over multiple comparisons. A general recommendation approach is: if the null hypothesis is rejected, we should proceed with a post-hoc procedure to characterize these differences [Carrasco et al., 2020]. Then, applying a post-hoc test, we can obtain a p-value that determines the degree of rejection of each pairwise hypothesis. Therefore, we conduct a post-hoc analysis using the adjusted p-values calculated by Shaffer's procedure [Derrac et al., 2011].

Table 6.9. Friedman ranks for HV mean for all config sets using $\gamma = 1/3$, $1/2$, and $2/3$. For k-neighbors distance distance mean, k-neighbors distance std dev, SP, UD, and ξ quality indicators, only the final ranking on all config tests are shown for brevity.

Problem sets	**Config sets using γ**		
	1/3	1/2	2/3
	HV Friedman test rank		
ZDT3	0.4817 (1)	0.4816 (2)	0.4815 (3)
MaF01	0.7275 (2)	0.7278 (1)	0.7272 (3)
MaF07	0.6813 (3)	0.6819 (2)	0.6823 (1)
MaF10	0.6792 (1)	0.6330 (2)	0.5543 (3)
MaF11	0.7386 (3)	0.7422 (1)	0.7392 (2)
MaF12	0.7700 (3)	0.7709 (1)	0.7701 (2)
Crashworthiness	0.7447 (3)	0.7460 (1)	0.7458 (2)
Carside impact	0.6823 (2)	0.6823 (3)	0.6826 (1)
DAS-CMOP7	0.7808 (1)	0.7794 (2)	0.7755 (3)
DAS-CMOP8	0.7884 (1)	0.7880 (2)	0.7845 (3)
DAS-CMOP9	0.5108 (1)	0.5015 (2)	0.4944 (3)
Ranking	*1.90*	***1.72***	*2.36*
k-neighbors distance mean Friedman test			
Ranking	*2.18*	***1.81***	*2.00*
k-neighbors distance std dev Friedman test			
Ranking	2.72	1.72	**1.54**
Spacing Friedman test			
Ranking	2.90	**1.54**	**1.54**
UD Friedman test			
Ranking	*2.27*	***1.86***	***1.86***
ξ Friedman test			
Ranking	*2.45*	***1.64***	*1.90*

Table 6.10 makes a pair-wise comparison of three γ configurations for k-neighbors distance standard deviation and spacing performance indicators on all problems. Considering a significance level of 0.05, it can be concluded that there is always a difference in performance with $\gamma = 1/3$ compared to $1/2$ or $2/3$. However, the comparisons between $\gamma = 1/2$ and $\gamma = 2/3$ do not present a statistical difference for both quality indicators. It leads us to conclude that a budget with a number of solution evaluations greater than $\gamma = 1/2$ performs better. Since the final adjustments with increased number of RVs in Stage 3 determines the quality of the overall approach, the parametric study finds that allocating more SEs at Stage 3 is a better strategy.

Table 6.10. HV Shaffer's adjusted p-values for tests considering multiple comparisons among all methods. The data is sorted by the adjusted p-value.

Hypothesis	Shaffer
k-neighbors distance std dev	
Config test $\gamma = 1/3$ vs Config test $\gamma = 2/3$	0.0426
Config test $\gamma = 1/3$ vs Config test $\gamma = 1/2$	0.0426
Config test $\gamma = 1/2$ vs Config test $\gamma = 2/3$	0.7060
Spacing	
Config test $\gamma = 1/3$ vs Config test $\gamma = 2/3$	0.0126
Config test $\gamma = 1/3$ vs Config test $\gamma = 1/2$	0.0200
Config test $\gamma = 1/2$ vs Config test $\gamma = 2/3$	0.5930

6.6 Final observations

Evolutionary algorithms are stochastic, thereby producing different outcomes in different runs. To make evolutionary multi-objective optimization algorithms reproduce a desired number of non-dominated solutions repeatedly, this chapter has proposed a three-stage reference vector (RV) based framework that takes multiple attempts to focus on various key aspects of multi-objective problem solving. The first stage attempts to find a skeleton of active RVs that contain at least one associated non-dominated (ND) solution. The second stage attempts to focus its search on inactive RVs to ensure no true associated ND solutions are left out. Having identified all active RVs, the third stage focuses on searching the desired number of ND points in two iterations by serially increasing the number of RVs. While each of the three stages has its individual role, if an earlier stage is able to find the desired number of ND points, the algorithm consumes the remaining budget of solution evaluations to make the convergence and distribution better without truly moving to the latter stages, thereby degenerating to a single-stage original algorithm, if adequate. A later stage starts from the final populations of the previous stages to make the overall algorithm more efficient.

The working principle of the multi-stage EMaO has been demonstrated on two-objective and three-objective problems. After that, results on 12 multi-objective test problems have shown the superiority of the multi-stage NSGA-III procedure in terms of achieving a better distribution of points measured through seven performance indicators. The multi-stage idea has been integrated with MOEA/D, C-TAEA, and CLIA algorithms, with an improved performance in each case. Thereafter, the multi-stage

NSGA-III was applied to many-objective problems having 5-10 objective problems. Better performance has been reported compared to the original NSGA-III for the same number of solution evaluations.

Parametric studies were then performed with a single parameter defining the proportion of SEs dedicated to each stage. With statistical analysis, it has been shown that the multi-stage concept works better with 50% or more total SEs assigned to Stage 3, considering the problem sets tested. Furthermore, the multi-stage EMaO has also worked well with different desired ND solutions, thereby making the overall procedure generically applicable.

The multi-stage philosophy proposed here makes the approach practical, as users are expected to execute the optimization algorithm for a known number of SEs allocated to solve a problem with the desired number of non-dominated solutions as a target. Multiple attempts to ensure that no critical non-dominated solutions are left out, and a search for a uniformly dense set of solutions on the entire efficient frontier to produce the requisite number of desired solutions stay as the hallmark of this proposal.

Part IV

Concluding remarks

Chapter 7

Conclusion

7.1 Introduction

In this chapter, we present a global view of the contributions of this thesis. We have summarised the research carried out within each chapter in Section7.2. Finally, we have explained how we deal with the challenges of multi- and many-objective optimization stated at the beginning of this work. Then, we outlined some directions for future works enabling some research ideas in Section 7.3.

7.2 Summary of Results

In Chapter 3 we have introduced DoM indicator. The first mixed-integer programming model to approach the DoM calculation has been discussed. Experiments were also presented with comparisons using MIP-DoM , ϵ-indicator, $IGD+$, and Hypervolume indicators. Some many-objective experiments have shown that our novel formulation could deal with problem sets with 5, 10, and 15 objectives with till 240 members in the solution sets. Some further extensions have been presented, showing the quality indicator and its use in other situations.

Based on this extensive study, we have observed that MIP-DoM brings the following advantages as a binary quality indicator:

- It did not require any pre-defined set of points, such as, a reference point or a reference set;
- It was not affected by dominance resistant solutions, unlike that in HV (it can be affected by the reference point definition);

- It considers all the solutions of both sets being used. In its calculation, there is a try to use all the information available.
- DoM was sensitive in capturing the four performance facets between two sets: convergence, spread, uniformity, and cardinality;
- It appeared to have a highly Pearson correlation with HV;
- It presented a monotonic decrease in its value when the first set ($\boldsymbol{Q}$) is fixed and the second set ($\boldsymbol{P}$) approaches towards the Pareto front;

However, the MIP-DoM quality indicator has still presented some drawbacks:

- MIP-DoM time calculation was polynomial to number of solutions;
- Although it could be used as a running quality indicator and presents a correct ranking among curves, in some cases, the difference in MIP-DoM values between two consecutive generations is in decimal places. Since MIP-DoM indicates the minimum Manhattan distance move to make one set dominated by other, for two close sets, the MIP-DoM can be very small, when compared to the change in hypervolume or IGD+ performance indicators;
- It demands an efficient MIP solver in its calculation.

In Chapter 4 a MIP-DoM compact formulation was presented. It was a straightforward and fast formulation compared to the former one. Moreover, the compact model brought a reduced number of binary and continuous variables, as well as a number of constraints. Experiments have assessed the compact formulation behavior with a time spent comparison. There was a significant improvement in the time spent by the model. However, even with the compact formulation improvement, we have observed a polynomial behavior regarding the number of objectives (hours to be calculated) which motivated us to investigate some situations with some practical limits that could be imposed on the compact model. There were still some cases that took hours to be computed.

Chapter 5 has finished our first part. It has analyzed an approximate MIP-DoM calculation using a clustering algorithm. The results have shown that the proposed approximation is computationally faster than the MIP-DoM compact model. Furthermore, the approximate MIP-DoM provided accurate estimates compared to the exact MIP-DoM with a tiny average approximation error. We have also explored some limits of the compact formulation, and we have observed that the approximate method could

alleviate the hard instances, bringing an interesting trade-off between the error rate and the time spent by the model.

In Part II, in Chapter 6 we have dealt with another challenge in the EMO/EMaO area (as initially proposed by this work): how to balance convergence and diversity while managing some common shortcomings. Our multi-stage proposal has produced the desired number of non-dominated solutions with confidence. Some results have been demonstrated on two and three-objective problems, testing on 12 multi-objective test problems. It has shown the superiority of the multi-stage approach using different base algorithms such as NSGA-III, MOEA/D, C-TAEA, and CLIA algorithms. Furthermore, many-objective problems having 5-10 objective problems have been tested. Compared to the original base algorithms, better performance has been reported for the same number of solution evaluations.

7.3 Future research

Despite this extensive study on analyzing properties of the DoM indicator and our EMO and EMaO multi-stage framework, several future studies are worth pursuing.

Investigate some inner solution sets characteristics in MIP-DoM, for some experiments in our study with MIP-DoM, we observed high variability in the computational time for the same problem size and population size. It was observed that this fact was inherent to the distribution or density and some inner characteristics involving the two solution sets, $\boldsymbol{P}$ and $\boldsymbol{Q}$. This issue deserves to be better investigated to improve a MIP-DoM model, creating, for example, new constraints in the MIP model. Furthermore, in the MIP approach for DoM, other directions also deserve to be investigated: i) How to efficiently calculate DoM using MIP and exploiting some inherent solution set features, such as solution set density; and ii) How to define a priori the minimum number of $\boldsymbol{p}_i \neq \boldsymbol{p}'_i$ and what benefits it can bring to the MIP model, for example.

Using the additional information generated by the MIP-DoM, besides its use as a quality indicator, MIP-DoM generates a new solution set that dominates another set, $\boldsymbol{P'}$. An evolutionary operator can use this information to create better solutions for faster convergence. One way to accelerate the convergence of an EMO or EMaO is by performing solutions improvement in the objective space [Adra et al., 2009] and [Gaspar-Cunha et al., 2004]. In general, there are two main steps in the methodology: 1) For some solutions, it must find a way to improve itself in the Pareto front direction, and 2) This improvement in the objective space must be valid in the de-

cision space as well; therefore, a validation and a possible correction must be done (the process of seeking to map an objective vector onto its corresponding decision vector is done approximating a 'mapping' using Neural Networks techniques, for example). A convergence operator can use $\boldsymbol{P'}$ as improved solutions in the objective space; afterward, we could use an online procedure (no pre-trained model) to estimate the map from the objective onto the decision space, for example.

Some other MIP-DoM ideas could explore using the DoM concept. For example, it could be used in the selection operator and measure each individual's contribution towards the convergence. Furthermore, it could be possible to understand how each individual helps in convergence and diversity using the dominance move concept during each generation. Still, is it possible to extend MIP-DoM and turn it into n-ary quality indicator in the performance quality indicator subject? In the end, the user would like to compare multiple algorithms, not only in the pair-wise method. An n-ary quality indicator would be helpful in many different scenarios.

Extending the muti-stage framework, in Part II Chapter 6, is possible. An extensive study for finding a reliable set of the desired number of ND points could be extended in various ways. First, the multi-stage concept could be extended to a strategy that does not try to use RVs identified as inactive regions in Stage 2; this fact could possibly improve the convergence by adding some SEs to Stage 3. Second, instead of pre-fixing a γ parameter to allocate a fixed number of SEs for each stage, performance indicators for stability in convergence and diversity could be used to terminate each stage properly. Care must be taken to ensure that sufficient SEs are kept for later stages, despite stability conditions not being met for earlier stages. Third, the effect of more iterations in Stage 3 can be tried. Fourth, comparisons of our approach with other shape-invariant EMaO algorithms may be studied as in Falcón-Cardona et al. [2019]; Ishibuchi et al. [2016]. Nevertheless, the multi-stage philosophy proposed has made the approach practical, as users are expected to execute the optimization algorithm for a known number of SEs allocated to solve a problem with the desired number of non-dominated solutions as a target.

These ideas could continuously be evolved by applying such approaches to deal with MOP and MaOP. Additionally, all the codes generated in this thesis are going to be publicly available on my personal GitHub repository, enabling its use as an extension.

Part V

Bibliography

Bibliography

Adra, S. F., Dodd, T. J., Griffin, I. A., and Fleming, P. J. (2009). Convergence acceleration operator for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 13(4):825–847.

Audet, C., Digabel, S. L., Cartier, D., Bigeon, J., and Salomon, L. (2018). Performance indicators in multiobjective optimization.

Bhattacharjee, K. S., Singh, H. K., and Ray, T. (2020). *Many-Objective Optimization with Limited Computing Budget*, pages 17--46. Springer International Publishing, Cham.

Blank, J. and Deb, K. (2020a). Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509.

Blank, J. and Deb, K. (2020b). A running performance metric and termination criterion for evaluating evolutionary multi- and many-objective optimization algorithms. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8.

Blank, J., Deb, K., Dhebar, Y., Bandaru, S., and Seada, H. (2021). Generating well-spaced points on a unit simplex for evolutionary many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 25(1):48–60.

Bozkurt, B., Fowler, J. W., Gel, E. S., Kim, B., Köksalan, M., and Wallenius, J. (2010). Quantitative comparison of approximate solution sets for multicriteria optimization problems with weighted tchebycheff preference function. *Operations Research*, 58(3):650–659.

Bradford, E., Schweidtmann, A., and Lapkin, A. (2018). Efficient multiobjective optimization employing gaussian processes, spectral sampling and a genetic algorithm. *Journal of Global Optimization*, 71.

Branke, J., Deb, K., Miettinen, K., and Słowiński, R., editors (2008). *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Springer-Verlag, Berlin, Heidelberg. ISBN 9783540889076.

Brockhoff, D. and Tušar, T. (2019). Benchmarking algorithms from the platypus framework on the biobjective bbob-biobj testbed. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, pages 1905--1911, New York, NY, USA. ACM.

Burkard, R., Dell'Amico, M., and Martello, S. (2012). *Assignment Problems*. Society for Industrial and Applied Mathematics, USA. ISBN 0898716632.

Carrasco, J., García, S., Rueda, M., Das, S., and Herrera, F. (2020). Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. *Swarm and Evolutionary Computation*, 54:100665. ISSN 2210-6502.

Chand, S. and Wagner, M. (2015). Evolutionary many-objective optimization: A quick-start guide. *Surveys in Operations Research and Management Science*, 20:35–42.

Cheng, R., Li, M., Tian, Y., Xiang, X., Zhang, X., Yang, S., Jin, Y., and Yao, X. (2018). Benchmark functions for the CEC'2018 competition on many-objective optimization. Technical report.

Cheng, R., Rodemann, T., Fischer, M., Olhofer, M., and Jin, Y. (2017). Evolutionary many-objective optimization of hybrid electric vehicle control: From general optimization to preference articulation. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 1(2):97–111.

Chugh, T., Sindhya, K., Hakanen, J., and Miettinen, K. (2019). A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Comput.*, 23(9):3137--3166. ISSN 1432-7643.

Coello, C. A. C., VanVeldhuizen, D. A., and Lamont, G. (2002). *Evol. Algorithms for Solving Multi-Objective Problems*. Boston: Kluwer.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619. ISSN 0162-8828.

d. V. Lopes, C. L., Martins, F. V. C., and Wanner, E. F. (2020). An assignment problem formulation for dominance move indicator. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley Sons, Inc., USA. ISBN 047187339X.

Deb, K. (2008). Scope of stationary multi-objective evolutionary optimization: A case study on a hydro-thermal power dispatch problem. *Journal of Global Optimization*, 41:479–515.

Deb, K. and Abouhawwash, M. (2016). An optimality theory-based proximity measure for set-based multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 20(4):515–528. ISSN 1941-0026.

Deb, K. and Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, Part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577--601.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: [nsga-ii]. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Deb, K. and Saxena, D. K. (2005). On finding pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. Technical report.

Deb, K. and Sundar, J. (2006). Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, page 635–642, New York, NY, USA. Association for Computing Machinery.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2002). Scalable multi-objective optimization test problems. In *Proceedings of the Congress on Evolutionary Computation (CEC-2002)*, pages 825--830.

Deisenroth, M. P., Faisal, A. A., and Ong, C. S. (2020). *Mathematics for Machine Learning*. Cambridge University Press.

Deng, J. and Zhang, Q. (2019). Approximating hypervolume and hypervolume contributions using polar coordinate. *IEEE Transactions on Evolutionary Computation*, 23(5):913–918. ISSN .

Derrac, J., García, S., Molina, D., and Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18. ISSN 2210-6502.

Faina, L. (2020). A survey on the cutting and packing problems. *Bollettino dell'Unione Matematica Italiana*, 13(4):567–572. ISSN 2198-2759.

Falcón-Cardona, J. G., Coello Coello, C. A., and Emmerich, M. (2019). Cri-emoa: A Pareto-front shape invariant evolutionary multi-objective algorithm. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 307--318. Springer.

Falcón-Cardona, J. G., Ishibuchi, H., and Coello, C. A. C. (2020). Riesz s-energy-based reference sets for multi-objective optimization. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8.

Fan, Z., Li, W., Cai, X., Li, H., Wei, C., Zhang, Q., Deb, K., and Goodman, E. (2020). Difficulty Adjustable and Scalable Constrained Multiobjective Test Problem Toolkit. *Evolutionary Computation*, 28(3):339–378. ISSN 1063-6560.

Fonseca, C. M., da Fonseca, V. G., and Paquete, L. (2005). Exploring the performance of stochastic multiobjective optimisers with the second-order attainment function. In *Third Int. Conf. on Evol. Multi-Criterion Optimization, EMO-2005*, pages 250--264. Berlin: Springer.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171--2175.

Fritsche, G. and Pozo, A. (2020). The analysis of a cooperative hyper-heuristic on a constrained real-world many-objective continuous problem. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8.

Gamrath, G., Anderson, D., Bestuzheva, K., Chen, W.-K., Eifler, L., Gasse, M., Gemander, P., Gleixner, A., Gottwald, L., Halbig, K., Hendel, G., Hojny, C., Koch, T., Bodic, P. L., Maher, S. J., Matter, F., Miltenberger, M., Mühmer, E., Müller, B., Pfetsch, M., Schlösser, F., Serrano, F., Shinano, Y., Tawfik, C., Vigerske, S.,

Wegscheider, F., Weninger, D., and Witzig, J. (2020). The SCIP Optimization Suite 7.0. Technical report, Optimization Online.

García, S., Fernández, A., Luengo, J., and Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064. ISSN 0020-0255. Special Issue on Intelligent Distributed Information Systems.

Gaspar-Cunha, A., Vieira, A., and Fonseca, C. M. (2004). Multi-objective optimization : Hybridization of an evolutionary algorithm with artificial neural networks for fast convergence.

Ge, H., Zhao, M., Sun, L., Wang, Z., Tan, G., Zhang, Q., and Chen, C. L. P. (2019). A many-objective evolutionary algorithm with two interacting processes: Cascade clustering and reference point incremental learning. *IEEE Transactions on Evolutionary Computation*, 23(4):572–586.

Giagkiozis, I., Purshouse, R. C., and Fleming, P. J. (2013). Generalized decomposition. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 428--442. Springer.

González-Álvarez, D. L. and Vega-Rodríguez, M. A. (2013). Analysing the scalability of multiobjective evolutionary algorithms when solving the motif discovery problem. *Journal of Global Optimization*, 57:467–497.

Guerreiro, A. P., Fonseca, C. M., and Paquete, L. (2021). The hypervolume indicator: Computational problems and algorithms. *ACM Comput. Surv.*, 54(6). ISSN 0360-0300.

Gurobi Optimization, L. (2019). Gurobi optimizer reference manual.

HAIMES, Y. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(3):296–297. ISSN .

Hua, Y., Liu, Q., Hao, K., and Jin, Y. (2021). A survey of evolutionary algorithms for multi-objective optimization problems with irregular pareto fronts. *IEEE/CAA Journal of Automatica Sinica*, 8(2):303–318.

Huband, S., Hingston, P., and Barone, L. (2011). A Review of Multi-objective Test Problems and a Scalable Test Problem Toolkit A Review of Multiobjective Test

Problems and a Scalable Test Problem Toolkit. *IEEE Transactions on Evolutionary Computation*, 10(2006):477--506.

Ibrahim, A., Rahnamayan, S., Martin, M. V., and Deb, K. (2018). 3d-radvis antenna: Visualization and performance measure for many-objective optimization. *Swarm and Evolutionary Computation*, 39:157--176.

Inselberg, A. and Dimsdale, B. (1990). Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *Proceedings of the First IEEE Conference on Visualization: Visualization '90*, pages 361–378.

Ishibuchi, H., Imada, R., Setoguchi, Y., and Nojima, Y. (2018). How to specify a reference point in hypervolume calculation for fair performance comparison. *Evol. Comput.*, 26(3):411?440. ISSN 1063-6560.

Ishibuchi, H., Masuda, H., and Nojima, Y. (2015a). Comparing solution sets of different size in evolutionary many-objective optimization. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2859–2866.

Ishibuchi, H., Masuda, H., Tanigaki, Y., and Nojima, Y. (2015b). Modified distance calculation in generational distance and inverted generational distance. In Gaspar-Cunha, A., Henggeler Antunes, C., and Coello, C. C., editors, *Evolutionary Multi-Criterion Optimization*, pages 110--125, Cham. Springer International Publishing.

Ishibuchi, H., Setoguchi, Y., Masuda, H., and Nojima, Y. (2016). Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes. *IEEE Transactions on Evolutionary Computation*, 21(2):169--190.

Ishibuchi, H., Tsukamoto, N., and Nojima, Y. (2008). Evolutionary many-objective optimization: A short review. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 2419–2426.

Izzo, D. (2012). Pygmo and pykep: Open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization). pages –.

Jain, H. and Deb, K. (2014). An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, Part II: Handling constraints and extending to an adaptive approach. *IEEE Trans, on Evol. Comput.*, 18(4):602--622.

Jünger, M., Liebling, T. M., Naddef, D., Nemhauser, G. L., Pulleyblank, W. R., Reinelt, G., Rinaldi, G., and Wolsey, L. A., editors (2010). *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*. Springer. ISBN 978-3-540-68274-5.

Kaisa, M. (1999). *Nonlinear Multiobjective Optimization*, volume 12 of *International Series in Operations Research & Management Science*. Kluwer Academic Publishers, Boston, USA.

Klotz, E. and Newman, A. (2013). Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science*, 18:18?32.

Knowles, J. D. and Corne, D. W. (2006). Quantifying the effects of objective space dimension in evolutionary multiobjective optimization. In *EMO*.

Kukkonen, S. and Deb, K. (2006). A fast and effective method for pruning of non-dominated solutions in many-objective problems. In Runarsson, T. P., Beyer, H.-G., Burke, E., Merelo-Guervós, J. J., Whitley, L. D., and Yao, X., editors, *Parallel Problem Solving from Nature - PPSN IX*, pages 553--562, Berlin, Heidelberg. Springer Berlin Heidelberg.

Lepenioti, K., Bousdekis, A., Apostolou, D., and Mentzas, G. (2020). Prescriptive analytics: Literature review and research challenges. *International Journal of Information Management*, 50:57 – 70. ISSN 0268-4012.

Li, B., Li, J., Tang, K., and Yao, X. (2015a). Many-objective evolutionary algorithms: A survey. *ACM Comput. Surv.*, 48(1). ISSN 0360-0300.

Li, H., Deb, K., Zhang, Q., Suganthan, P., and Chen, L. (2019a). Comparison between [moea/d] and [nsga-iii] on a set of novel many and multi-objective benchmark problems with challenging difficulties. *Swarm and Evolutionary Computation*, 46:104 – 117. ISSN 2210-6502.

Li, K., Chen, R., Fu, G., and Yao, X. (2019b). Two-archive evolutionary algorithm for constrained multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 23(2):303–315.

Li, K., Wang, R., Zhang, T., and Ishibuchi, H. (2018). Evolutionary many-objective optimization: A comparative study of the state-of-the-art. *IEEE Access*, 6:26194–26214.

Li, M. (2015). *Evolutionary Many-Objective Optimisation: Pushing the Boundaries.* PhD dissertation, Department of Computer Science, Brunel University London, London, UB8 3PH, U.K.

Li, M., Yang, S., and Liu, X. (2015b). A performance comparison indicator for pareto front approximations in many-objective optimization. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 703--710, New York, NY, USA. ACM.

Li, M. and Yao, X. (2017). Dominance move: A measure of comparing solution sets in multiobjective optimization. *CoRR*, abs/1702.00477.

Li, M. and Yao, X. (2019). Quality evaluation of solution sets in multiobjective optimisation: A survey. *ACM Comput. Surv.*, 52(2):26:1--26:38. ISSN 0360-0300.

Liang, Z., Liang, W., Wang, Z., Ma, X., Liu, L., and Zhu, Z. (2021). Multiobjective evolutionary multitasking with two-stage adaptive knowledge transfer based on population distribution. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–13.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., and Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58. ISSN 2214-7160.

Ma, L., Li, N., Guo, Y., Wang, X., Yang, S., Huang, M., and Zhang, H. (2021). Learning to optimize: Reference vector reinforcement learning adaption to constrained many-objective optimization of industrial copper burdening system. *IEEE Transactions on Cybernetics*, pages 1–14.

Ma, X., Yu, Y., Li, X., Qi, Y., and Zhu, Z. (2020). A survey of weight vector adjustment methods for decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 24(4):634--649.

Meneghini, I. R., Alves, M. A., Gaspar-Cunha, A., and Guimarães, F. G. (2020). Scalable and customizable benchmark problems for many-objective optimization. *Applied Soft Computing*, 90:106139. ISSN 1568-4946.

Messac, A. (2004). Normal constraint method with guarantee of even representation of complete pareto frontier.

Mitchell, T. M. (1997). *Machine Learning.* McGraw-Hill, New York. ISBN 978-0-07-042807-2.

Pellicer, P. V., Escudero, M. I., Alzueta, S. F., and Deb, K. (2020). Gap finding and validation in evolutionary multi- and many-objective optimization. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 578–586, New York, NY, USA. Association for Computing Machinery.

Pentico, D. W. (2007). Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176(2):774–793.

Pham, D. T., Dimov, S. S., and Nguyen, C. D. (2005). Selection of k in k-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1):103–119.

Purshouse, R. C. and Fleming, P. J. (2007). On the evolutionary optimization of many conflicting objectives. *IEEE Transactions on Evolutionary Computation*, 11(6):770–784.

Qi, Y., Ma, X., Liu, F., Jiao, L., Sun, J., and Wu, J. (2014). [moea/d] with adaptive weight adjustment. *Evolutionary Computation*, 22(2):231–264.

Ramshaw, L. and Tarjan, R. E. (2012). On minimum-cost assignments in unbalanced bipartite graphs. *HP Labs, Palo Alto, CA, USA, Tech. Rep. HPL-2012-40R1.*

Schott, J. (1995). Fault tolerant design using single and multicriteria genetic algorithm optimization /.

Seada, H., Abouhawwash, M., and Deb, K. (2019). Multiphase balance of diversity and convergence in multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 23(3):503–513.

Shukla, P. and Deb, K. (2005). Comparing classical generating methods with an evolutionary multi-objective optimization method. In *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization (EMO-2005)*, pages 311--325.

Sierksma, G. and Zwols, Y. (2018). *Linear and Integer Optimization; Theory and Practice*, volume 83.

Steuer, R. E. (1986). *Multiple Criteria Optimization: Theory, Computation and Application.* New York: Wiley.

Talukder, A. K. A. and Deb, K. (2020). PaletteViz: A visualization method for functional understanding of high-dimensional pareto-optimal data-sets to aid multi-criteria decision making. *IEEE Computational Intelligence Magazine*, 15(2):36–48.

Tan, K., Lee, T., and Khor, E. (2001). Evolutionary algorithms for multi-objective optimization: performance assessments and comparisons. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, volume 2, pages 979–986 vol. 2.

Tian, Y., Cheng, R., Zhang, X., Cheng, F., and Jin, Y. (2018). An indicator-based multiobjective evolutionary algorithm with reference point adaptation for better versatility. *IEEE Transactions on Evolutionary Computation*, 22(4):609–622. ISSN 1941-0026.

Tian, Y., Cheng, R., Zhang, X., and Jin, Y. (2017). PlatEMO: A MATLAB platform for evolutionary multi-objective optimization. *IEEE Computational Intelligence Magazine*, 12(4):73--87.

Tian, Y., He, C., Cheng, R., and Zhang, X. (2021a). A multistage evolutionary algorithm for better diversity preservation in multiobjective optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(9):5880–5894.

Tian, Y., Zhang, Y., Su, Y., Zhang, X., Tan, K. C., and Jin, Y. (2021b). Balancing objective optimization and constraint satisfaction in constrained evolutionary multiobjective optimization. *IEEE Transactions on Cybernetics*, pages 1–14.

Vesikar, Y., Deb, K., and Blank, J. (2018). Reference point based [nsga-iii] for preferred solutions. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1587–1594.

Wagner, T., Hochstrate, N., and Naujoks, B. (2006). Pareto-, aggregation-, and indicator-based methods in many-objective optimization. *EMO 2007*, pages 742–756.

Wang, L., Hao, Z., and Sun, W. (2019). A novel self-adaptive affinity propagation clustering algorithm based on density peak theory and weighted similarity. *IEEE Access*, 7:175106–175115.

While, R., Bradstreet, L., and Barone, L. (2012). A fast way of calculating exact hypervolumes. *IEEE Trans. Evolutionary Computation*, 16:86–95.

Yang, K., Emmerich, M., Deutz, A., and Bäck, T. (2019). Efficient computation of expected hypervolume improvement using box decomposition algorithms. *Journal of Global Optimization*, 75(1):3--34. ISSN 1573-2916.

Yizong Cheng (1995). Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799.

Yu, P. (1973). A class of solutions for group decision problems. *Management Science. Ser. B, Application Series*, 19.

Yuan, Y., Ong, Y. S., Gupta, A., and Xu, H. (2018). Objective Reduction in Many-Objective Optimization: Evolutionary Multiobjective Approaches and Comprehensive Analysis. *IEEE Transactions on Evolutionary Computation*, 22(2):189--210. ISSN 1089778X.

Zaki, M. and Meira, W. (2020). *Data Mining and Machine Learning: Fundamental Concepts and Algorithms*. Cambridge University Press. ISBN 9781108473989.

Zhang, Q. and Li, H. (2007). [moea/d]: A multiobjective evolutionary algorithm based on decomposition. *Trans. Evol. Comp*, 11(6):712–731. ISSN 1089-778X.

Zhou, A., Zhang, Q., and Jin, Y. (2009). Approximating the set of Pareto optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm. *IEEE Transacations on Evolutionary Computation*, 13(5):1167--1189.

Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, 8(2):173–195. ISSN 1063-6560.

Zitzler, E., Knowles, J., and Thiele, L. (2008). *Quality Assessment of Pareto Set Approximations*, pages 373--404. Springer Berlin Heidelberg, Berlin, Heidelberg.

Zitzler, E. and Künzli, S. (2004). Indicator-based selection in multiobjective search. In *in Proc. 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII*, pages 832--842. Springer.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In Giannakoglou et al., K. C., editor, *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95--100. Int. Center for Numerical Methods in Engg (CIMNE).

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *Trans. Evol. Comp*, 3(4):257–271. ISSN 1089-778X.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132. ISSN .

Part VI

Appendix

Appendix A

Supplementary results of Multi-stage reference-vector-based framework to identify efficient fronts reliably

A.1 Introduction

This supplementary document provides additional discussions and results in support to the multi-stage evolutionary algorithm framework proposed in Chapter 6. Each section is self-explanatory and presents specific results related to the performance of the proposed procedure.

A.2 Results with Different Total Solution Evaluations

To investigate the performance of MuSt-EMaO on different SEs, we have considered one smaller and one larger T_S than 20,000 SEs used in the main study.

Table A.1 presents the average quality indicator values obtained by NSGA-III and MuSt-NSGA-III over 50 runs of both algorithms on 12 problems. In all cases, N is set to 100. In all tests, MuSt-NSGA-III is executed with $T_S = 10,000$ for all problems, except for DAS problems, for which we use $T_S = 50,000$.

Table A.2 presents the average quality indicator values obtained by NSGA-III and MuSt-NSGA-III over 50 runs of both algorithms on 12 problems. In all cases, N is

Table A.1. Experimental results of baseline NSGA-III with complete ND population ($N = 100$) and MuSt-NSGA-III. Seven performance metrics are used to statistically compare both algorithms over 50 runs. In all tests, MuSt-NSGA-III is executed with $T_S = 10,000$ for all problems, except for DAS problems, for which we use $T_S = 50,000$. Symbols ↑ and ↓ indicate better metric values for large and small values, respectively.

Problem	*Algorithm*	*M*	*N*	*HV*		*MIP-DoM* ↓	*k-neighbors distance*		*SP* ↓	*UD* ↑	ξ ↓	*#RVs* Stage 3, i=2
				mean ↑	*std dev*		*mean* ↑	*std dev* ↓				
ZDT3	NSGA-III	2	100	*0.4790*	***0.0012***	**0.0089**	0.0065	0.0043	0.0062	0.7061	*1.7793*	-
	MuSt-NSGA-III		100	***0.4791***	*0.0024*	0.5016	**0.0071**	**0.0050**	**0.0040**	**0.8348**	***1.7914***	232
DTLZ2	NSGA-III	3	100	***0.7914***	***0.0002***	***0.9677***	**0.1013**	***0.0073***	***0.0104***	*.9901*	***0.1232***	-
	MuSt-NSGA-III		100	*0.7909*	*0.0005*	*0.9939*	0.0952	*0.0104*	*0.0109*	***1.9975***	*0.1847*	100
MaF01	NSGA-III	3	100	0.0148	0.0002	**1.7403**	0.0178	0.0147	0.0228	0.5321	0.7992	-
	MuSt-NSGA-III		100	**0.0163**	**0.0001**	1.7822	**0.0422**	**0.0078**	**0.0112**	**0.9387**	**0.2707**	343
MaF07	NSGA-III	3	100	0.3346	0.0133	**1.3347**	0.0267	0.0194	0.0302	0.6309	0.9466	-
	MuSt-NSGA-III		100	**0.3385**	**0.0117**	1.5228	**0.0501**	**0.0133**	**0.0202**	**0.8889**	**0.7224**	236
MaF10	NSGA-III	3	100	***0.3138***	*0.0210*	**0.4571**	0.0287	0.0144	0.0225	0.6138	0.6022	-
	MuSt-NSGA-III		100	*0.3067*	***0.0161***	0.8077	**0.0320**	**0.0112**	**0.0175**	**0.6907**	**0.4518**	163
MaF11	NSGA-III	3	100	*0.7190*	*0.0637*	*1.5394*	*0.0653*	0.0196	0.0291	0.8304	*0.3860*	-
	MuSt-NSGA-III		100	***0.7262***	***0.0614***	***1.5053***	***0.0666***	**0.0135**	**0.0202**	**0.9707**	***0.3620***	122
MaF12	NSGA-III	3	100	*0.6361*	***0.0052***	2.3250	0.0781	0.0217	0.0317	0.8589	0.2890	-
	MuSt-NSGA-III		100	***0.6366***	*0.0047*	**2.2829**	**0.0803**	**0.0138**	**0.0201**	**0.9923**	**0.2560**	108
Carside impact	NSGA-III	3	100	0.2235	**0.0012**	*2.0859*	0.0483	0.0311	0.0467	0.6927	0.4962	-
	MuSt-NSGA-III		100	**0.2243**	0.0022	***2.0044***	**0.0669**	**0.0130**	**0.0189**	**0.9452**	**0.2647**	146
Crashworthiness	NSGA-III	3	100	0.1177	0.0010	1.4018	0.0201	0.0171	0.0267	0.5426	0.9383	-
	MuSt-NSGA-III		100	**0.1210**	**0.0007**	**1.1126**	**0.0407**	**0.0109**	**0.0166**	**0.8194**	**0.6417**	312
DAS-CMOP7	NSGA-III	3	100	***0.6900***	*0.0402*	***0.9081***	*0.0549*	0.0272	0.0408	0.7254	0.4819	-
	MuSt-NSGA-III		100	*0.6685*	***0.0625***	*0.9476*	***0.0593***	**0.0145**	**0.0225**	**0.9709**	**0.3947**	186
DAS-CMOP8	NSGA-III	3	100	***0.6710***	***0.0502***	***0.9437***	0.0527	0.0284	0.0426	0.7166	0.5536	-
	MuSt-NSGA-III		100	*0.6682*	*0.0566*	*0.9642*	**0.0658**	**0.0201**	**0.0303**	**0.9693**	**0.4073**	209
DAS-CMOP9	NSGA-III	3	100	***0.3095***	*0.0533*	***0.4008***	0.0112	***0.0113***	***0.0175***	*0.4071*	*1.0796*	-
	MuSt-NSGA-III		100	*0.3086*	***0.0447***	*0.4248*	**0.0138**	*0.0121*	*0.0196*	***0.4085***	***1.0543***	523

The Wilcoxon signed-rank test with a significance level of 0.05 was applied to perform a statistical comparison between the approaches. Thus, the data in italics means that it is not possible to detect differences between NSGA-III and MuSt-NSGA-III. Data in bold indicates the best value between the approaches.

set to 100. In all tests, MuSt-NSGA-III is executed with $T_S = 30,000$ for all problems, except for DAS problems, for which we use $T_S = 150,000$.

For both cases, the results of MuSt-NSGA-III is better than the baseline NSGA-III approach.

A.3 MuSt-EMaO Results on Further Test Problems

A.3.1 ZDT3 Problem

The ZDT3 problem Zitzler et al. [2000] has disjointed efficient front. Results are presented in Table A.3 with three base EMO and multi-stage algorithms: C-TAEA, MOEA/D, and NSGA-III. In Figures A.1 to A.3, solution sets for the median HV run are depicted, respectively, for NSGA-III, MOEA/D and C-TAEA. Since MOEA/D could not find $N = 100$ ND points, we do not compare MuSt-MOEA/D with the

Table A.2. Experimental results of baseline NSGA-III with complete ND population ($N = 100$) and MuSt-NSGA-III. Seven performance metrics are used to statistically compare both algorithms over 50 runs. In all tests, MuSt-NSGA-III is executed with $T_S = 30,000$ for all problems, except for DAS problems, for which we use $T_S = 150,000$. Symbols ↑ and ↓ indicate better metric values for large and small values, respectively.

Problem	*Algorithm*	*M*	*N*	*HV*		*MIP-DoM* ↓	*k-neighbors distance*		*SP* ↓	*UD* ↑	ξ ↓	*#RVs* Stage 3, i=2
				mean ↑	*std dev*		*mean* ↑	*std dev* ↓				
ZDT3	NSGA-III	2	100	0.4813	0.0023	0.1262	0.0073	0.0050	0.0071	0.6965	1.6376	-
	MuSt-NSGA-III		100	**0.4822**	**0.0004**	**0.1123**	**0.0086**	**0.0026**	**0.0037**	**0.9897**	**1.5951**	161
DTLZ2	NSGA-III	3	100	***0.7917***	***0.0000***	*0.6131*	*0.1055*	***0.0030***	***0.0048***	***1.0000***	***0.0816***	-
	MuSt-NSGA-III		100	***0.7917***	***0.0000***	**0.4731**	***0.1053***	*0.0031*	*0.0050*	***1.0000***	*0.0832*	100
MaF01	NSGA-III	3	100	0.0146	0.0002	1.7825	0.0119	0.0147	0.0229	0.5425	0.8209	-
	MuSt-NSGA-III		100	**0.0162**	**0.0001**	**1.7229**	**0.0455**	**0.0072**	**0.0099**	**0.9869**	**0.2305**	341
MaF07	NSGA-III	3	100	0.2379	0.0185	*1.5625*	0.0250	0.0172	0.0267	0.6354	0.9222	-
	MuSt-NSGA-III		100	**0.2477**	**0.0066**	***1.3717***	**0.0504**	**0.0133**	**0.0193**	**0.8776**	**0.6922**	224
MaF10	NSGA-III	3	100	**0.5908**	0.0553	**1.2927**	0.0403	0.0201	0.0225	0.6921	0.7454	-
	MuSt-NSGA-III		100	0.6342	**0.0184**	1.4554	**0.0595**	**0.0142**	**0.0217**	**0.9819**	**0.5063**	153
MaF11	NSGA-III	3	100	*0.7276*	*0.0654*	1.6437	*0.0759*	0.0160	0.0228	*0.9046*	*0.2902*	-
	MuSt-NSGA-III		100	***0.7425***	***0.0610***	**1.5259**	***0.0792***	**0.0133**	**0.0186**	***0.9912***	***0.2732***	103
MaF12	NSGA-III	3	100	*0.6511*	*0.0025*	*2.1812*	0.0864	0.0189	0.0267	0.9111	0.2152	-
	MuSt-NSGA-III		100	***0.6517***	***0.0016***	***2.1699***	**0.0876**	**0.0150**	**0.0206**	**0.9687**	**0.2041**	102
Carside impact	NSGA-III	3	100	0.2299	**0004**	*2.0318*	0.0514	0.0349	0.0518	0.6887	0.4875	-
	MuSt-NSGA-III		100	**0.2315**	0.0015	***1.9891***	**0.0745**	**0.0133**	**0.0183**	**0.9403**	**0.2149**	141
Crashworthiness	NSGA-III	3	100	0.1160	**0.0010**	1.4750	0.0192	0.0168	0.0262	0.5464	0.9664	-
	MuSt-NSGA-III		100	**0.1201**	0.0019	**1.0096**	**0.0412**	**0.0116**	**0.0174**	**0.8048**	**0.6368**	295
DAS-CMOP7	NSGA-III	3	100	***0.7358***	*0.0004*	***0.9875***	*0.0813*	0.0224	0.0328	0.8735	0.2885	-
	MuSt-NSGA-III		100	*0.7355*	***0.0005***	*0.9957*	***0.0821***	**0.0172**	**0.0248**	**0.9951**	**0.2735**	104
DAS-CMOP8	NSGA-III	3	100	***0.7179***	***0.0004***	***0.9848***	**0.0877**	0.0176	0.0255	0.9498	0.2604	-
	MuSt-NSGA-III		100	*0.7174*	*0.0004*	*0.99074*	0.0828	**0.0166**	**0.0245**	**0.9877**	**0.2918**	102
DAS-CMOP9	NSGA-III	3	100	***0.4752***	*0.0979*	***0.5567***	0.0175	0.0152	0.0235	0.4933	0.92176	-
	MuSt-NSGA-III		100	*0.4554*	***0.0783***	*0.6120*	**0.0230**	**0.0114**	**0.0173**	**0.5552**	**0.7307**	357

The Wilcoxon signed-rank test with a significance level of 0.05 was applied to perform a statistical comparison between the approaches. Thus, the data in italics means that it is not possible to detect differences between NSGA-III and MuSt-NSGA-III. Data in bold indicates the best value between the approaches.

Table A.3. ZDT3 problem set. The experiment involves 50 trials, and each algorithm has set with $T_S = 20,000$. Some quality indicators are presented.

Algorithm	*N*	*HV* ↑	*MIP-DoM* ↓	*k-neighbors distance*		*SP* ↓	*UD* ↑	ξ ↓
				mean	*std dev*			
NSGA-III	100	*0.4806*	**0.1086**	0.0071	0.0048	0.0068	0.6983	*1.6641*
MuSt-NSGA-III	100	***0.4816***	0.1594	**0.0083**	**0.0028**	**0.0040**	**0.9634**	***1.6300***
MOEA/D	77	-	-	-	-	-	-	-
MuSt-MOEA/D	100	0.4630	0.1150	0.0069	0.0034	0.0049	0.8827	1.8220
CTAEA	100	0.4798	0.1346	0.0069	0.0076	0.0108	0.6721	*1.6821*
MuSt-C-TAEA	100	**0.4805**	**0.1030**	**0.0079**	**0.0029**	**0.0041**	**0.9870**	***1.6385***

We applied the Wilcoxon signed-rank test with a significance level of 0.05. Data in italics means that it is not possible to detect differences between algorithms. Data in bold indicates the best value.

base MOEA/D algorithm. Figures show that the original algorithms find less dense solutions for the left-most two efficient subsets, while the multi-stage version finds a better distribution – a matter which is also clear from the table.

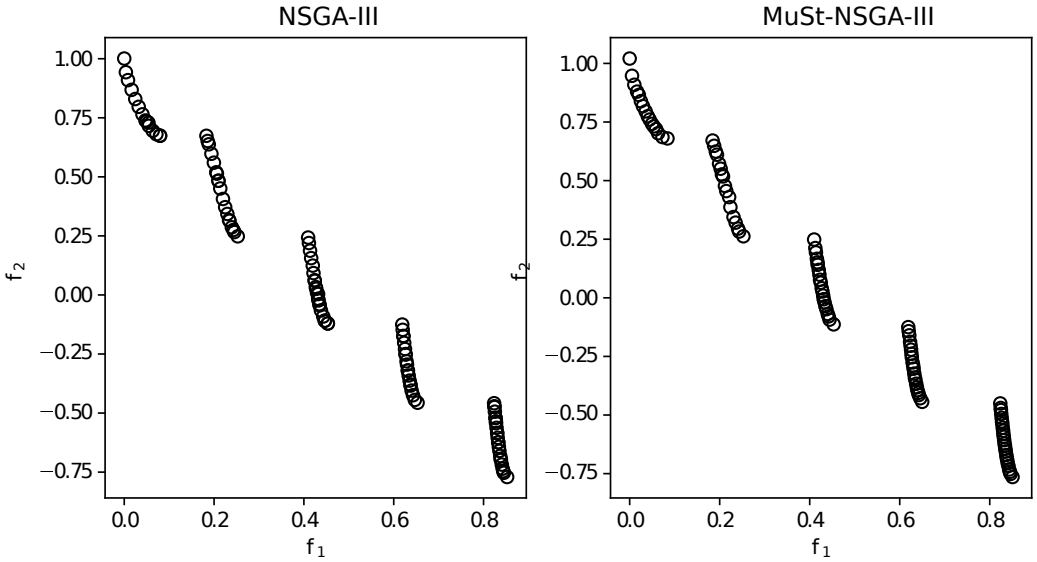

Figure A.1. NSGA-III and MuSt-NSGA-III results for the ZDT3 problem. Results are shown for the median HV run for each case.

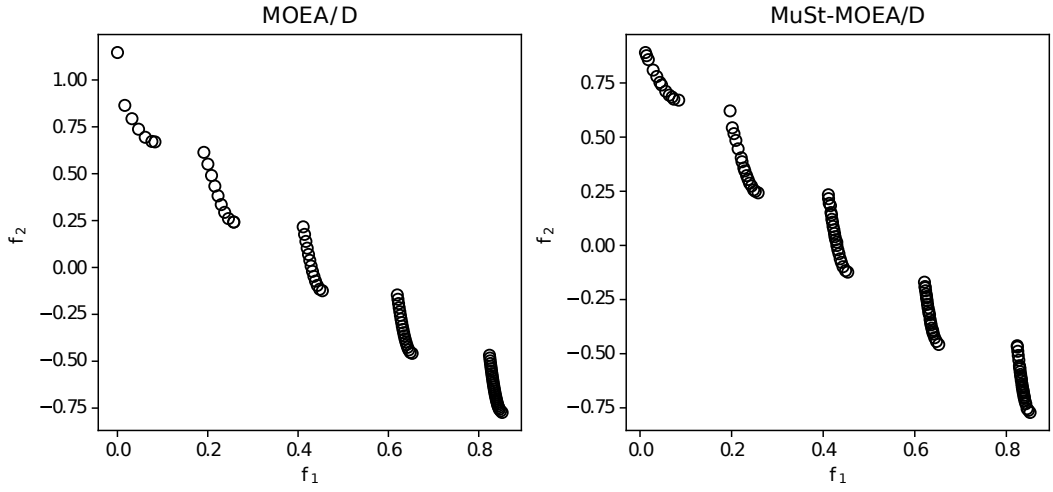

Figure A.2. MOEA/D and MuSt-MOEA/D results for the ZDT3 problem. Results are shown for the median HV run for each case.

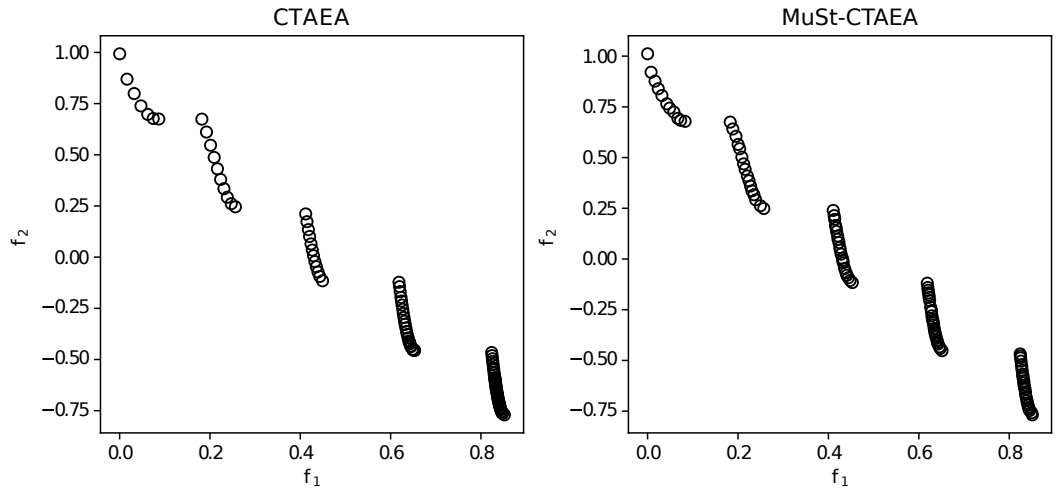

Figure A.3. CTAEA and MuSt-C-TAEA results for the ZDT3 problem. Results are shown for the median HV run for each case.

A.3.2 DTLZ2 Problem

The DTLZ2 problem Deb et al. [2002] has a continuous efficient front – octant of a sphere. Ideally, every RV corresponds to an associated efficient point, hence theoretically there is no need for Stages 2 and 3 for the multi-stage framework. Table A.4 shows that both NSGA-III and MuSt-NSGA-III produce almost an identical statistical performance. The slight difference in performance metrics occur due to the classification operator in Line 4 and an extra non-domination check between $\boldsymbol{S_{1c}^{T1}}$ and $\boldsymbol{S_{1c}^{T23}}$ in Line 8 of the original MuSt-EMaO algorithm.

Table A.4. DTLZ2 problem set with $M = 3$. The experiment involves 50 trials, and each algorithm has set with $T_S = 20,000$. Some quality indicators are presented.

Algorithm	***N***	***HV*** ↑	***MIP-DoM*** ↓	***k-neighbors distance***		***SP*** ↓	***UD*** ↑	ξ ↓
				mean	***std dev***			
NSGA-III	100	***0.7916***	***0.7101***	***0.1052***	***0.0050***	*0.0056*	***1.0000***	***0.0870***
MuSt-NSGA-III	100	***0.7916***	*0.7460*	*0.1035*	*0.0051*	***0.0055***	***1.0000***	*0.0897*
MOEA/D	100	***0.7879***	*0.7799*	*0.1045*	***0.0030***	***0.0042***	*0.4136*	*0.0887*
MuSt-MOEA/D	100	***0.7879***	***0.7668***	***0.1054***	*0.0032*	*0.0048*	***0.4212***	***0.0863***
CTAEA	100	***0.7913***	*0.7012*	*0.1006*	*0.0079*	*0.0106*	*0.4335*	*0.1295*
MuSt-C-TAEA	100	*0.7912*	***0.6958***	***0.1015***	***0.0072***	***0.0102***	***0.4338***	*0.1295*

A.3.3 DTLZ5 Problem

Results on the DTLZ5 problem Deb et al. [2002] are presented next. This problem has a degenerate efficient frontier. Despite three objectives in the problem, the efficient frontier is a single-dimensional curve. Objectives f_1 and f_2 are correlated to each other, making a trade-off between f_1 or f_2 and f_3. Table A.5 shows that MuSt-NSGA-III produces a much denser set of efficient front compared to NSGA-III. MOEA/D and C-TAEA are not able to find close to 100 ND solutions, as most of the RVs do not pass through the efficient curve.

Table A.5. DTLZ5 problem set with $M = 3$. The experiment involves 50 trials, and each algorithm has set with $T_S = 20,000$. Some quality indicators are presented.

Algorithm	***N***	***HV*** ↑	***MIP-DoM*** ↓	***k-neighbors distance***		***SP*** ↓	***UD*** ↑	ξ ↓
				mean	***std dev***			
NSGA-III	100	0.0799	0.3560	0.0051	0.0068	***0.0112***	1.3111	1.2839
MuSt-NSGA-III	100	**0.0814**	**0.2387**	**0.0069**	**0.0060**	*0.0122*	**1.4743**	**0.7454**
MOEA/D	49	-	-	-	-	-	-	-
MS-MOEA	100	0.1251	0.1509	0.0063	0.0066	0.0241	0.4751	0.9384
CTAEA	90	-	-	-	-	-	-	-
MuSt-C-TAEA	100	0.0827	0.4171	0.0062	0.0070	0.0250	0.4238	1.1909

Interestingly, Iteration 2 of Stage 3 in MuSt-NSGA-III uses 1,844 RVs to find 100 ND points for this problem. This is because when only 100 RVs are initialized in Stage 1, very few efficient solutions can be found. Most of the RVs do not end up with any efficient point. When the number of RVs is increased to about 19 times to the number of ND points required, the algorithm is able to find 100 uniformly distributed ND points.

Figures A.4 to A.6 present the median HV solution sets, for NSGA-III, MOEA/D, and C-TAEA, respectively. In all cases, the ND sets from the multi-stage framework are better distributed than their baseline versions.

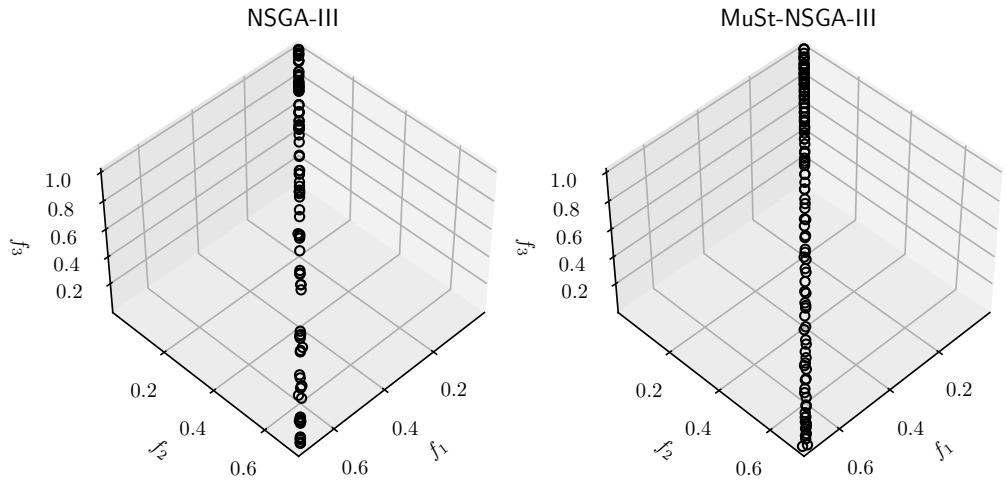

Figure A.4. NSGA-III and MuSt-NSGA-III results for the DTLZ5 problem. MuSt-NSGA-III uses 1,844 RV's on Stage 3, $i = 2$. Results are shown for the median HV run for each case.

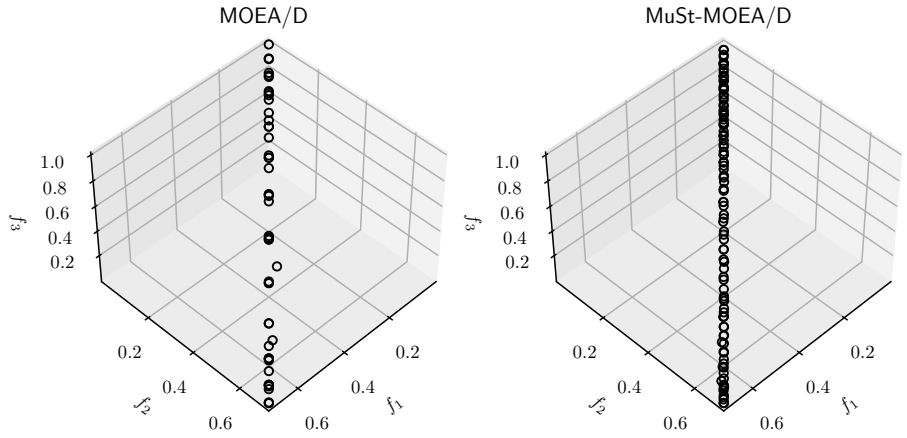

Figure A.5. MOEA/D and MuSt-MOEA/D results for the DTLZ5 problem. MuSt-MOEA/D uses 1,956 RV's on Stage 3, $i = 2$. Results are shown for the median HV run for each case.

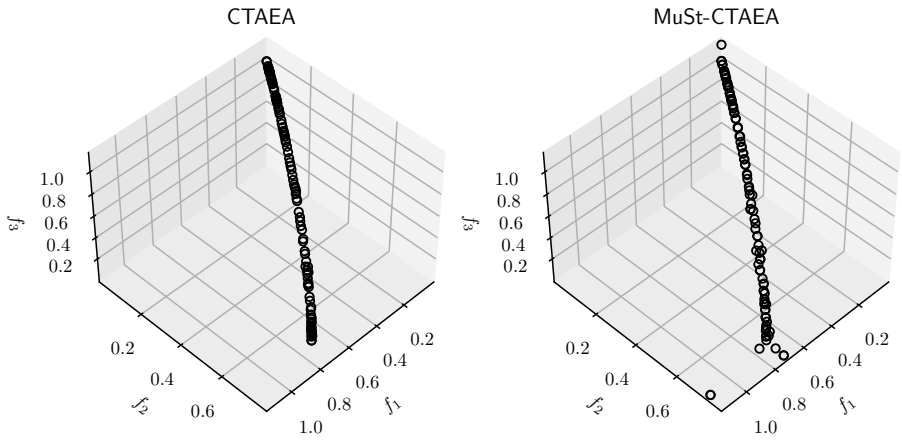

Figure A.6. CTAEA and MuSt-C-TAEA results for the DTLZ5 problem. MuSt-C-TAEA uses 1,785 RV's on Stage 3, $i = 2$. Results are shown for the median HV run for each case.

A.3.4 MaF01 Problem

Results on the MaF01 (same as the inverted DTLZ1 Deb and Jain [2014]) problem is presented in Table A.6 with a number of base algorithms and their multi-stage versions: NSGA-III, MOEA/Dm C-TAEA, two versions of CLIA.

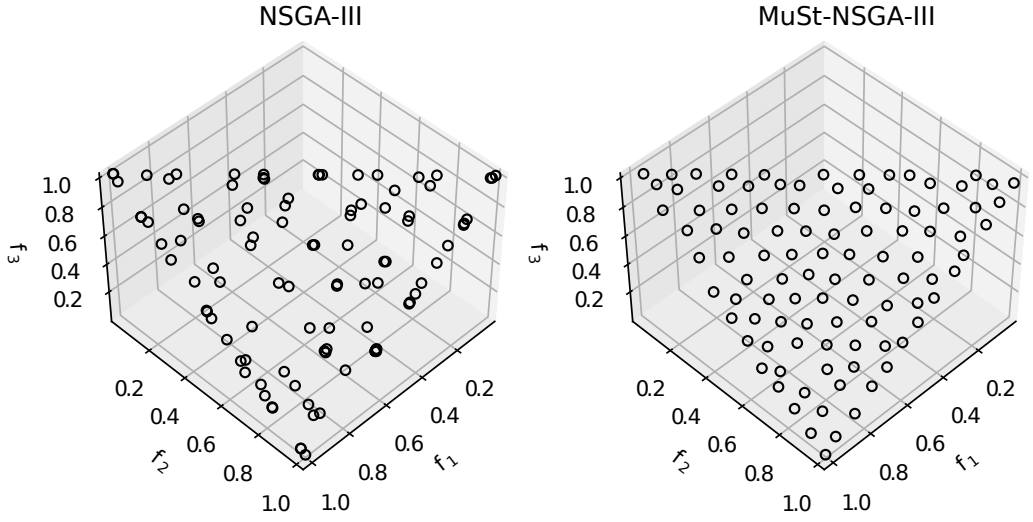

Figure A.7. NSGA-III and MuSt-NSGA-III results are presented for the MaF01 problem. 50 trials are executed and the ND set for the median HV run is plotted.

Table A.6. MaF01 problem set with $M = 3$. The experiment involves 50 trials, and each algorithm has set with $T_S = 20,000$. Some quality indicators are presented.

Algorithm	*N*	*HV* ↑	*MIP-DoM* ↓	*k-neighbors distance*		*SP* ↓	*UD* ↑	ξ ↓
				mean	*std dev*			
NSGA-III	100	0.0146	1.7711	0.0175	0.0147	0.0227	0.5329	0.8066
MuSt-NSGA-III	100	**0.0162**	**1.7393**	**0.0439**	**0.0073**	**0.0102**	**0.9748**	**0.2392**
MOEA/D	97	0.0129	1.7946	0.0261	0.0269	0.0409	0.4709	0.8589
MuSt-MOEA/D	100	**0.0161**	**1.7473**	**0.0424**	**0.0085**	**0.0122**	**0.9282**	**0.2808**
C-TAEA	100	0.0153	***1.7449***	0.0289	0.0130	0.0205	0.5552	0.5800
MS-C-TAEA	100	**0.0162**	*1.7770*	**0.0450**	**0.0079**	**0.0108**	**0.9646**	**0.2233**
CLIA	100	***0.0164***	1.8025	0.0345	0.0128	0.0198	0.7148	0.4338
MuSt-CLIA	100	***0.0164***	**1.7332**	**0.0450**	**0.0085**	**0.0110**	**0.9687**	**0.2318**
CLIA	100	*0.0163*	1.8537	0.0344	0.0128	0.0199	0.7123	0.4323
MuSt-CC	100	***0.0164***	**1.6558**	**0.0438**	**0.0087**	**0.0120**	**0.9798**	**0.2466**

We applied the Wilcoxon signed-rank test with a significance level of 0.05. Data in italics means that it is not possible to detect differences between algorithms. Data in bold indicates the best value.

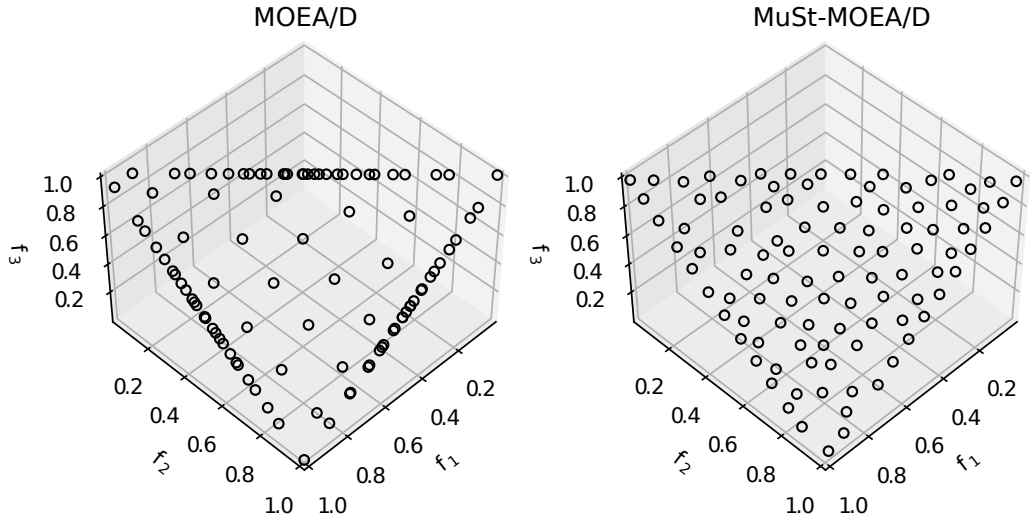

Figure A.8. MOEA/D and MuSt-MOEA/D results are presented for the MaF01 problem. 50 trials are executed and the ND set for the median HV run is plotted.

Since MOEA/D is able to produce 97 ND points (close to desired number 100), we compute the performance metrics. Not only does the MuSt-MOEA/D finds better performance metrics, it also does so with more ND points.

Figures A.7 to A.11 present the median HV solution sets , for NSGA-III, MOEA/D, C-TAEA, CLIA and MuSt-CC , respectively. In all cases, the ND sets from the multi-stage framework are better distributed than their baseline versions. ND points from the original CLIA comes closer to MuSt-CLIA, but the points from the latter is more uniformly distributed visually as well as with respect to the performance metrics shown in Table A.6.

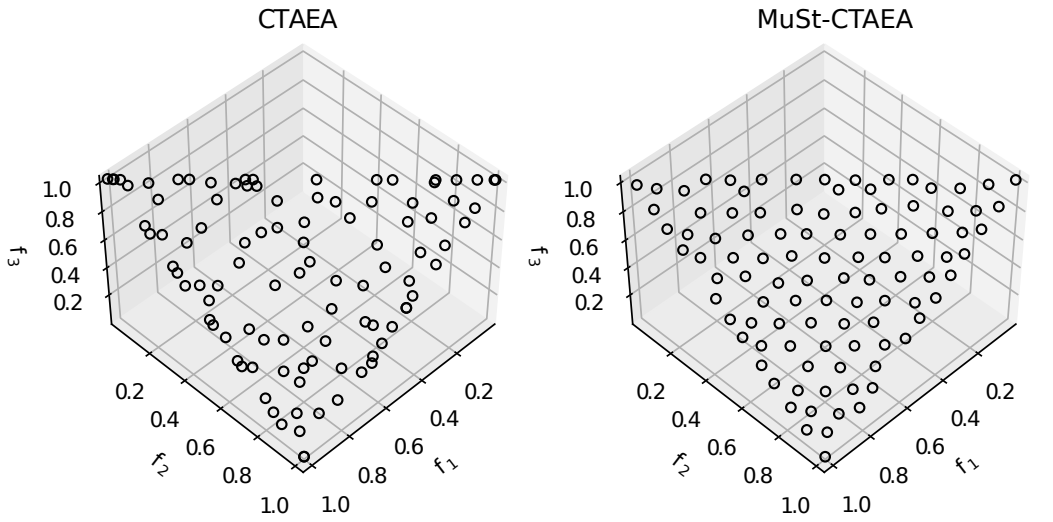

Figure A.9. C-TAEA and MuSt-C-TAEA results are presented for the MaF01 problem. 50 trials are executed and the ND set for the median HV run is plotted.

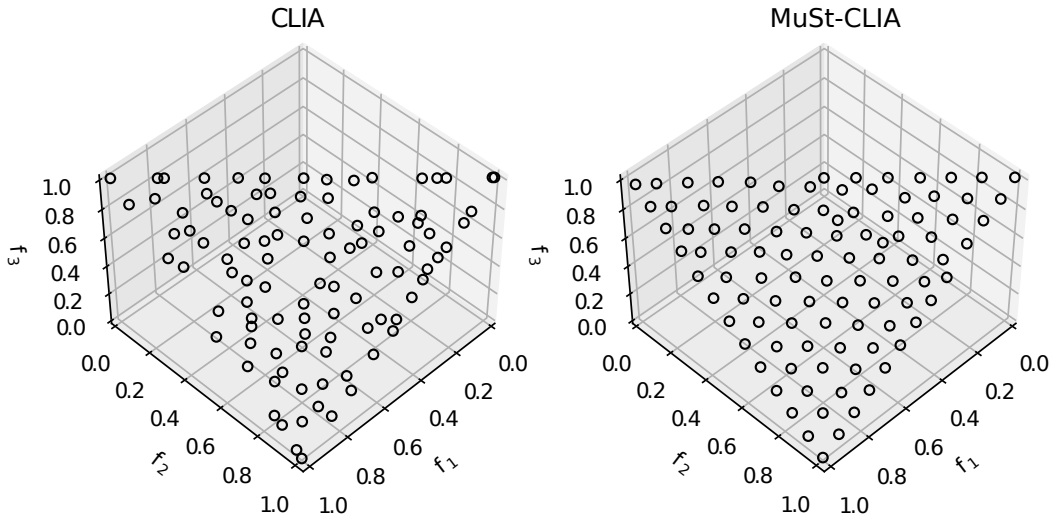

Figure A.10. CLIA and MuSt-CLIA results are presented for the MaF01 problem. 50 trials are executed and the ND set for the median HV run is plotted.

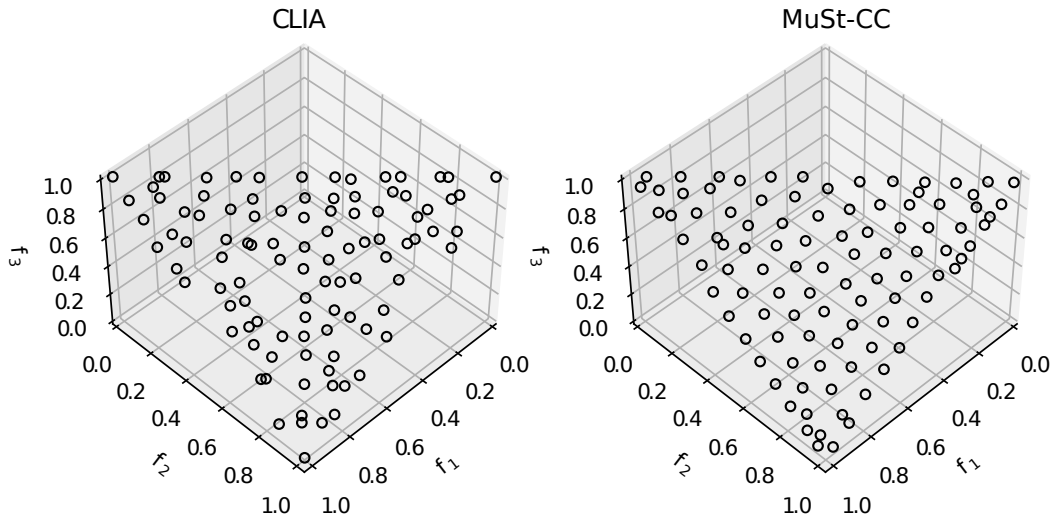

Figure A.11. CLIA and MuSt-CC results are presented for the MaF01 problem. 50 trials are executed and the ND set for the median HV run is plotted.

A.3.5 MaF07 Problem

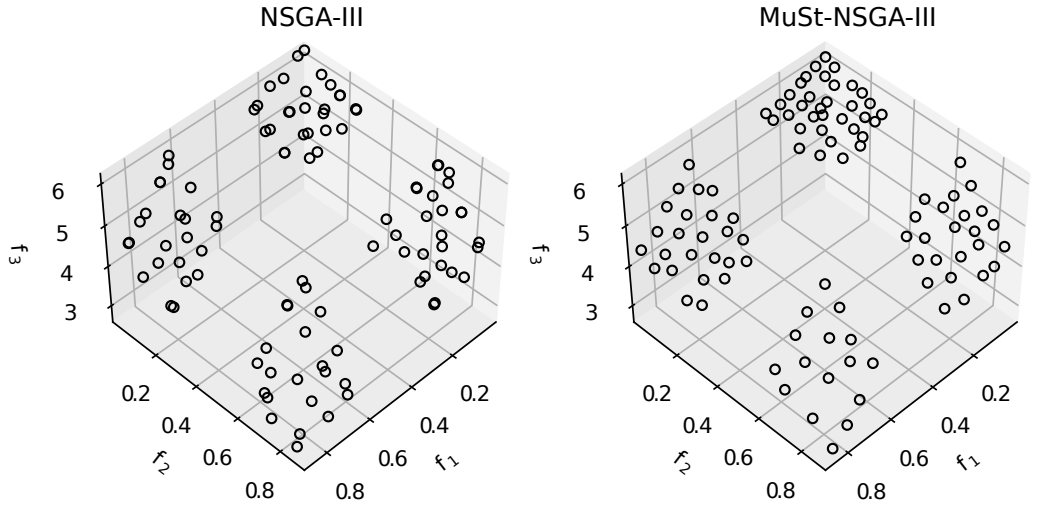

Figure A.12. NSGA-III and MuSt-NSGA-III results are presented for the MaF07 problem. 50 trials are executed and the ND set for the median HV run is plotted.

Resuls on the MaF07 (same as the original DTLZ7 Deb et al. [2002]) problem is presented in Table A.7 for the five base algorithms and their multi-stage versions: NSGA-III, MOEA/D, C-TAEA, and two versions of CLIA. This problem produces four disjointed set of efficient points. Since MOEA/D cannot find close to 100 desired ND

solutions, we do not compare its performance with its multi-stage version, which is able to find 100 desired ND points.

Table A.7. MaF07 problem set with $M = 3$. The experiment involves 50 trials, and each algorithm has set with $T_S = 20,000$. Some quality indicators are presented.

Algorithm	***N***	***HV*** ↑	***MIP-DoM*** ↓	***k-neighbors distance***		***SP*** ↓	***UD*** ↑	ξ ↓
				mean	***std dev***			
NSGA-III	100	0.2503	*1.4809*	0.0254	0.0179	0.0279	0.6400	0.9236
MuSt-NSGA-III	100	**0.2551**	***1.4508***	**0.0498**	**0.0133**	**0.0190**	**0.8781**	**0.6991**
MOEA/D	85	-	-	-	-	-	-	-
MuSt-MOEA/D	100	0.2205	1.1091	0.0374	0.0148	0.0229	0.7366	0.7080
C-TAEA	100	0.2603	1.3762	0.0272	0.0224	0.0335	0.4616	1.1244
MS-C-TAEA	100	**0.2702**	**1.3036**	**0.0467**	**0.0143**	**0.0211**	**0.8916**	**0.7503**
CLIA	100	*0.2076*	*1.5099*	*0.0314*	0.0221	0.0341	0.4909	0.9842
MuSt-CLIA	100	***0.2218***	***1.3957***	***0.0441***	**0.0110**	**0.0164**	**0.8425**	**0.7239**
CLIA	100	*0.2066*	1.6067	0.0310	0.0225	0.0347	0.5065	0.9870
MuSt-CC	100	***0.2162***	**1.2410**	**0.0436**	**0.0110**	**0.0163**	**0.8264**	**0.7390**

We applied the Wilcoxon signed-rank test with a significance level of 0.05. Data in italics means that it is not possible to detect differences between *algorithm* and *MuSt-EMaO* . Data in bold indicates the best value.
MOEA/D algorithm was not able to generate 95 solutions. In this sense, we decide not to compare the quality indicators.

For all problems, the performance of the multi-stage version is better or equivalent than their original versions.

Figures A.12 to A.16 show the ND points for the median HV run for NSGA-III, MOEA/D, C-TAEA, CLIA and MuSt-CC algorithms and their multi-stage versions, respectively. Clearly, the distribution of points for each case is better (more points and more uniformly distributed) than the original algorithm.

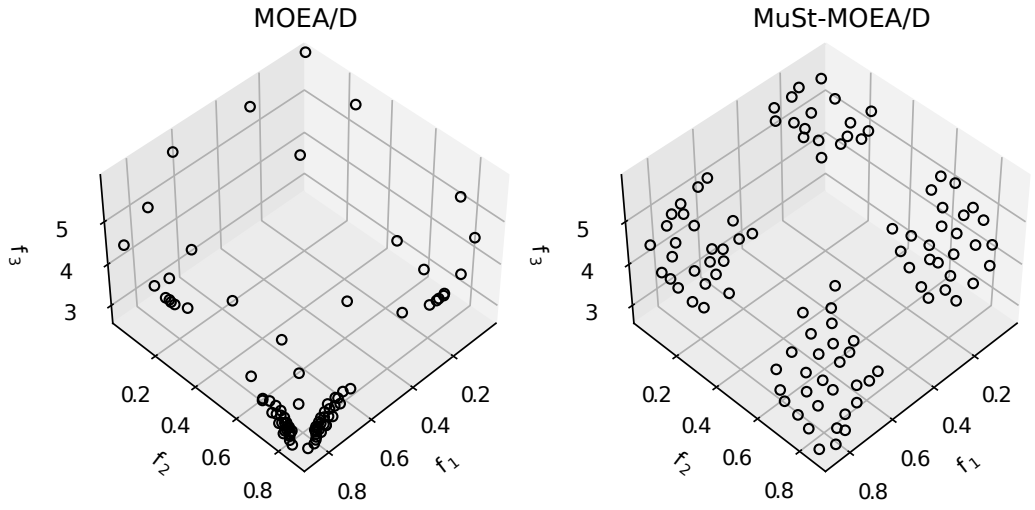

Figure A.13. MOEA/D and MuSt-MOEA/D results are presented for the MaF07 problem. 50 trials are executed and the ND set for the median HV run is plotted.

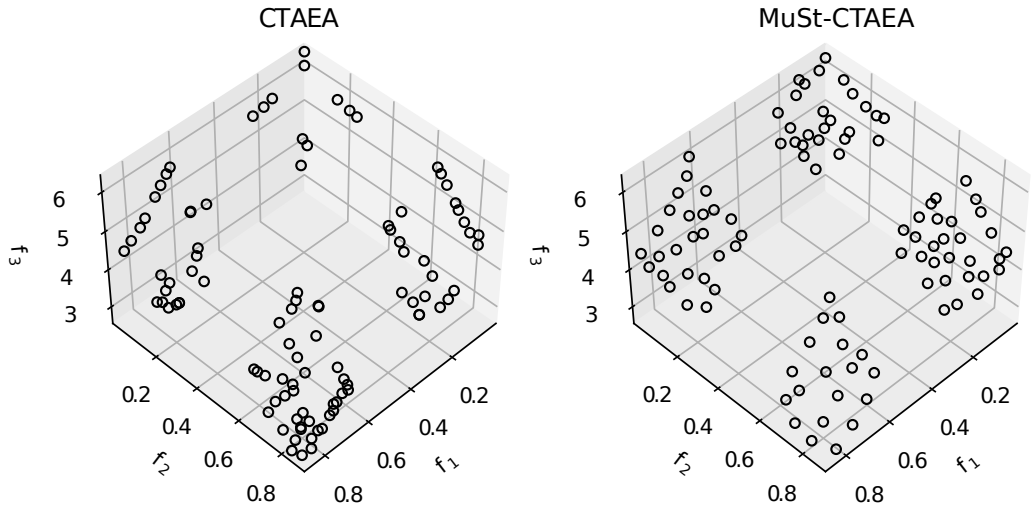

Figure A.14. C-TAEA and MuSt-C-TAEA results are presented for the MaF07 problem. 50 trials are executed and the ND set for the median HV run is plotted.

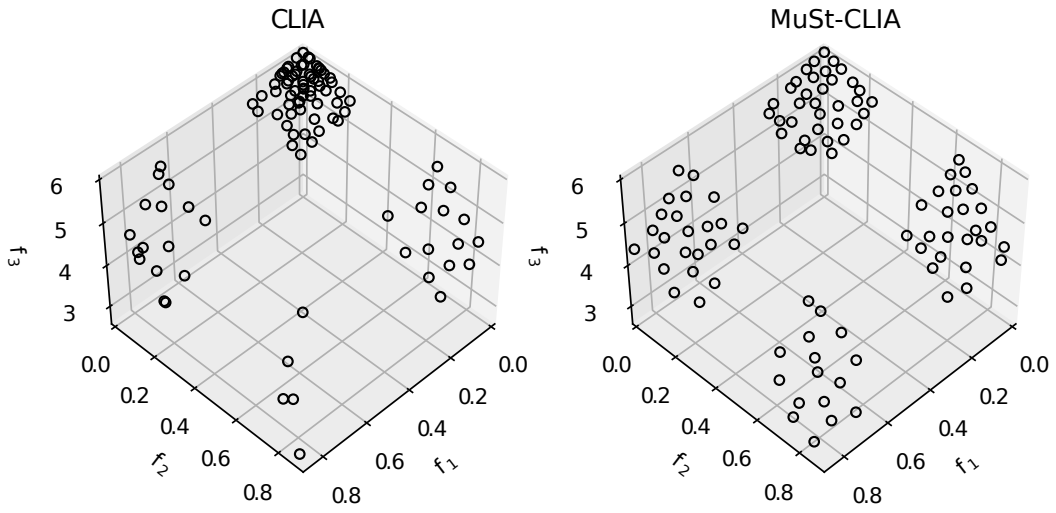

Figure A.15. CLIA and MuSt-CLIA results are presented for the MaF07 problem. 50 trials are executed and the ND set for the median HV run is plotted.

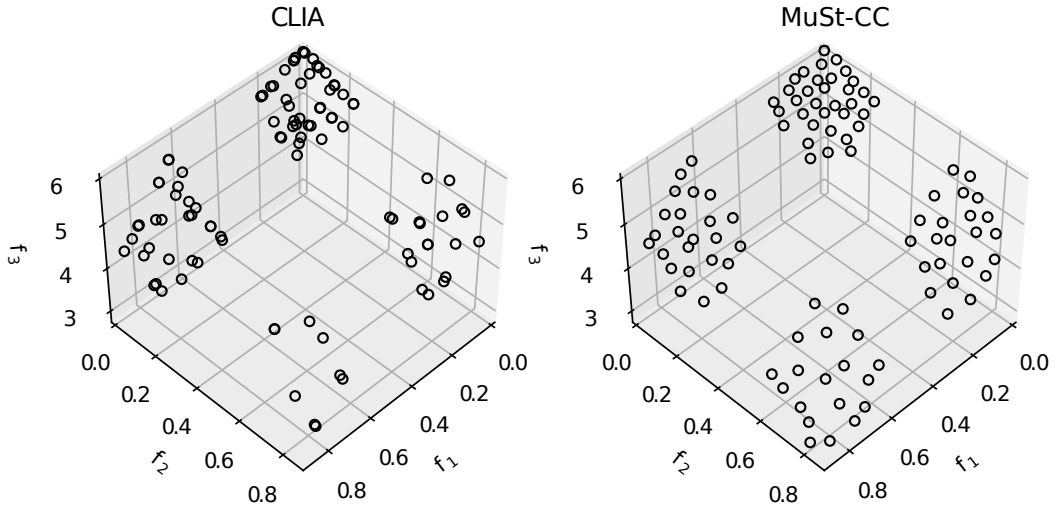

Figure A.16. CLIA and MuSt-CC results are presented for the MaF07 problem. 50 trials are executed and the ND set for the median HV run is plotted.

A.3.6 MaF10 Problem

Results on the MaF10 (similar to convex DTLZ2 Deb [2001]) problem are presented in Table A.8 with five base algorithms and their multi-stage versions: NSGA-III,

MOEA/D, C-TAEA, and two versions of CLIA. This problem produces a convex efficient front.

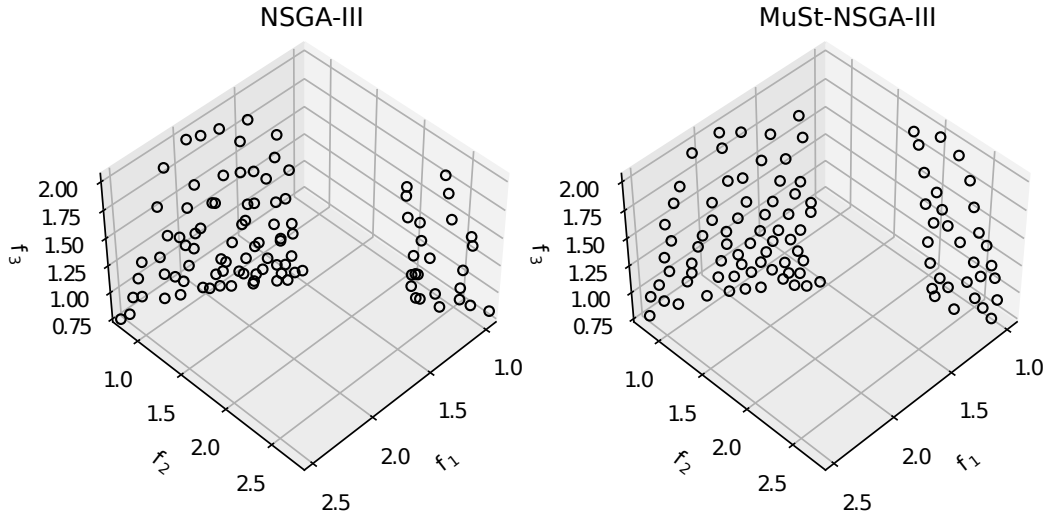

Figure A.17. NSGA-III and MuSt-NSGA-III results are presented for the MaF10 problem. 50 trials are executed and the ND set for the median HV run is plotted.

Table A.8. MaF10 problem set with $M = 3$. The experiment involves 50 trials, and each algorithm has set with $T_S = 20,000$. Some quality indicators are presented.

Algorithm	***N***	***HV*** ↑	***MIP-DoM*** ↓	***k-neighbors distance***		***SP*** ↓	***UD*** ↑	ξ ↓
				mean	***std dev***			
NSGA-III	100	0.5153	***1.0939***	0.0290	0.0171	0.0266	0.5922	*0.8987*
MuSt-NSGA-III	100	**0.6177**	*1.2600*	**0.0500**	**0.0141**	**0.0218**	**0.9359**	***0.7319***
MOEA/D	97	**0.4156**	**1.2049**	**0.0666**	0.0270	0.0425	0.8820	**0.4619**
MuSt-MOEA/D	100	0.4021	1.4494	0.0561	**0.0142**	**0.0221**	**0.9667**	0.5328
CTAEA	100	*0.3046*	**1.4632**	0.0503	0.0253	0.0384	0.7515	0.6150
MuSt-C-TAEA	100	***0.3065***	1.5278	**0.0538**	**0.0180**	**0.0276**	**0.9141**	**0.4838**
CLIA	100	*0.7022*	*1.4997*	0.0607	0.0434	0.0691	0.7532	0.6696
MuSt-CLIA	100	***0.7065***	***1.4441***	**0.0670**	**0.0350**	**0.0567**	**0.9824**	**0.5187**
CLIA	100	0.7463	*1.4975*	*0.0738*	0.0246	0.0379	0.9765	0.4759
MuSt-CC	100	**0.7485**	***1.3939***	***0.0745***	**0.0192**	**0.0304**	**1.0000**	**0.3898**

We applied the Wilcoxon signed-rank test with a significance level of 0.05. Data in italics means that it is not possible to detect differences between algorithms. Data in bold indicates the best value.

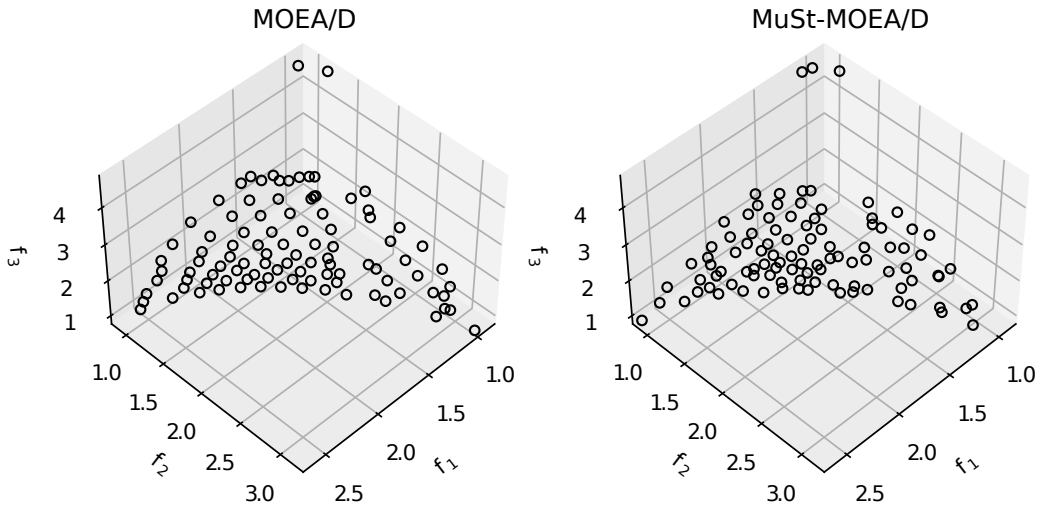

Figure A.18. MOEA/D and MuSt-MOEA/D results are presented for the MaF10 problem. 50 trials are executed and the ND set for the median HV run is plotted.

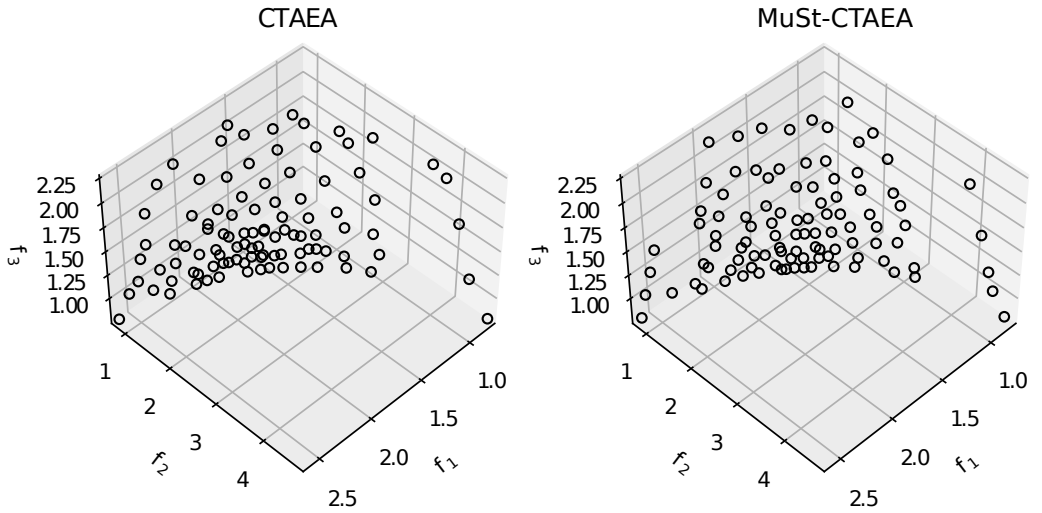

Figure A.19. C-TAEA and MS-C-TAEA sample results from MaF10. 50 trials are done, and the presented sample comes from the HV median value sample.

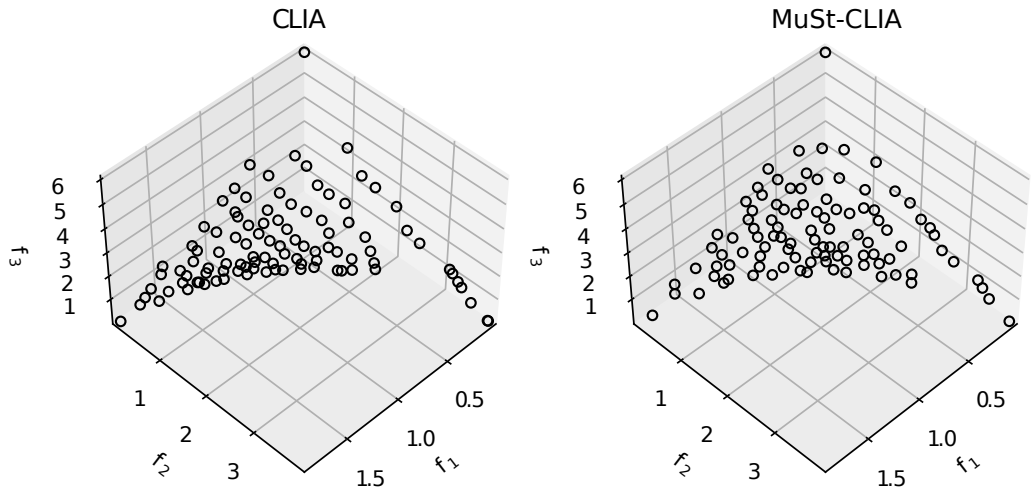

Figure A.20. CLIA and MuSt-CLIA results are presented for the MaF10 problem. 50 trials are executed and the ND set for the median HV run is plotted.

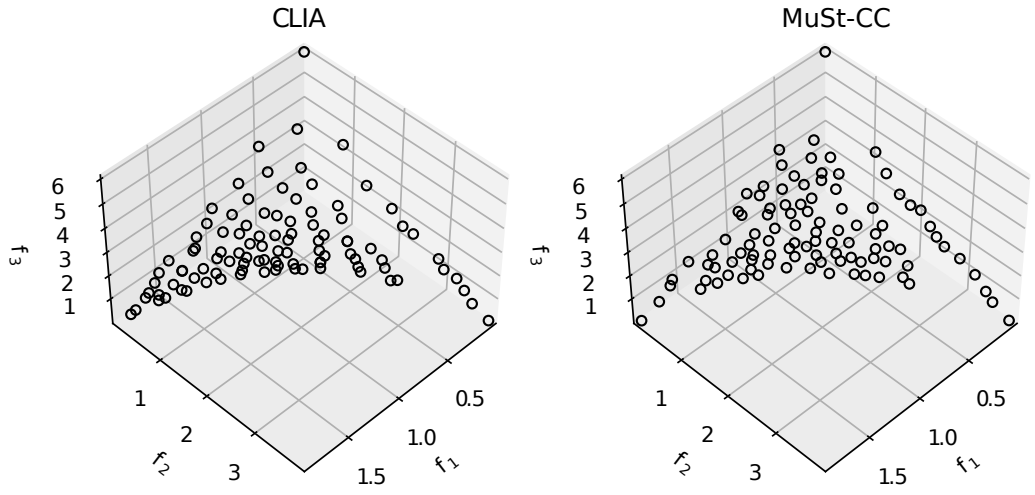

Figure A.21. CLIA and MuSt-CC results are presented for the MaF10 problem. 50 trials are executed and the ND set for the median HV run is plotted.

A.3.7 MaF11 Problem

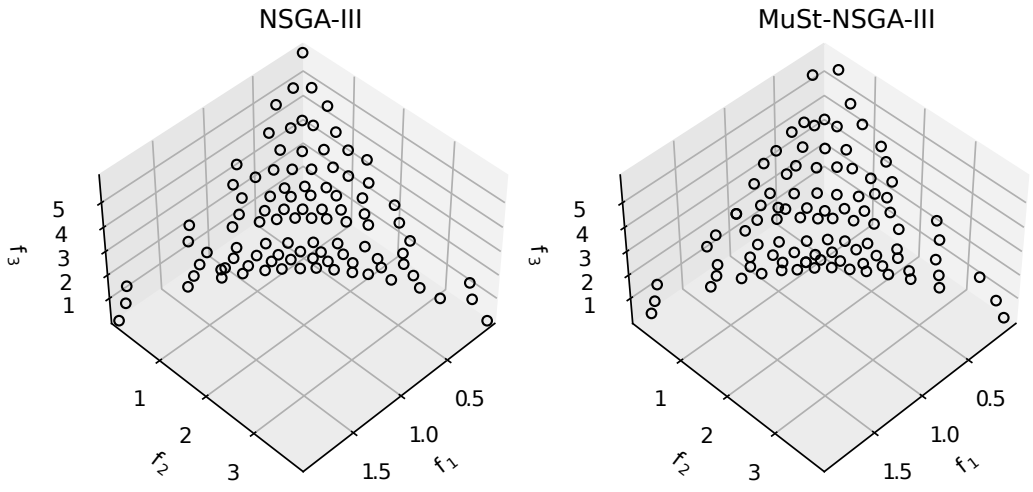

Figure A.22. NSGA-III and MuSt-NSGA-III results are presented for the MaF11 problem. 50 trials are executed and the ND set for the median HV run is plotted.

For this problem, some performance metrics are better for the original EMaO algorithm, but in most cases the respective MuSt-EMaO algorithm produces a statistically better or equivalent performance compared to their original versions.

In order to visualize the ND points, Figures A.17 to A.21 present obtained ND points for the median HV run for NSGA-III, MOEA/D, C-TAEA, CLIA and MuSt-CC algorithms, respectively.

Results on the MaF11 problem are presented in Table A.9 with five base algorithms and their multi-stage versions: NSGA-III, MOEA/D, C-TAEA, and two versions of CLIA.

Table A.9. MaF11 problem set with $M = 3$. The experiment involves 50 trials, and each algorithm has set with $T_S = 20,000$. Some quality indicators are presented.

Algorithm	***N***	***HV*** ↑	***MIP-DoM*** ↓	***k-neighbors distance***		***SP*** ↓	***UD*** ↑	ξ ↓
				mean	***std dev***			
NSGA-III	100	*0.7273*	*1.5877*	*0.0855*	0.0198	0.0282	0.9030	0.2285
MuSt-NSGA-III	100	***0.7371***	***1.5733***	***0.0868***	**0.0146**	**0.0204**	**0.9868**	**0.2098**
MOEA/D	85	-	-	-	-	-	-	-
MuSt-MOEA/D	100	0.3593	0.7533	0.0535	0.0103	0.0162	0.6854	0.5377
C-TAEA	100	0.5782	***1.3290***	0.0608	0.0297	0.0446	0.7901	0.6010
MS-C-TAEA	100	**0.7500**	*1.3526*	**0.0675**	**0.0174**	**0.0264**	**0.9696**	**0.4102**
CLIA	100	0.7389	***1.5326***	0.0727	0.0264	0.0408	*0.9729*	0.4795
MuSt-CLIA	100	**0.7406**	*1.5397*	**0.0737**	**0.0195**	**0.0303**	***0.9803***	**0.3794**
CLIA	100	0.7398	***1.5356***	*0.0723*	0.0258	0.0399	*0.9819*	0.4827
MuSt-CC	100	**0.7421**	*1.5423*	***0.0743***	**0.0188**	**0.0293**	***0.9893***	**0.3746**

We applied the Wilcoxon signed-rank test with a significance level of 0.05. Data in italics means that it is not possible to detect differences between *algorithm* and *MuSt-EMaO* . Data in bold indicates the best value.
MOEA/D algorithm was not able to generate 95 solutions. In this sense, we decide not to compare the quality indicators.

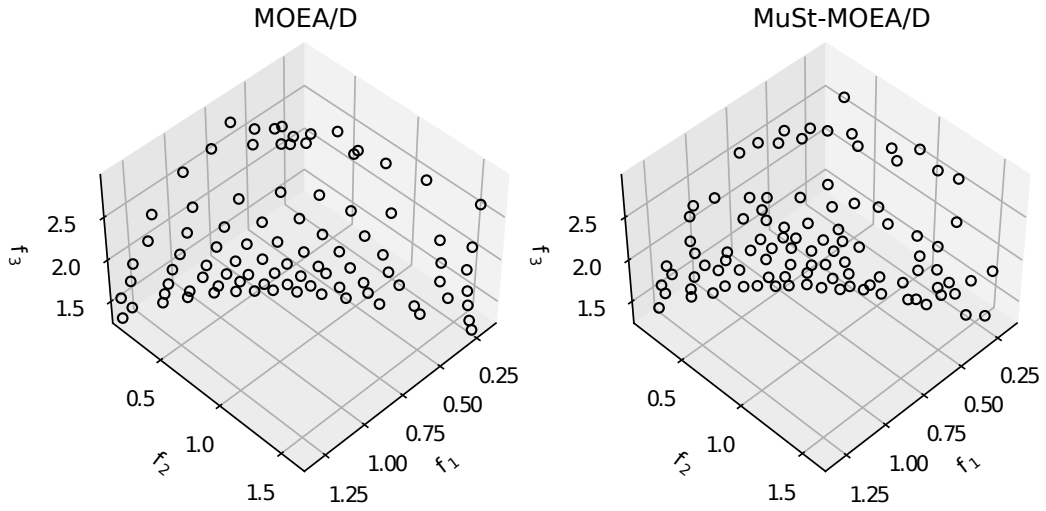

Figure A.23. MOEA/D and MuSt-MOEA/D results are presented for the MaF11 problem. 50 trials are executed and the ND set for the median HV run is plotted.

MOEA/D could not produce close to 100 ND solutions; hence its performance is

not compared with its multi-stage version. Performance for the multi-stage version of each algorithm is statistically better than its original version.

Figures A.22 to A.26 present the ND points for the median HV run for NSGA-III, MOEA/D, C-TAEA, CLIA and MuSt-CC , respectively.

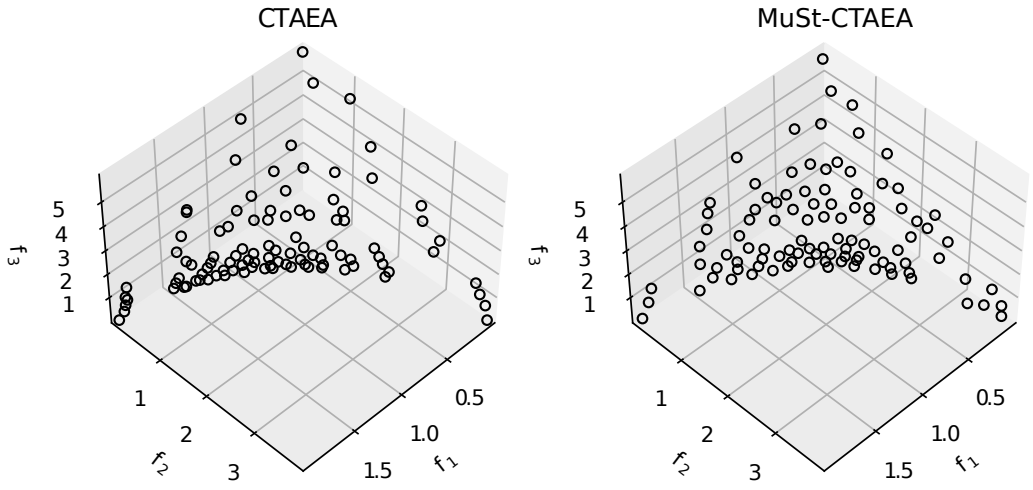

Figure A.24. C-TAEA and MuSt-C-TAEA results are presented for the MaF11 problem. 50 trials are executed and the ND set for the median HV run is plotted.

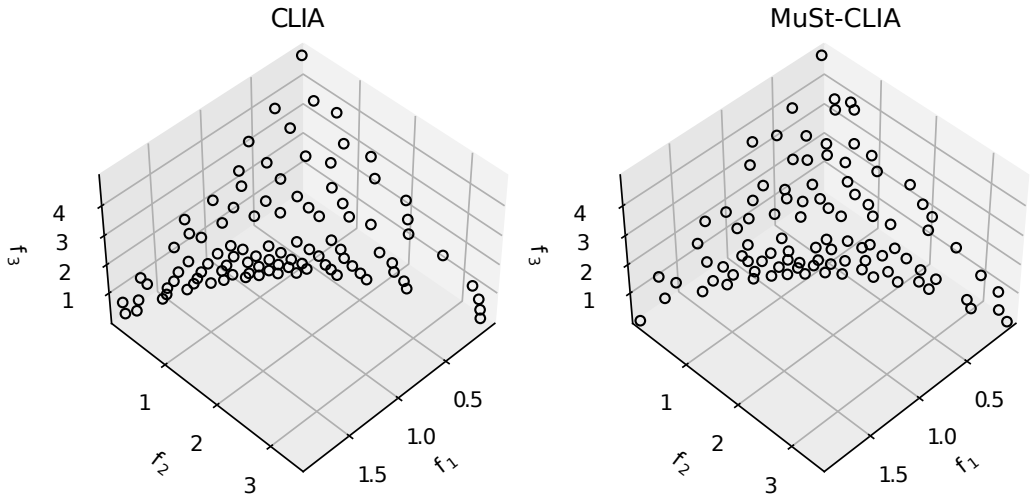

Figure A.25. CLIA and MuSt-CLIA results are presented for the MaF11 problem. 50 trials are executed and the ND set for the median HV run is plotted.

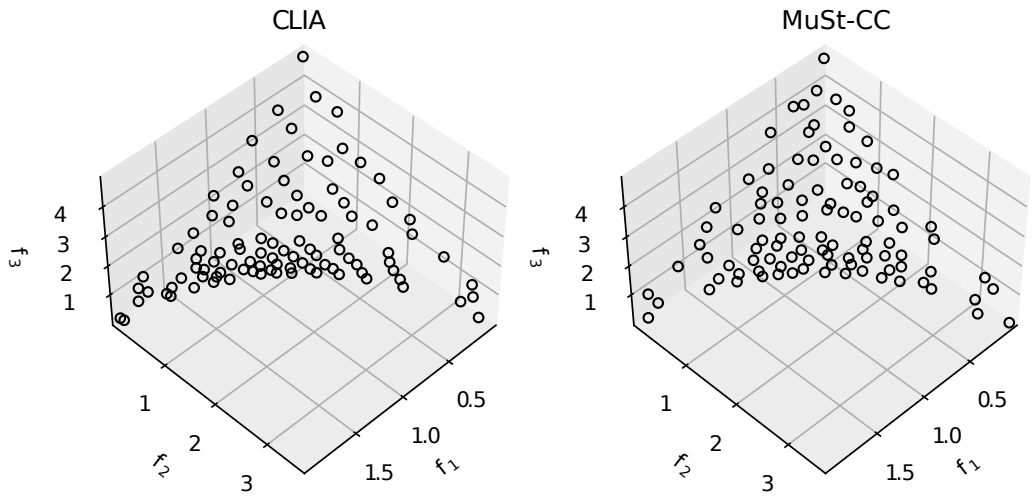

Figure A.26. CLIA and MuSt-CC results are presented for the MaF11 problem. 50 trials are executed and the ND set for the median HV run is plotted.

A.3.8 MaF12 Problem

Results on the MaF11 problem are presented in Table A.10 with five base algorithms and their multi-stage versions: NSGA-III, MOEA/D, C-TAEA, and two versions of CLIA.

MOEA/D could not produce close to 100 ND solutions; hence its performance is not compared with its multi-stage version. Performance for the multi-stage version of each algorithm is statistically better than its original version.

Figures A.27 to A.31 present the ND points for the median HV run for NSGA-III, MOEA/D, C-TAEA, CLIA and MuSt-CC algorithms and their multi-stage versions, respectively. For this algorithm, the distributions of original EMaO algorithms are reasonably good, except a denser set points can be observed for small f_2-f_3 part of the efficient set. In all cases, performance of the multi-stage version is superior.

Table A.10. MaF12 problem set with $M = 3$. The experiment involves 50 trials, and each algorithm has set with $T_S = 20,000$. Some quality indicators are presented.

Algorithm	***N***	***HV*** ↑	***MIP-DoM*** ↓	***k-neighbors distance***		***SP*** ↓	***UD*** ↑	ξ ↓
				mean	***std dev***			
NSGA-III	100	*0.6640*	***2.2017***	0.0837	0.0192	0.0272	0.9049	0.2311
MuSt-NSGA-III	100	***0.6646***	*2.2286*	**0.0858**	**0.0141**	**0.0194**	**0.9951**	**0.2119**
MOEA/D	85	-	-	-	-	-	-	-
MuSt-MOEA/D	100	0.5183	2.2671	0.0627	0.0165	0.0258	0.9785	0.4116
CTAEA	100	0.5171	***2.1397***	0.0646	0.0286	0.0423	0.7668	0.5209
MuSt-C-TAEA	100	**0.5199**	*2.1451*	**0.0670**	**0.0175**	**0.0267**	**0.9940**	**0.4022**
CLIA	100	0.7090	***2.0211***	0.0726	0.0285	0.0419	0.8697	0.5006
MuSt-CLIA	100	**0.7114**	*2.0221*	**0.0735**	**0.0180**	**0.0267**	**0.9941**	**0.3794**
CLIA	100	0.7265	2.0776	*0.0747*	0.0254	0.0366	0.9226	0.4737
MuSt-CC	100	**0.7278**	**2.0093**	***0.0750***	**0.0184**	**0.0266**	**0.9998**	**0.3846**

We applied the Wilcoxon signed-rank test with a significance level of 0.05. Data in italics means that it is not possible to detect differences between *algorithm* and *MuSt-EMaO* . Data in bold indicates the best value.

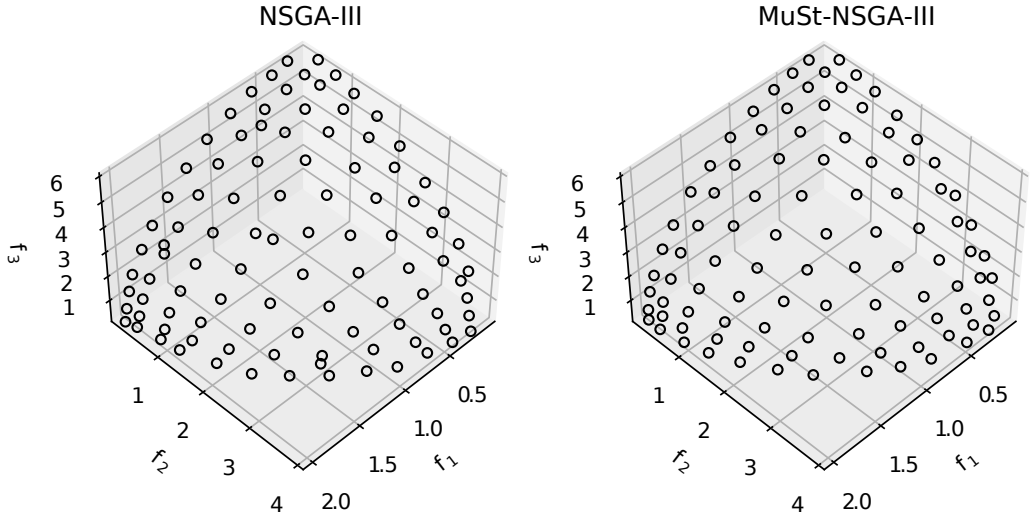

Figure A.27. NSGA-III and MuSt-NSGA-III results are presented for the MaF12 problem. 50 trials are executed and the ND set for the median HV run is plotted.

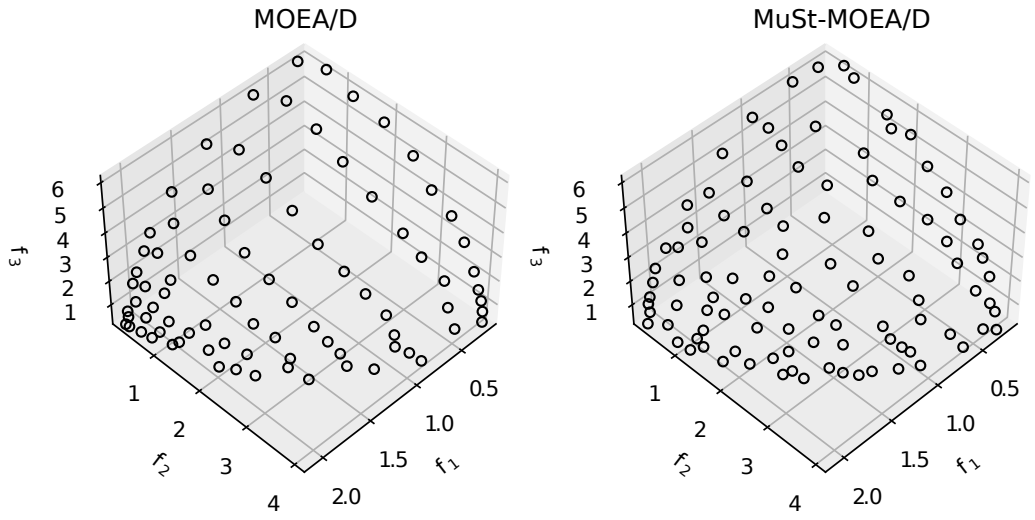

Figure A.28. MOEA/D and MuSt-MOEA/D results are presented for the MaF12 problem. 50 trials are executed and the ND set for the median HV run is plotted.

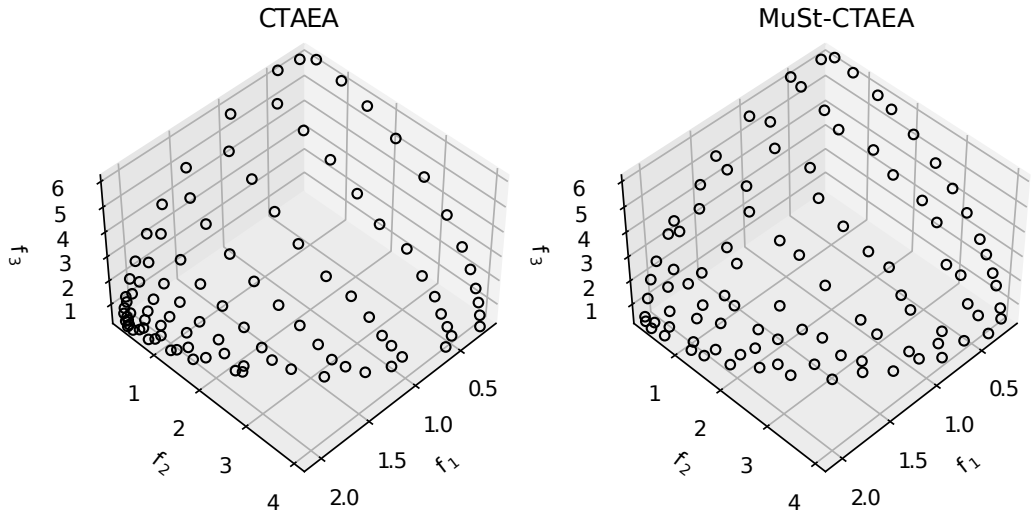

Figure A.29. C-TAEA and MuSt-C-TAEA results are presented for the MaF12 problem. 50 trials are executed and the ND set for the median HV run is plotted.

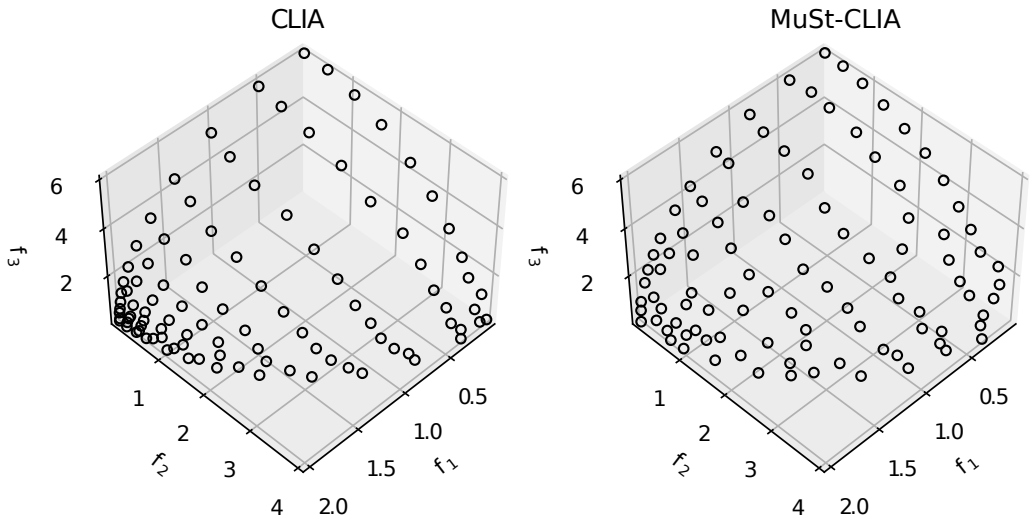

Figure A.30. CLIA and MuSt-CLIA results are presented for the MaF12 problem. 50 trials are executed and the ND set for the median HV run is plotted.

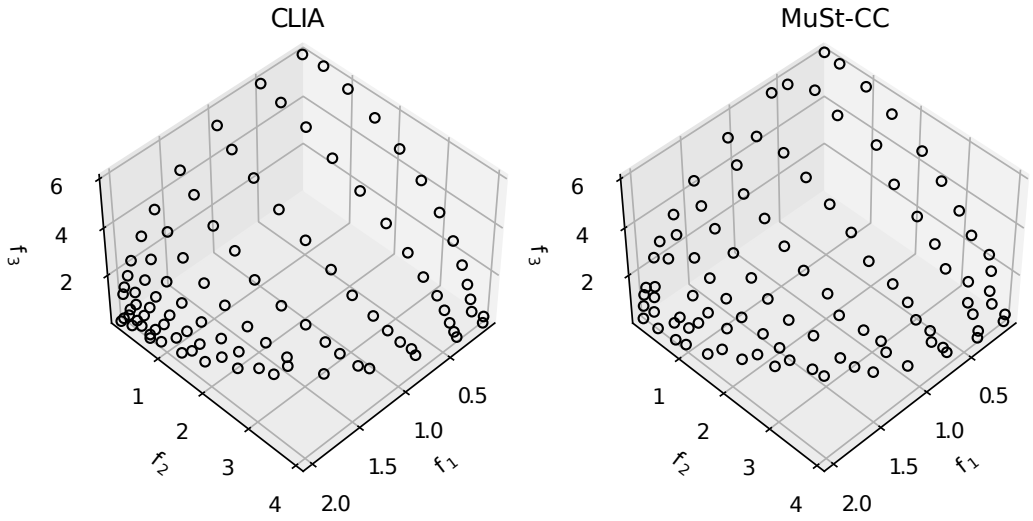

Figure A.31. CLIA and MuSt-CC results are presented for the MaF12 problem. 50 trials are executed and the ND set for the median HV run is plotted.

A.3.9 C2-DTLZ2 Problem

Results on the C2-DTLZ2 problem – constrained three-objective problem Jain and Deb [2014] – are presented in Table A.11 with three original EMaO algorithms and with their multi-stage versions: C-TAEA, MOEA/D, and NSGA-III.

Table A.11. C2DTLZ2 problem set with $M = 3$. The experiment involves 50 trials, and each algorithm has set with $T_S = 20,000$. Some quality indicators are presented.

Algorithm	*N*	*HV* ↑	*MIP-DoM* ↓	*k-neighbors distance*		*SP* ↓	*UD* ↑	ξ ↓
				mean	*std dev*			
NSGA-III	100	0.7379	0.9331	0.0573	0.0365	0.0540	0.6965	0.4780
MuSt-NSGA-III	100	**0.7476**	**0.9221**	**0.0710**	**0.0140**	**0.0199**	**1.0000**	**0.3506**
MOEA/D	57	-	-	-	-	-	-	-
MuSt-MOEA/D	100	0.7505	1.0164	0.0570	0.0137	0.0214	0.9922	0.5021
CTAEA	100	0.7610	0.9329	0.0589	0.0323	0.0488	0.7297	0.4544
MuSt-C-TAEA	100	**0.7638**	**0.9088**	**0.0677**	**0.0144**	**0.0211**	**0.9883**	**0.3686**

We applied the Wilcoxon signed-rank test with a significance level of 0.05. Data in italics means that it is not possible to detect differences between *algorithm* and *MuSt-EMaO* . Data in bold indicates the best value.

MOEA/D could not produce close to 100 ND solutions; hence its performance is not compared with its multi-stage version. Performance for the multi-stage version of each algorithm is statistically better than its original version.

Figures A.32 to A.34 present the ND points for the median HV run for NSGA-III, MOEA/D, and C-TAEA algorithms and their multi-stage versions, respectively. For each algorithm, the distribution of multi-stage version is more uniform and contains more ND points than its original version.

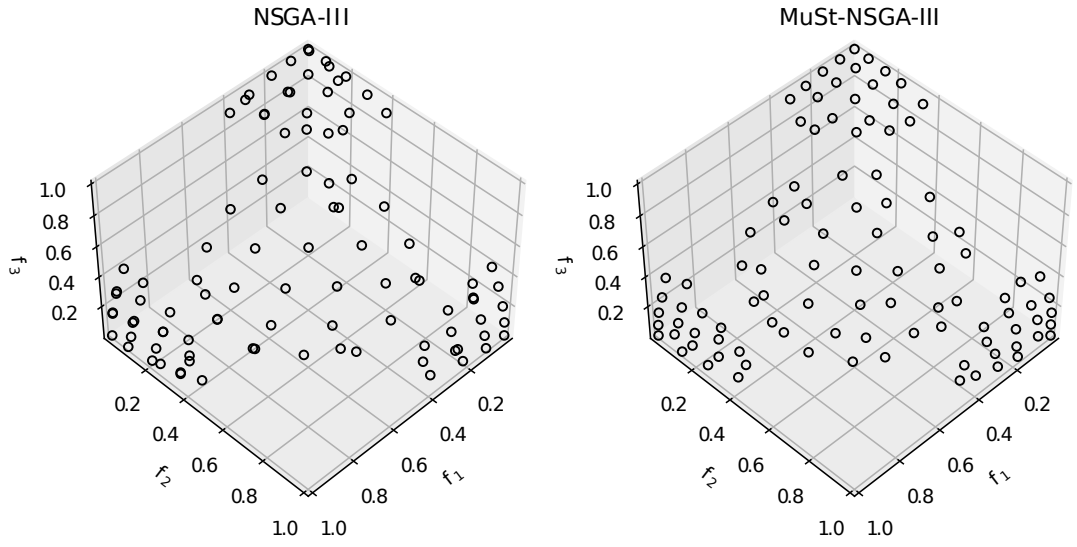

Figure A.32. NSGA-III and MuSt-NSGA-III results are presented for the C2-DTLZ2 problem. 50 trials are executed and the ND set for the median HV run is plotted.

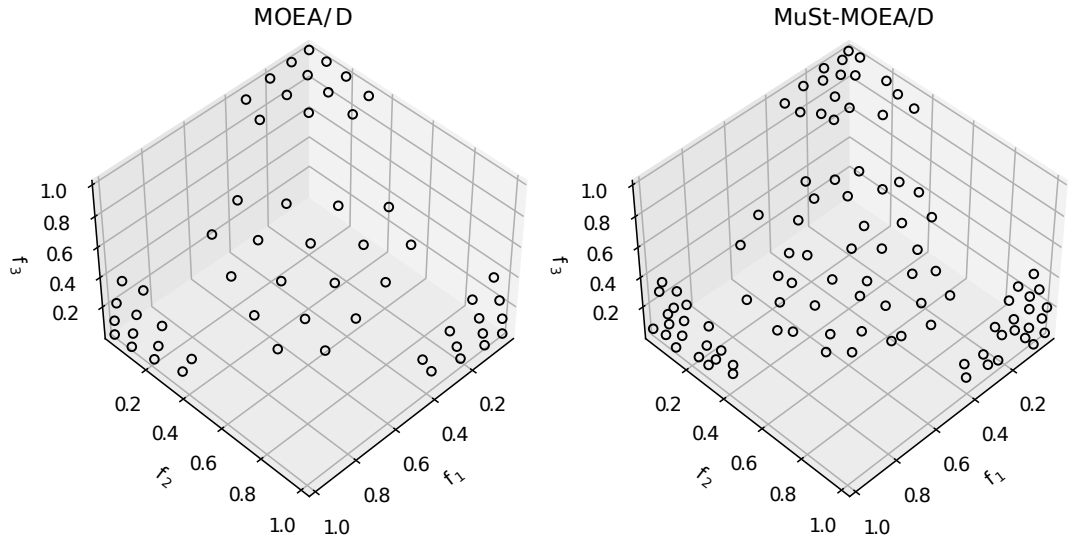

Figure A.33. MOEA/D and MuSt-MOEA/D results are presented for the C2-DTLZ2 problem. 50 trials are executed and the ND set for the median HV run is plotted.

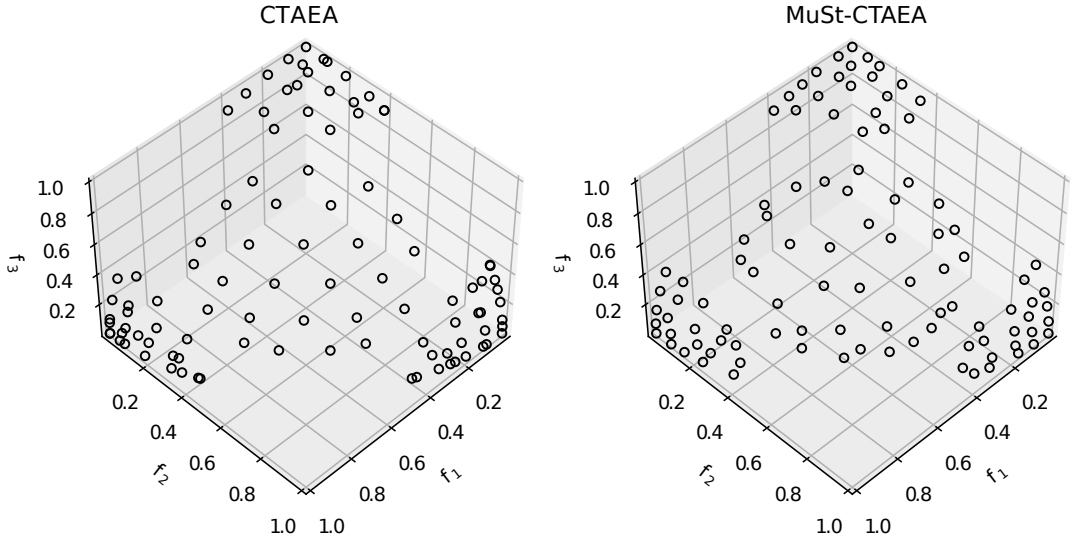

Figure A.34. CTAEA and MuSt-C-TAEA results are presented for the C2-DTLZ2 problem. 50 trials are executed and the ND set for the median HV run is plotted.

A.3.10 Crashworthiness Problem

Results on the crashworthiness problem – constrained three-objective problem Jain and Deb [2014] – are presented in Table A.12 with three original EMaO algorithms and with their multi-stage versions: C-TAEA, MOEA/D, and NSGA-III.

Table A.12. Crashworthiness problem set. The experiment involves 50 trials, and each algorithm has set with $T_S = 20,000$. Some quality indicators are presented.

Algorithm	***N***	***HV*** ↑	***MIP-DoM*** ↓	***k-neighbors distance***		***SP*** ↓	***UD*** ↑	ξ ↓
				mean	***std dev***			
NSGA-III	100	0.1101	1.4844	0.0195	0.0166	0.0260	0.5440	0.9333
MuSt-NSGA-III	100	**0.1139**	**1.0682**	**0.0403**	**0.0112**	**0.0169**	**0.7991**	**0.6205**
MOEA/D	95	0.0724	0.6727	*0.0085*	0.0191	0.0306	0.1597	1.4103
MuSt-MOEA/D	100	**0.0729**	**0.4181**	***0.0093***	**0.0065**	**0.0103**	**0.3356**	**0.6550**
CTAEA	100	0.0961	1.6186	0.0199	0.0253	0.0390	0.1824	1.5259
MuSt-C-TAEA	100	**0.1113**	**0.5136**	**0.0357**	**0.0121**	**0.0189**	**0.7440**	**0.7232**

We applied the Wilcoxon signed-rank test with a significance level of 0.05. Data in italics means that it is not possible to detect differences between *algorithm* and *MuSt-EMaO* . Data in bold indicates the best value.

For each algorithm, the performance of the multi-stage version is better than its original version.

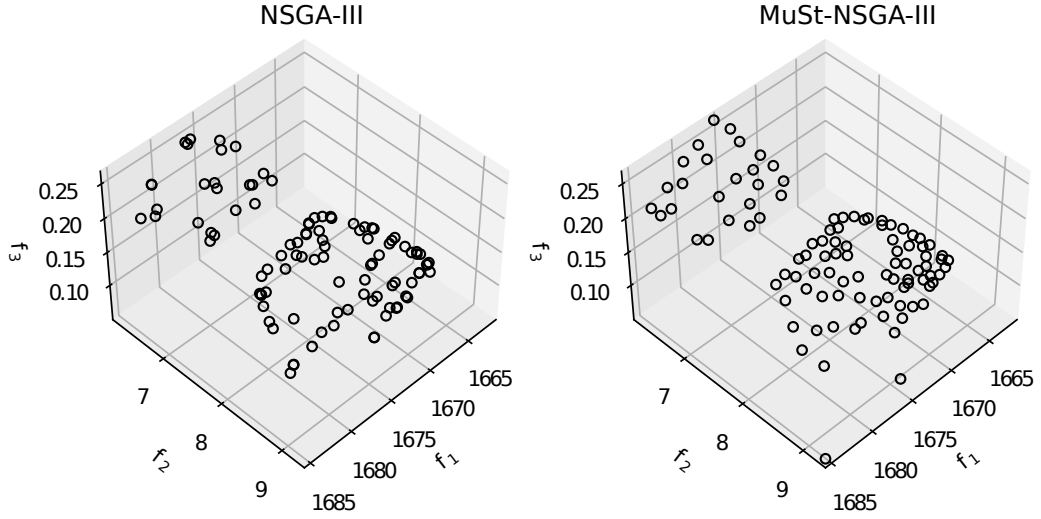

Figure A.35. NSGA-III and MuSt-NSGA-IIIresults are presented for the crashworthiness problem. 50 trials are executed and the ND set for the median HV run is plotted.

Figures A.35 to A.37 present the ND points for the median HV run for NSGA-III, MOEA/D, and C-TAEA algorithms and their multi-stage versions, respectively. For each algorithm, the multi-stage version improves the performance of the original version. MOEA/D is not able to find the entire efficient set and its multi-stage version has also failed to produce the entire efficient set. C-TAEA is also not able to find

the entire efficient set, but its multi-stage version is able to find more points in the region discovered by the original version. MuSt-EMaO algorithm uses the original EMaO algorithm in all three stages. Thus, for a problem if the original EMaO cannot perform well, the multi-stage procedure may improve its performance by finding a better distribution in the regions discovered by the original algorithm.

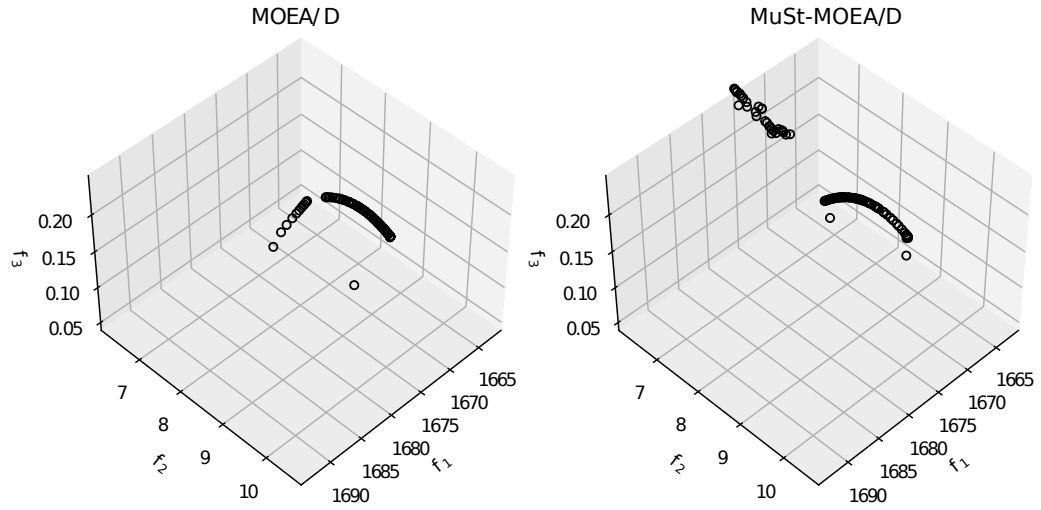

Figure A.36. MOEA/D and MuSt-MOEA/D results are presented for the crashworthiness problem. 50 trials are executed and the ND set for the median HV run is plotted.

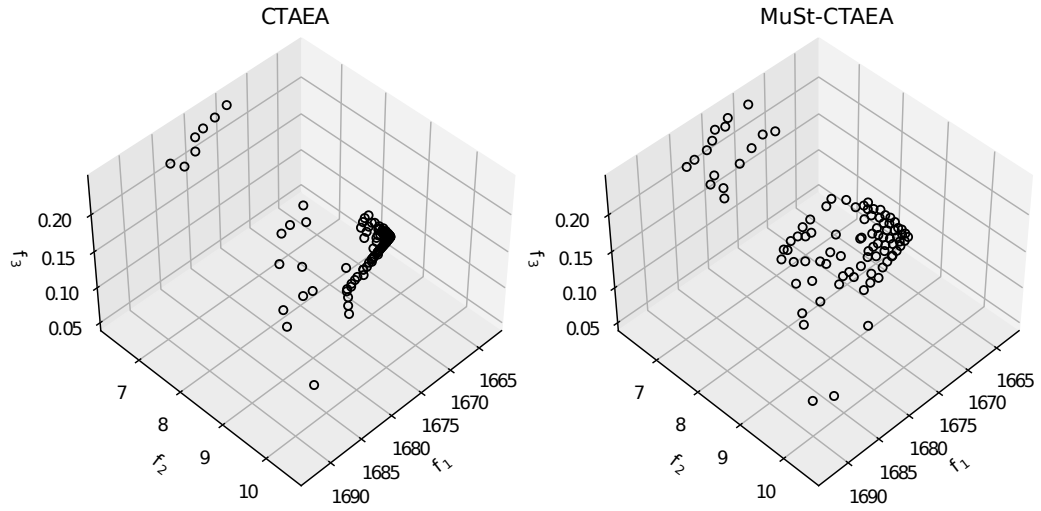

Figure A.37. C-TAEA and MuSt-C-TAEA results are presented for the crashworthiness problem. 50 trials are executed and the ND set for the median HV run is plotted.

A.3.11 MW14 problem

Results on the MW14 problem are presented in Table A.13 with three original EMaO algorithms and with their multi-stage versions: C-TAEA, MOEA/D, and NSGA-III.

Table A.13. MW14 with $M = 3$. The experiment involves 50 trials, and each algorithm has set with $T_S = 40,000$. Some quality indicators are presented.

Algorithm	*N*	*HV* ↑	*MIP-DoM* ↓	*k-neighbors distance*		*SP* ↓	*UD* ↑	ξ ↓
				mean	*std dev*			
NSGA-III	100	**0.2502**	***2.1277***	0.0302	0.0246	0.0374	0.6327	0.7727
MuSt-NSGA-III	100	0.2388	*2.3451*	**0.0475**	**0.0120**	**0.0179**	**0.9150**	**0.6329**
MOEA/D	76	-	-	-	-	-	-	-
MuSt-MOEA/D	100	0.1259	2.0203	0.0319	0.0142	0.0220	0.6462	0.7934
CTAEA	100	*0.2476*	**2.1812**	0.0294	0.0250	0.0376	0.4550	1.0030
MuSt-C-TAEA	100	***0.2486***	2.5768	**0.0472**	**0.0140**	**0.0210**	**0.9268**	**0.6265**

We applied the Wilcoxon signed-rank test with a significance level of 0.05. Data in italics means that it is not possible to detect differences between *algorithm* and *MuSt-EMaO* . Data in bold indicates the best value.

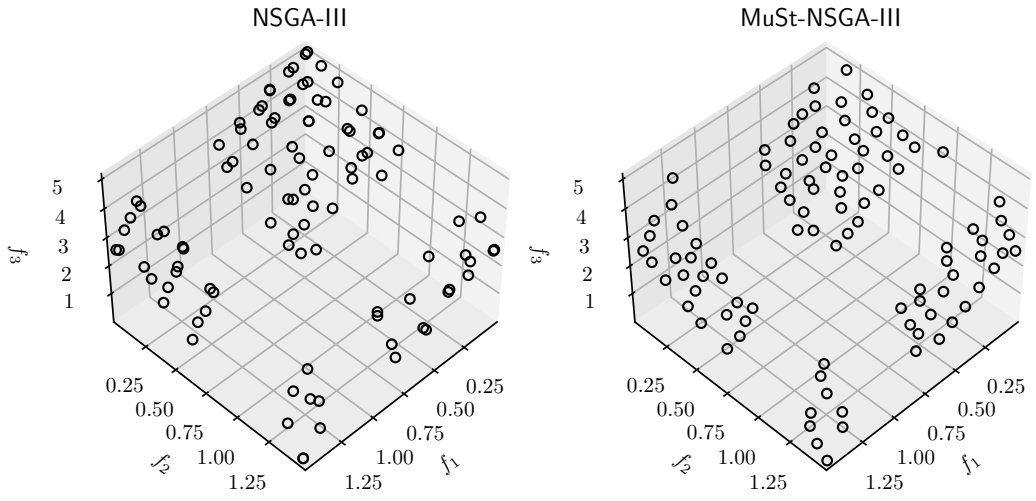

Figure A.38. NSGA-III and MuSt-NSGA-IIIresults are presented for the MW14 problem. 50 trials are executed and the ND set for the median HV run is plotted.

Figures A.38 to A.40 present the ND points for the median HV run for NSGA-III, MOEA/D, and C-TAEA algorithms and their multi-stage versions, respectively.

For each algorithm, the multi-stage version improves the performance of the original version.

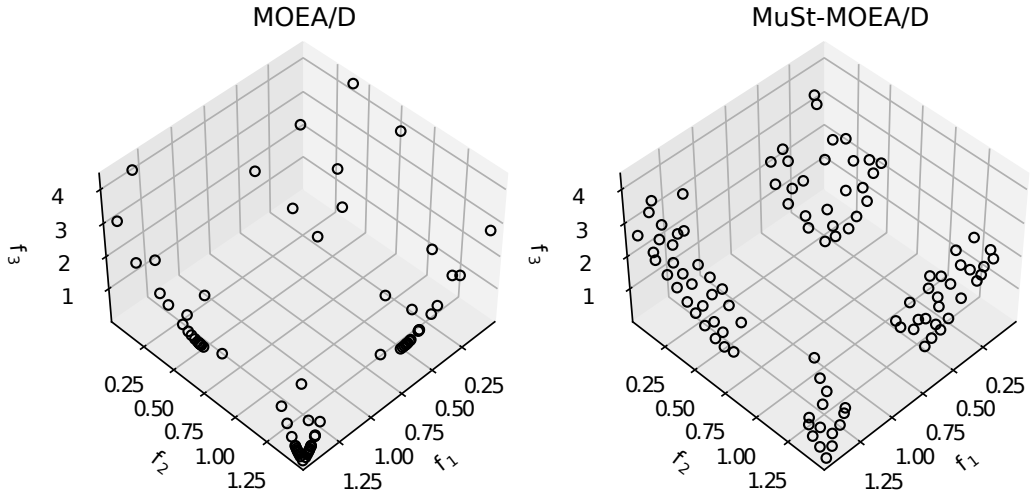

Figure A.39. MOEA/D and MuSt-MOEA/D results are presented for the MW14 problem. 50 trials are executed and the ND set for the median HV run is plotted.

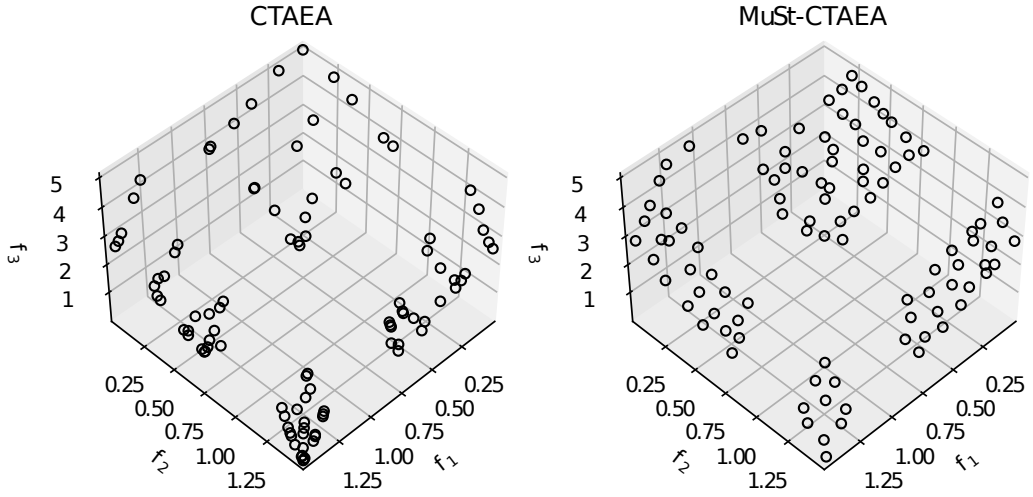

Figure A.40. C-TAEA and MuSt-C-TAEA results are presented for the MW14 problem. 50 trials are executed and the ND set for the median HV run is plotted.

A.4 Additional Performance Quality Indicators for the Crashworthiness Problem

Chapter 6 presented the hypervolume (HV) convergence curve on the crashworthiness problem, depicting the multi-stage HV behavior through different stages of the MuSt-NSGA-III algorithm. Here, we show similar plots but using other quality indicators. It is worth noting that the variation of the number of ND solutions is not shown in these plots, as it is identical to that in the HV plot in the chapter 6.

The k-neighbor distance mean is presented in Figure A.41. It is expected that a larger value of this metric will occur for fewer ND points, but it may settle down to a value when the number of ND points stabilizes. At the beginning, more and more ND points are being found and this metric values reduced with SEs. At the beginning of Stage 3, Iteration 1, we can see an abrupt decrease in the indicator, as more points are discovered with an increase in number of RVs. clearly, the end of Iteration 1, almost 100 ND points are already obtained and there is little effect at the second iteration of Stage 3. At the end, almost $N = 100$ points are found, and even with this number of solutions, the k-neighbor distance mean is still higher than that of the base NSGA-III algorithm, meaning that with MuSt-NSGA-III is able to find a better distribution than NSGA-III with 100 solutions in each set.

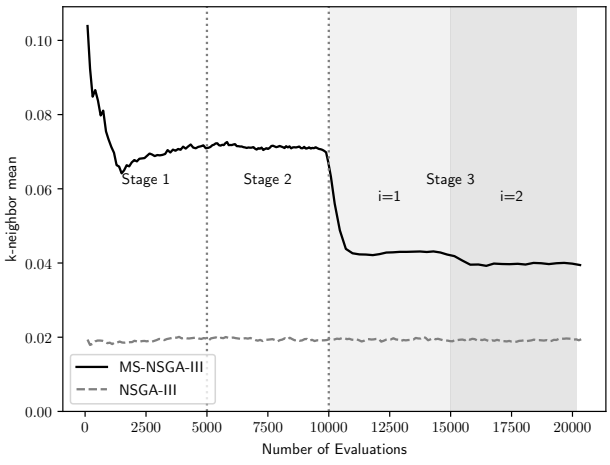

Figure A.41. K-neighbor Mean for crashworthiness problem. Convergence curve using the average from 50 runs, comparing NSGA-III and MuSt-NSGA-III.

The k-neighbor distance standard deviation is shown in Figure A.42. A small standard deviation indicates a more uniform distribution of points. With stages, the

distribution gets better and the final standard deviation of MuSt-NSGA-III is smaller (better) than that of NSGA-III.

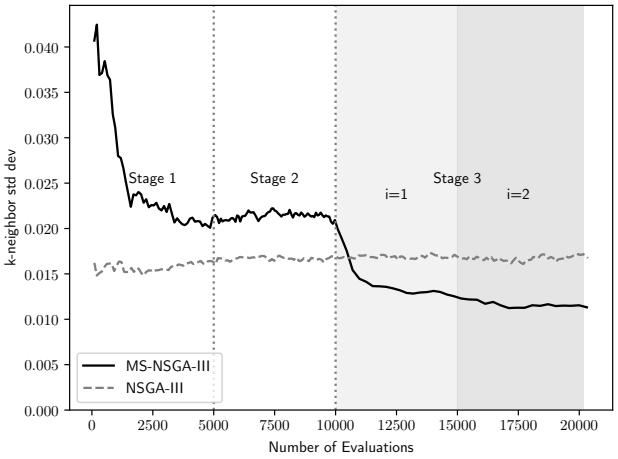

Figure A.42. K-neighbor standard deviation for crashworthiness problem. Convergence curve using the average from 50 runs, comparing NSGA-III and MuSt-NSGA-III.

A small spacing metric value is better. In Figure A.43 we can observe the spacing indicator which show a similar reducion trend compared to Figure A.42. In fact, we have observed similarities in these two metric values in other problems as well. These two quality indicators appears to be correlated. The final spacing value is smaller (better) than that of NSGA-III solutions.

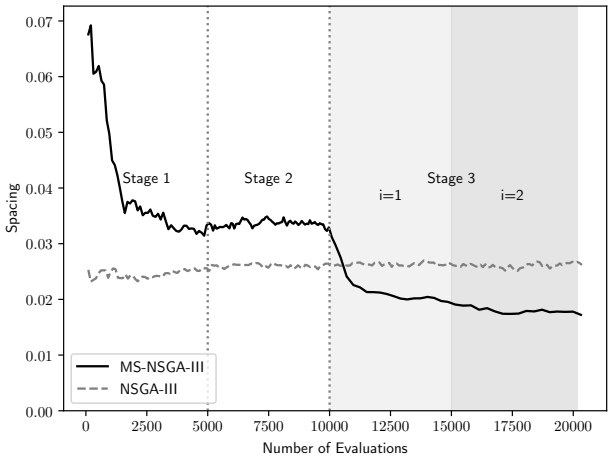

Figure A.43. Spacing performance quality indicator for crashworthiness problem. Convergence curve using the average from 50 runs, comparing NSGA-III and MuSt-NSGA-III.

The variation in the UD indicator is shown in Figure A.44. Higher this metric value, better is the distribution. An increase in ND points at the beginning of Stage 3 reduces the UD metric value. Importantly, the final value is higher (better) compared to NSGA-III solution set.

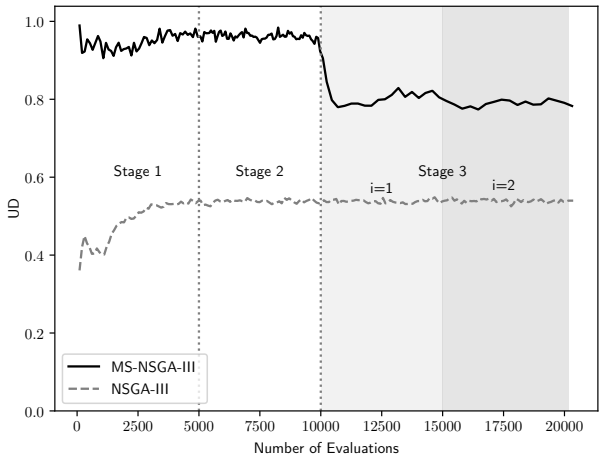

Figure A.44. UD performance quality indicator for crashworthiness problem. Convergence curve using the average from 50 runs, comparing NSGA-III and MuSt-NSGA-III.

Evenness is the final quality indicator presented here. Figure A.45 depicts its behavior. Smaller this metric value, better is the distribution. Having an identical number of solutions, MuSt-NSGA-III is able to produce an ND set having better evenness metric value.

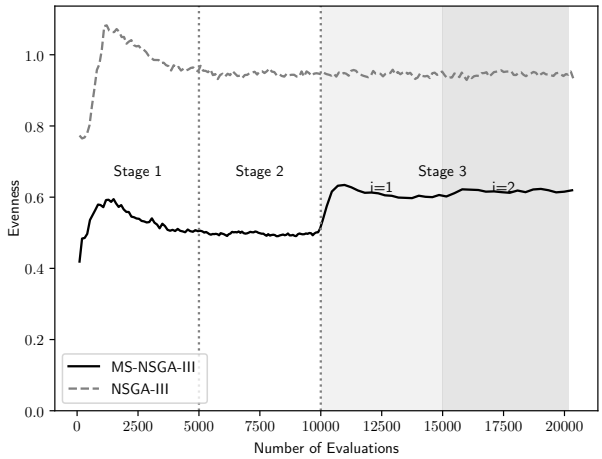

Figure A.45. Evenness performance quality indicator for crashworthiness problem. Convergence curve using the average from 50 runs, comparing NSGA-III and MuSt-NSGA-III.

Along with the HV variation, these performance metric plots clearly indicate that MuSt-NSGA-III is able to find an DN solution set which has a more uniform-like distribution compared to the set found by NSGA-III. Similar variations in performance metrics are also obtained for other problems of this study.

A.5 Additional Performance Quality Indicators for the MaF7 Problem

Additionaly to the crashworthiness problem we present here similar plots for MaF7 Problem set. In plot A.46 we present HV convergence curve with the number of solutions in three different information: MuSt-NSGA-III Extended NSGA-III, and final ND NSGA-III.

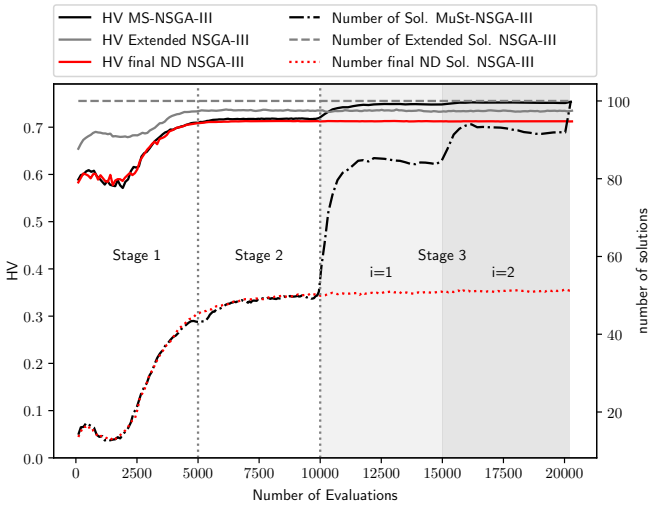

Figure A.46. Average hypervolume (solid lines) and the number of active RVs (dotted lines) obtained using NSGA-III calculated from two sets – closest for each ARV and all ND solutions – and MuSt-NSGA-III ND set for the MaF7 problem.

In Figure A.47 a view with a different scale is presented, showing the HV convergence information. It is possible to observe some increases in the quality indicator from Stage 2 to Stage 3 $i = 1$,and from Stage 3 $i = 1$ to Stage 3 $i = 2$.

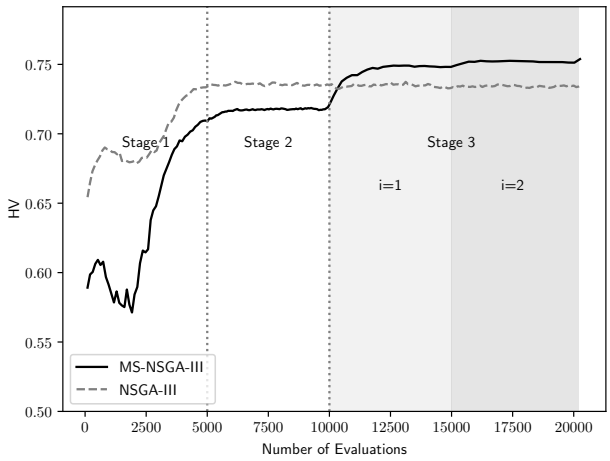

Figure A.47. Average hypervolume obtained using NSGA-III and MuSt-NSGA-III ND set for the MaF7 problem.

The k-neighbor distance mean is presented in Figure A.48. There is a decrease, considering MuSt-NSGA-IIIand the changes among the Stages. However, it is still superior when compared with NSGA-III.

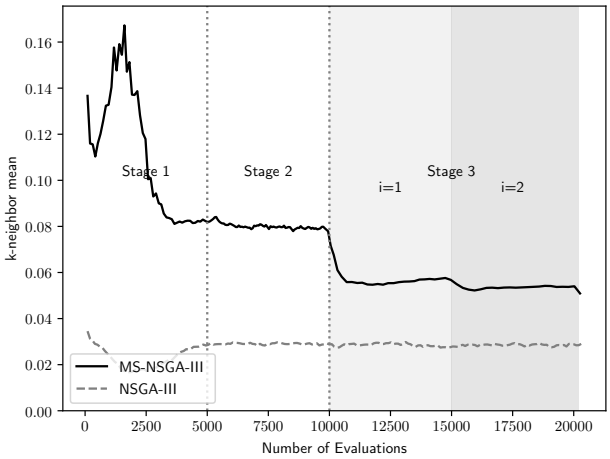

Figure A.48. K-neighbor Mean for MaF7 problem. Convergence curve using the average from 50 runs, comparing NSGA-III and MuSt-NSGA-III.

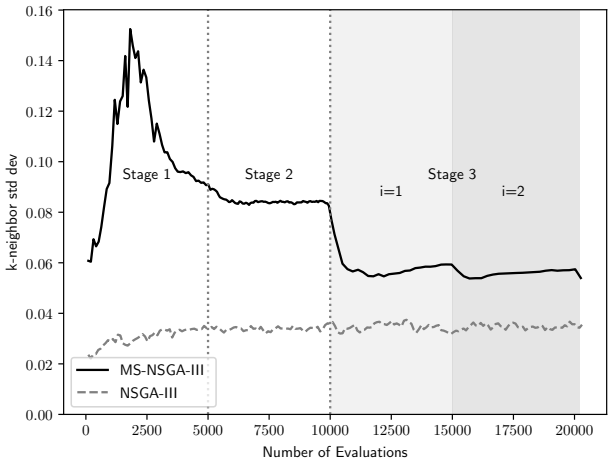

Figure A.49. K-neighbor standard deviation for MaF7 problem. Convergence curve using the average from 50 runs, comparing NSGA-III and MuSt-NSGA-III.

The k-neighbor distance standard deviation is shown in Figure A.49. In Figure A.50 the spacing indicator is also depicted. There is a similarity between the two indicators. The same fact happens to the crashworthiness test case.

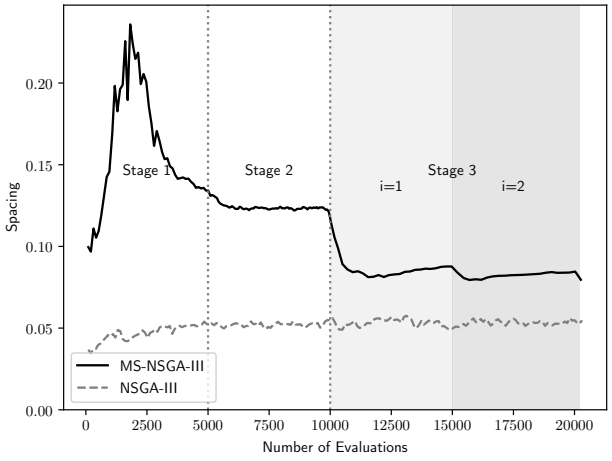

Figure A.50. Spacing performance quality indicator for MaF7 problem. Convergence curve using the average from 50 runs, comparing NSGA-III and MuSt-NSGA-III.

Finally , the Figures A.51 and A.52 presenting the UD and evenness quality indicator. In the evenness case, the value stays unchangeable even after the stage change.

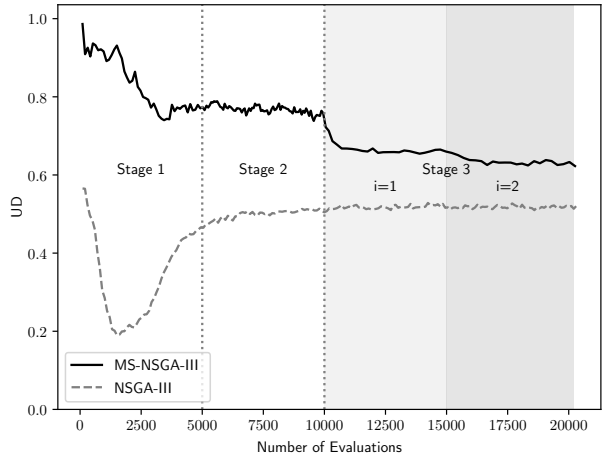

Figure A.51. UD performance quality indicator for MaF7 problem. Convergence curve using the average from 50 runs, comparing NSGA-III and MuSt-NSGA-III.

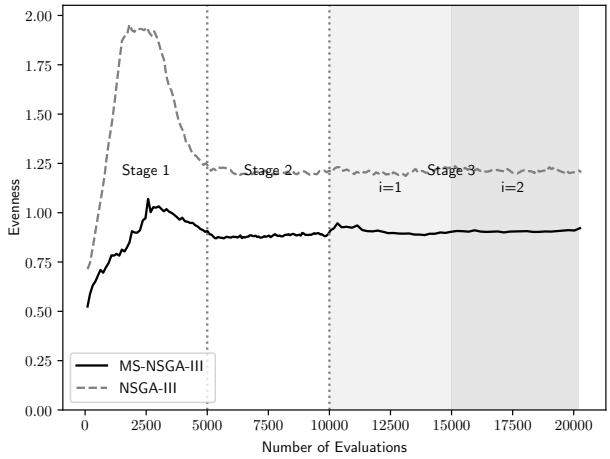

Figure A.52. Evenness performance quality indicator for MaF7 problem. Convergence curve using the average from 50 runs, comparing NSGA-III and MuSt-NSGA-III.

A.6 Additional Statistical Results for the Parametric Study

In chapter 6, we defined three config tests, using T_S as total number of evaluations functions and γ as a parameter, such that, the configuration tests could be defined related to γ. We used: $T_1 = T_2 = \frac{1-\gamma}{2}T$ and $T_3 = \gamma T$ and varied γ to have 1/3, 1/2, and 2/3. We collected the results for each config test and quality indicator and created

some basic statistics using the HV metric that helped us find a suitable value of γ. Here, we tabulate statistics for other six performance metrics. Tables A.14 to A.18 present the details of the Friedman test results on 11 problems.

Table A.14. Friedman ranks for k-neighbors distance mean for all config sets using $\gamma = 1/3$, $1/2$, and $2/3$.

Problem sets	**Config sets using γ**		
	1/3	1/2	2/3
	HV Friedman test rank		
ZDT3	0.0084 (1)	0.0083 (2)	0.0083 (3)
MaF01	0.0495 (3)	0.0498 (2)	0.0499 (1)
MaF07	0.0435 (3)	0.0439 (2)	0.04438 (1)
MaF10	0.0494 (1)	0.0421 (2)	0.03324 (3)
MaF11	0.0733 (3)	0.0762 (1)	0.07523 (2)
MaF12	0.0842 (3)	0.0855 (2)	0.08570 (1)
Crashworthiness	0.0705 (3)	0.07323 (2)	0.07340 (1)
Carside impac	0.0395 (3)	0.04031 (2)	0.04036 (1)
DAS-CMOP7	0.0794 (1)	0.07935 (2)	0.07853 (3)
DAS-CMOP8	0.0807 (2)	0.08142 (1)	0.07963 (3)
DAS-CMOP9	0.0185 (1)	0.01823 (2)	0.01794 (3)
Overall Ranking	*2.18*	***1.81***	*2.00*

Table A.15. Friedman ranks for k-neighbors distance std dev for all config sets using $\gamma = 1/3$, $1/2$, and $2/3$.

Problem sets	**Config sets using γ**		
	1/3	1/2	2/3
	HV Friedman test rank		
ZDT3	0.0029 (3)	0.0028 (2)	0.0027 (1)
MaF01	0.0133 (2)	0.0134 (3)	0.0126 (1)
MaF07	0.0073 (2)	0.0073 (3)	0.0071 (1)
MaF10	0.0138 (3)	0.0135 (2)	0.0132 (1)
MaF11	0.0141 (3)	0.0133 (1)	0.0138 (2)
MaF12	0.0150 (3)	0.0144 (2)	0.0141 (1)
Crashworthiness	0.0137 (3)	0.0117 (1)	0.0121 (2)
Carside impact	0.0121 (3)	0.0111 (1)	0.0114 (2)
DAS-CMOP7	0.0166 (3)	0.0158 (2)	0.0139 (1)
DAS-CMOP8	0.0170 (2)	0.0160 (1)	0.0187 (3)
DAS-CMOP9	0.0120 (3)	0.0112 (1)	0.0115 (2)
Overall Ranking	2.72	1.72	**1.54**

Table A.16. Friedman ranks for spacing indicator, for all config sets using $\gamma = 1/3$, $1/2$, and $2/3$.

Problem sets	**Config sets using γ**		
	1/3	1/2	2/3
	HV Friedman test rank		
ZDT3	0.0042 (3)	0.0040 (2)	0.0039 (1)
MaF01	0.0192 (3)	0.0191 (2)	0.0183 (1)
MaF07	0.0102 (2)	0.0102 (3)	0.0100 (1)
MaF10	0.0211 (3)	0.0210 (2)	0.0205 (1)
MaF11	0.0205 (3)	0.0192 (1)	0.0195 (2)
MaF12	0.0209(3)	0.0201 (2)	0.0194 (1)
Crashworthiness	0.0195 (3)	0.0163 (1)	0.0166 (2)
Carside impact	0.0181 (3)	0.0169 (1)	0.0171 (2)
DAS-CMOP7	0.0247 (3)	0.0237 (2)	0.0215 (1)
DAS-CMOP8	0.0252 (2)	0.0239 (1)	0.0282 (3)
DAS-CMOP9	0.0179 (3)	0.0173 (1)	0.0173 (2)
Overall Ranking	2.90	**1.54**	**1.54**

Table A.17. Friedman ranks for UD indicator, for all config sets using $\gamma = 1/3$, $1/2$, and $2/3$.

Problem sets	**Config sets using γ**		
	1/3	1/2	2/3
	HV Friedman test rank		
ZDT3	0.9769 (1)	0.9634 (2)	0.9591 (3)
MaF01	0.8640 (3)	0.8742 (2)	0.8944 (1)
MaF07	0.9850 (1)	0.9748 (2)	0.9723 (3)
MaF10	0.9351 (1)	0.8605 (2)	0.6717 (3)
MaF11	0.9709 (3)	0.9836 (2)	0.9893 (1)
MaF12	0.9797(3)	0.9868 (2)	0.9951 (1)
Crashworthiness	0.9330 (3)	0.9671 (1)	0.9655 (2)
Carside impact	0.7688 (3)	0.7991 (2)	0.7994 (1)
DAS-CMOP7	0.9975 (3)	1.0000 (2)	1.0000 (1)
DAS-CMOP8	0.9910 (3)	0.9975 (2)	1.0000 (1)
DAS-CMOP9	0.4872 (1)	0.4767 (2)	0.4645 (3)
Overall Ranking	*2.27*	***1.86***	***1.86***

Table A.18. Friedman ranks for Evenness indicator, for all config sets using $\gamma = 1/3$, $1/2$, and $2/3$.

Problem sets	Config sets using γ		
	1/3	1/2	2/3
	HV Friedman test rank		
ZDT3	1.6232 (1)	1.6304 (2)	1.6392 (3)
MaF01	0.7058 (3)	0.6996 (2)	0.6942 (1)
MaF07	0.2503 (3)	0.2392 (2)	0.2284 (1)
MaF10	0.7278 (1)	0.7847 (3)	0.7440 (2)
MaF11	0.3136 (3)	0.2949 (1)	0.3035 (2)
MaF12	0.2255(3)	0.2099 (1)	0.2117 (2)
Crashworthiness	0.2460 (3)	0.2117 (2)	0.2110 (1)
Carside impact	0.6364 (3)	0.6210 (2)	0.6242 (1)
DAS-CMOP7	0.2896 (3)	0.2881 (1)	0.2938 (2)
DAS-CMOP8	0.3069 (2)	0.2974 (1)	0.3305 (3)
DAS-CMOP9	0.8611 (3)	0.8535 (2)	0.8332 (1)
Overall Ranking	*2.45*	***1.64***	*1.90*

It is clear from the tables that while $\gamma = 2/3$ produces the best outcome, on most cases $\gamma = 1/2$ and $2/3$ produce better results than $\gamma = 1/3$.

A.7 Final comments

Along with the description and results presented in the main document, this supplementary document has provided a number of supporting figures, tables, and text in support of our proposed multi-stage procedure. The procedure has been well tested on a number of two to 10-objective problems on a number of EMaO algorithms. It is now ready to be integrated with other EMO/EMaO algorithms to make the baseline algorithm more reliable and to be applied to more problems.