Université Libre de Bruxelles
Faculté des Sciences Appliquées
CODE - Computers and Decision Engineering
IRIDIA - Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

A COMPONENT-WISE APPROACH TO MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

From Flexible Frameworks to Automatic Design

Leonardo César TEONÁCIO BEZERRA

Promoteur:

Dr. Thomas STÜTZLE

Co-promoteur:

Dr. Manuel LOPEZ-IBANEZ

Thèse présentée à la Faculté des Sciences Appliquées de l'Université Libre de Bruxelles
en vue de l'obtention du titre de Docteur en Sciences de l'Ingenieur.

Année académique 2015/2016

Abstract

Multi-objective optimization is a growing field of interest for both theoretical and applied research, mostly due to the higher accuracy with which multi-objective problems (MOPs) model real-world scenarios. While single-objective models simplify real-world problems, MOPs can contain several (and often conflicting) objective functions to be optimized at once. This increased accuracy, however, comes at the expense of a higher difficulty that MOPs pose for optimization algorithms in general, and so significant research effort has been dedicated to the development of approximate and heuristic algorithms. In particular, a number of proposals concerning the adaptation of *evolutionary algorithms* (EAs) for multi-objective problems can be found in the literature, evidencing the interest these algorithms have received from the research community.

This large number of proposals, however, does not mean that the full search power offered by multi-objective EAs (MOEAs) has been properly exploited. For instance, in an attempt to propose significantly novel algorithms, many authors propose a number of algorithmic components at once, but evaluate their proposed algorithms as monolithic blocks. As a result, each time a novel algorithm is proposed, several questions that should be addressed are left unanswered, such as (i) the effectiveness of individual components, (ii) the benefits and drawbacks of their interactions, and (iii) whether a better algorithm could be devised if some of the selected/proposed components were replaced by alternative options available in the literature. This component-wise view of MOEAs becomes even more important when tackling a new application, since one cannot antecipate how they will perform on the target scenario, neither predict how their components may interact. In order to avoid the expensive experimental campaigns that this analysis would require, many practitioners choose algorithms that in the end present suboptimal performance on the application they intend to solve, wasting much of the potential MOEAs have to offer.

In this thesis, we take several significant steps towards redefining the existing algorithmic engineering approach to MOEAs. The first step is the proposal of a flexible and representative algorithmic framework that assembles components originally used by many different MOEAs from the literature, providing a way of seeing algorithms as instantiations of a *unified template*. In addition, the components of this framework can be freely combined to devise novel algorithms, offering the possibility of tailoring MOEAs according to the given application. We empirically demonstrate the efficacy of this component-wise approach by designing effective MOEAs for different target applications, ranging from continuous to combinatorial optimization. In particular, we show that the MOEAs one can tailor from a collection of algorithmic components are able to outperform the algorithms from which those components were originally gathered. More importantly, the improved MOEAs we present have been designed without manual assistance by means of *automatic algorithm design*. This algorithm engineering approach considers algorithmic components of flexible frameworks as parameters of a tuning problem, and automatically selects the component combinations that lead to better performance on a given application. In fact, this thesis also represents significant advances in this research direction. Primarily, this is the first work in the

literature to investigate this approach for problems with any number of objectives, as well as the first to apply it to MOEAs. Secondarily, our efforts have led to a significant number of improvements in the automatic design methodology applied to multi-objective scenarios, as we have refined several aspects of this methodology to be able to produce better quality algorithms.

A second significant contribution of this thesis concerns understanding the effectiveness of MOEAs (and in particular of their components) for the application domains we consider. Concerning combinatorial optimization, we have conducted several investigations on the multi-objective permutational flowshop problem (MO-PFSP) with four variants differing as to the number and nature of their objectives. Through thorough experimental studies, we have shown that some components are only effective when used jointly. In addition, we have demonstrated that well-known algorithms could easily be improved by replacing some of their components by other existing proposals from the literature. Regarding continuous optimization, we have conducted a thorough and comprehensive performance assessment of MOEAs and their components, a concrete first step towards clearly defining the state-of-the-art for this field. In particular, this assessment also encompasses many-objective optimization problems (MaOPs), a sub-field within multi-objective optimization that has recently generate much interest within the MOEA community given its theoretical and practical demands. In fact, our analysis is instrumental to better understand the application of MOEAs to MaOPs, as we have discussed a number of important insights for this field. Among the most relevant, we highlight the empirical verification of performance metric correlations, and also the interactions between structural problem characteristics and the difficulty increase incurred by the high number of objectives.

The last significant contribution from this thesis concerns the previously mentioned automatically generated MOEAs. In an initial feasibility study, we have shown that MOEAs automatically generated from our framework are able to consistently outperform the original MOEAs from where their components were gathered both for the MO-PFSP and for MOPs/MaOPs. The major contribution from this subset, however, regards multi-objective continuous optimization, as we significantly advance the state-of-the-art for this field. To accomplish this goal, we have extended our framework to encompass approaches that are primarily used for this continuous problems, although the conceptual modeling we use is general enough to be applied to any domain. From this extended framework we have then automatically designed state-of-the-art MOEAs for a wide range of experimental scenarios. Moreover, we have conducted an in-depth analysis to explain their effectiveness, correlating the role of algorithmic components with experimental factors such as the stopping criterion or the performance metric adopted.

Finally, we highlight that the contributions of this thesis have been increasingly recognized by the scientific community. In particular, the contributions to the research of MOEAs applied to continuous optimization are remarkable given that this is the primary application domain for MOEAs, having been extensively studied for a couple decades now. As a result, chapters from this work have been accepted for publication in some of the best conferences and journals from our field.

Acknowledgements

The academic challenge of producing a thesis is significant, but it is very far from the life challenges I faced during this time. Before acknowleding anyone else, I want to acknowledge myself for having made it here. I'll use Zygmunt Bauman's words about living abroad to briefly describe what this experience was like for me:

> There is always something to explain, to apologize for, to hide or on the contrary to boldly display, to negotiate, to bid for and to bargain for; there are differences to be smoothed or glossed over, or to be on the contrary made more salient and legible. 'Identities' float in the air, some of one's own choice but others inflated and launched by those around, and one needs to be constantly on the alert to defend the first against the second; there is a heightened likelihood of misunderstanding, and the outcome of the negotiation forever hangs in the balance. The more one practises and masters the difficult skills needed to get by in such an admittedly ambivalent condition, the less sharp and hurting the rough edges feel, the less overwhelming the challenges and the less irksome the effects. One can even begin to feel everywhere *chez soi*, 'at home' – but the price to be paid is to accept that nowhere will one be fully and truly at home.

In my case, I lived both the "embrace other cultures" and the "fight for your own" experiences. The latter was pretty difficult, but I am glad that I can still feel at home now that I'm back.

Having said so, let's get back to the true acknowledegements. The number of people one must acknowledge in a thesis is exponential in the number of hours spent awake through the nights. In my case, that is definitely not P. But on this most auspicious of nights, like the fella once said, let me make the most of the, by far, best aspect of this life experience, and thank as many people as I can in the languages they shared with me.

> Thanks, Thomas and Manuel, for the research experience you shared with me. It has definitely been the most challenging course I have ever taken, but it makes me very proud to now say that this thesis is as much mine as yours. From now on, see you in Natal.
>
> Thanks, jury members, for sharing your time to contribute to this thesis. I am most glad that the best researchers from my field took part in my work, and I hope I can help you push the boundaries over the next decades. See you in Natal.
>
> Grazie mille, italiani di IRIDIA, per la lingua e per le ricette. Grazie anche agli italiani dal calcetto, però siete scarsissimi. Ci vediamo a Natal.
>
> Gracias Manuel, Leslie, Francisco, Silvia, Helena, Javier, Manu. Seguro que nos encontraremos otras veces en España o en Chile pero, por ahora, nos vemos en Natal.
>
> Danke Thomas, Holger, Nithin, and Arne. Die These ist auf Lager. That's pretty much all the German I know and clearly Amazon taught me wrong.

O desafio acadêmico de produzir uma tese é grande, mas está longe de se comparar aos desafios de vida que eu enfrentei durante esse tempo. Antes de agradecer a qualquer pessoa, eu quero agradecer a mim mesmo por ter chegado aqui. Faço uso das palavras de Zygmunt Bauman pra tentar dar uma noção do que foi esta experiência para mim:

> Estar total ou parcialmente "deslocado" em toda parte, não estar totalmente em lugar algum (ou seja, sem restrições e embargos, sem que alguns aspectos da pessoa "se sobressaiam" e sejam vistos por outros como estranhos), pode ser uma experiência desconfortável, por vezes perturbadora. Sempre há alguma coisa a explicar, desculpar, ou, pelo contrário, ressaltadas e tornadas mais claras. As "identidades" flutuam no ar, algumas de nossa própria escolha, mas outras infladas e lançadas pelas pessoas em nossa volta, e é preciso estar em alerta constante para defender as primeiras em relação às últimas. Há uma ampla possibilidade de desentendimento, e o resultado da negociação permanece eternamente pendente. Quanto mais praticamos e dominamos as difíceis habilidades necessárias para enfrentar essa condição reconhecidamente ambivalente, menos agudas e dolorosas as arestas ásperas parecem, menos grandiosos os desafios e menos irritantes os efeitos. Pode-se até começar a sentir-se *chez soi*, "em casa", em qualquer lugar – mas o preço a ser pago é a aceitação de que em lugar algum se vai estar total e plenamente em casa.

No meu caso, vivenciei tanto o lado de "abraçar novas culturas" como o de "lutar pela minha". Este último foi bem difícil, mas eu me sinto feliz de poder me sentir em casa agora que voltei.

Dito isto, volto aos reais agradecimentos.

> Obrigado, mãe, por ter investido em minha educação ao custo de sua própria qualidade de vida. Seus investimentos naquela época já foram suficientes pro resto da vida, então use da mesma dedicação em formar para agora viver e ser feliz.
>
> Obrigado, amor, pelo companheirismo diário. Dizem que um doutorado é um casamento complicado entre orientandos, orientadores e a tese. Conciliar dois casamentos, então, é um desafio enorme e eu espero ter me saído bem nessa. Com certeza, sem você nem o mestrado nem o doutorado teriam dado certo, então esta tese é tão sua quanto minha.
>
> Obrigado, Aninha e Naide, pelo humor nosso de cada dia. Neste quesito também entram Allysson, Tuiza e até Bibi, porque sem vocês a vida teria sido muito mais sem graça. Obrigado, na verdade, a toda minha família de Carapebas e de Nova Iorque de Angicos, que infelizmente eu não vou poder citar nominalmente porque as páginas não são amarelas.
>
> Obrigado, sardinhas, pelo aprendizado de vida do dia a dia. Obrigado principalmente pelos churrascos e feijoadas, pelas arraias, camarões e lagostas, pelos caçuletes e escablefes, ... Enfim, pela azia e pelos desafios pra me manter em forma.
>
> Obrigado, Beth, Marco, Luciana, Wagner e meus colegas de LAE, pelo incentivo e investimento durante os tempos de mestrado que hoje renderam como fruto esta tese. Como diria um amigo de faculdade, o delta foi bastante grande entre o meus primeiros dias na pesquisa e a formação com a qual eu saí. Sem esta etapa, eu ainda teria alguns longos anos pra desenvolver um trabalho deste nível. Aliás, falando em amigos de turma, obrigado ao pessoal da faculdade e de todas as escolas por onde eu passei.
>
> Obrigado, brasileiros que me ajudaram a me sentir em casa estando tão longe durante esses cinzas e depressivos dias belgas. A lista é grande, mas obrigado Sabrina, Teresa, Carla, Luh, Edi, Zé de Iran, Daniel, Miná, Chicó, Isa, (...).
>
> Por último, mas com certeza alguns dos mais importantes, obrigado aos meus irmãos que eu tenho o prazer e a honra de chamar de amigos. Muitos investiram em mim do ponto de vista da formação acadêmica, mas foi com vocês que eu formei meu caráter.

The research leading to the results presented in this report has received funding from the COMEX project (P7/36) within the Interuniversity Attraction Poles Programme of the Belgian Science Policy Office. I acknowledge the support from the Belgian F.R.S.-FNRS, of which I have been a FRIA fellow.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

Optimization problems arise in several real-world scenarios, particularly in the business and industry areas. In industry, one example is the scheduling of intermediary steps of the production lines in a workshop, which is a critical factor for the profit of an enterprise. This type of problem has been modeled as shop scheduling problems in operations research [83]. In particular, when a processing order must be respected for all jobs, this problem is known as the *permutation flowshop problem* (PFSP) [87], one of the most studied *combinatorial optimization* problems in the literature. In business, predictive models are used to extrapolate the values of a given variable given a sample distribution. Selecting the best model, however, is a complex *continuous optimization* task that involves minimizing the error ratio for the prediction of the known data [142]. In fact, this problem has become ever more important with the advent of big data and the ongoing fourth industrial revolution, stirred on machine learning techniques and deep learning [47].

Whether combinatorial or continuous, solving an optimization problem may prove to be a challenging task. In theoretical computer science, this is captured by the definition of the class of NP-hard problems, i.e., problems for which the currently most efficient algorithms require (at least) exponential running time w.r.t. the number of variables to be optimized [86]. Whether it is possible to develop polynomial running time algorithms for this class of problems remains the most important open question in computer science, even after decades of research in this field. To be able to address many of these problems, researchers have used theoretical models that simplify the challenges of the true real-world problems. For instance, the bulk of the research on optimization algorithms has concentrated on theoretical problems that are (i) *static*, meaning the problem information is fully available before the algorithm is run; (ii) *deterministic*, meaning that the information is known with certainty, and; (iii) *single-objective*, meaning that a single cost function is to be optimized.

The practical disadvantages of such models are clear. Concerning the number of objectives considered, for instance, the traditional modeling had to either (i) rank cost functions and optimize only the most important, or (ii) use an aggregated function representing some compromise between the different criteria. Although neither of these approaches were ideal, they were required during the early years of optimization research for the feasibility of those investigations. Over the past decades, however, the optimization community has seen an ever growing movement towards adopting models that are more accurate w.r.t. real-world problems [57, 105]. In particular, the field of multi-objective optimization has been the subject

of an increasing number of investigations for at least three decades now [48, 50, 57, 149, 176, 199]. As one might deduce, the already significant challenge posed by some single-objective optimization problems makes the development of algorithms for multi-objective problems an even more difficult task. In fact, even single-objective problems for which polynomial runtime algorithms have been developed may become NP-hard with the addition of a second objective, such as shortest path problems [209].

The largest challenge posed by an extra objective concerns solution optimality. In multi-objective problems it is rarely the case that a single solution will be considered optimal for all objectives at once. This way, the traditional goal of multi-objective optimizers is to retrieve the optimal set of solutions according to a given preference relation, an order relation that allows the algorithm to compare solutions. The goal of an algorithm is then to retrieve the solution subset that is minimal (or maximal) according to the given relation. The most adopted preference relation, *Pareto dominance* [84], states that a solution s_1 is only considered better than another solution s_2 if s_1 is not worse than s_2 for all objectives, and it is better for at least one. Clearly, Pareto dominance might lead to a significant number of optimal solutions in a given problem – the more, the larger the number of objectives considered.

A significant share of the additional difficulties posed by multi-objective problems have been, however, successfully dealt with by metaheuristic algorithms. In particular, metaheuristics are problem-independent heuristics that can be applied to any domain given a few adaptations [91]. Originally, this kind of algorithm was devised with single-objective problems in mind, but it was quickly shown to be particularly suited to MOPs. A subset of metaheuristics are population-based, i.e., they maintain a set of candidate solutions being simultaneously optimized. In addition, information learned from a given solution is used to improve others, making this kind of algorithm actually collaborative learning approaches. Clearly, population-based algorithms are intrinsically applicable to MOPs, as they can be adapted to maintain and simultaneously optimize a number of trade-off solutions within a single run of the algorithm. As a result, an extensive research effort has been conducted over the past couple of decades concerning proposals of population-based metaheuristics to MOPs [50, 85, 157, 176, 199].

The best-established metaheuristic community within MOP research is the one focusing on evolutionary algorithms (EAs). In fact, the first proposals of multi-objective metaheuristics concern EAs [204], although nearly two decades were required for algorithms to start maturing [56]. Concretely, although the population-based nature of multi-objective EAs (MOEAs) makes them natural candidates for optimizing MOPs, preserving the learned information (convergence) and spread (diversity) throughout the run of the algorithms is a critical ingredient that has been addressed ever more often. Initially, MOEAs were reported to suffer from speciation (converging to a single trade-off region) or lacking convergence at all [233]. Over time, key concepts such as elitism [60, 137, 233, 234], external archives [137, 160], and indicator-driven search [19, 232] were proposed by MOEA designers, and their effectiveness experienced a major improvement. Later, however, the increase in the number of objectives was shown to significantly undermine the good performance of MOEAs [132, 195], initiating the current race for improving the understanding of MOEAs and their effectiveness on *many-objective problems* (MaOPs), i.e., problems with more than three objectives [3, 46, 79, 149].

A wide range of factors could likely explain the difficulties MOEAs still face when dealing with MOPs and particularly MaOPs. For instance, in multi-objective continuous optimization some problem features are known to pose additional difficulties for EAs, such as the presence of Pareto local optimum fronts, multi-modality, or search space density bias [61, 112]. In MaOPs, this is made more challenging by the large proportion of the objective space that becomes incomparable, a phenomenon known as *dominance resistance* [195]. Concerning combinatorial problems, MOEAs are highly dependent on the quality of the variation operators designed for the application domain, responsible for producing novel solutions that should represent improvements over the original solutions. Since this can be a quite difficult task to accomplish, it is not surprising that MOEAs benefit from additional convergence pressure provided, for instance, by problem-specific local search heuristics [110, 187, 198].

Another factor that influences directly the performance of MOEAs on many-objective problems con-

cerns the metrics used for their evaluation. In more detail, the evaluation of MOEAs has been traditionally performed by assessing the image of the solution set they return. Formally, this image is called an *approximation front*, and the goal of a multi-objective optimizer is to retrieve a front that best approximates the image of the true subset of optimal solutions according to the preference relation used. In the case of Pareto dominance, that image is called the *Pareto-optimal front*. However, comparing fronts is an ever increasingly complex task as the number of objectives considered grows. Several analytical metrics have then been proposed to simplify this task, in an attempt to make it independent of the number of objectives presented by the target application. As one might infer, measuring the quality of an approximation front of a multi-objective problem by means of real-valued, one-dimension metrics means losing a considerable amount of information. More precisely, a performance metric is only able to grasp a few of the desirable characteristics an approximation front should present, and hence it is advisable to use a set of metrics to make assessments more complete. However, it has been shown that some of these metrics present considerable inconsistencies, in some cases ever growing with the increase in the number of objectives [129]. This way, MOEAs that were originally designed with a given metric in mind are now being evaluated according to a more diverse set, and it becomes more and more clear that the fronts produced by these algorithms could be improved in ways that had not previously been considered.

Notwithstanding the traditional motives used to explain the need for improving MOEAs, we believe that a significant portion of the responsibility should be laid upon MOEA researchers themselves or, more precisely, on the traditionally adopted algorithmic engineering approach. On one hand, the MOEA community is accredited with the major advancements on multi-objective performance assessment theory and tools [61, 97, 140, 235], and with having proposed highly effective algorithmic ideas that have later been reused by researchers from other metaheuristics. On the other hand, the theoretical work on the performance assessment of MOEAs has never been properly matched by a rigorous evaluation of newly proposed algorithms, or even by experimental campaigns benchmarking algorithms. In fact, the only significant analyses of the latter type date now from over a decade ago [61, 132, 233], and in those studies most of the now well-established performance metrics were not used. As a result, most of the MOEAs proposed over the past ten years have only been compared to a few selected algorithms believed to be high-performing, such as NSGA-II [60] or SPEA2 [234], which have been chosen due to their popularity and historical relevance.

An additional factor that undermines the general conclusions taken from existing experimental analyses is the lack of proper *numerical parameter tuning*. Specifically, metaheuristic algorithms are stochastic methods that present numerical parameters used to regulate the balance between intensification and diversification. More precisely, depending on the characteristics of the target problem, pushing the search for convergence (intensification) may cause the algorithm to get stuck either on plateaus, i.e., regions where many solutions present the same quality, or in the basis of attraction of local optima, i.e., solutions that are only optimal w.r.t. a subset of the feasible solution set [110]. Algorithm configuration hence constitutes a meta-optimization problem, where one must optimize the numerical parameter values used by a given metaheuristic algorithm to ensure its maximum performance when tackling a target problem. As a manual task, configuring algorithms can be both time-consuming and of little effectiveness, as the most common approaches either require a large computational effort (for running and comparing different configurations) or deep statistical and/or problem knowledge (to narrow down the number of configurations to be tested). More recently, however, the proposal of several *automatic algorithm configurators* has made this task far easier and more effective [16, 36, 37, 113]. Concretely, such configurators are in essence heuristic optimizers, presenting some parameters of their own and thus establishing a cyclic issue. Nonetheless, they are able to address a number of peculiarities of the algorithm configuration problem, namely the mixed nature of the variables to be optimized and the conditional dependencies between them. As a result, automatic configurators have been ever more often used by researchers [107], and are now even an integral part of some comercial optimizers such as the ILOG-IBM-CPLEX [118]. So

far, however, no MOEA experimental analysis so far has used automatic configurators, and even novel algorithms being proposed are still configured using manual approaches.

To further illustrate the issues with the traditional algorithm engineering methodology also adopted by the MOEA community, consider a practitioner dealing with a novel application domain, or simply little familiar with the literature on MOEAs. A quick literature review will reveal dozens of proposals over the years, but no clearly defined outperforming algorithm and a yet more restricted set of analyses of MOEAs on different application domains. A dedicated practitioner might still select a few popular or recent algorithms and run some exploratory analysis on his/her target domain, but reach the wrong conclusion about which MOEA suits his application best simply by not knowing which algorithms to use in his analysis, or by using a single performance metric that is unable to evaluate MOEAs in a holistic way. More importantly, by adopting parameter configurations originally suggested for an experimental setup different from the one under consideration now, the practitioner may reach the wrong conclusions even if he selects the appropriate subset of MOEAs to analyze.

From a research perspective, the way MOEAs have been proposed over the years poses a yet more significant drawback, although this is not an issue exclusive to MOEAs. Specifically, MOEAs are designed as collections of algorithmic components but evaluated as monolithic blocks. It is rarely the case that individual components are assessed for their contribution to effectiveness or interactions with other components. This lack of more general, unified model proposals has major consequences on the overall learning speed of the community. As a result, there is a clear disconnection between some recent proposed approaches and the algorithms they were meant to be built upon. Two important examples concern (i) algorithms that are unable to clearly improve over their predecessors [13, 59], or; (ii) identical algorithms that are independently proposed [19, 216]. Even more regrettable is the fact that MOEAs have been seen as belonging to different design paradigms, and the research on combining these search paradigms is yet incipient. The most relevant example of the potential benefits of these combinations regards *dominance*- and *indicator*-based MOEAs. When first proposed, using quality indicators within a MOEA to guide its search appeared an infeasible task due to the high computational complexity many such indicators present. Later, however, researchers proposed using indicators as a refinement technique to improve dominance relations, while reducing the computational overhead required to compute the indicators. More recently, similar proposals have attempted to combine dominance- and *decomposition*-based algorithmic components [59]. Altogether, these efforts help demonstrate the importance of considering MOEAs as a collection of algorithmic components rather than monolithic blocks.

One of the major drawbacks with a component-wise algorithm engineering approach is practicality. Revisiting the example from a practitioner searching for a MOEA effective for his target application, his investigation now becomes an issue of which components to select rather than which MOEA to use. Even with some MOEAs presenting common components, it is clear that the design space for the practitioner is now substantially expanded. However, it is possible to address this issue using one of the most promising applications of automatic algorithm configurators. Concretely, recent works on *automatic algorithm design* are based on using configurators to search an augmented configuration space, including both algorithmic components to be selected and parameters to be configured. These works have demonstrated the potential of configurators to effectively search these *design spaces* for several artificial intelligence fields [70, 133, 154, 157, 208, 217, 226]. In particular, works on template- and grammar-based automatic design of single- and multi-objective metaheuristics have shown not only the feasibility of this approach, but also its effectiveness [70, 133, 157]. In more detail, such works initially analyze the literature on metaheuristics proposed for a given field, and propose a structured, unified model (a template or a grammar) that can be used to flexibly combine existing algorithmic components in human-reasonable ways. Besides automating design, unified models are also instrumental for analyzing the effectiveness of selected designs, a task that becomes straightforward as adding or removing components is made trivial.

1.1 Objectives and Contributions

The main objective of this thesis is **to propose a novel, automated, and more effective way to (i) design and (ii) analyze multi- and many-objective optimization algorithms**. Specifically, the goal of this thesis is to investigate the effectiveness and applicability of the recent works on automatic algorithm engineering for the field of MOEAs, using template-based algorithmic frameworks. To achieve this goal, we (i) review the literature on MOEAs and propose a unified model that can be used as the basis for a flexible and representative template-based algorithmic framework; (ii) demonstrate that effective state-of-the-art algorithms can be designed in an automated way using this framework, and; (iii) show that high-performing algorithms instantiated with this framework can be analyzed in an automated way, producing relevant insights. To ensure the representativeness of our study, we consider two of the most relevant domains where MOEAs have been applied, namely continuous and combinatorial optimization problems.

The main contributions of this thesis are listed below:

1. **A component-wise MOEA algorithmic framework based on a unified template:** we propose and later refine a component-wise MOEA algorithm framework based on a unified template from which a practitioner can deal with MOEAs as freely combinable algorithmic components. The proposed framework is based on the most representative MOEA unified model proposed to date, evidenced by two important abstractions we adopt. The first concerns general preference relations, a concept that we adapt from its original definition to make it practical for our purposes. In particular, our modeling of preference relations allows algorithmic components from different MOEAs to be freely integrated in human-reasonable ways. More importantly, our proposal allows designers to combine, in a single algorithm, different search paradigms that so far had never been jointly considered, such as the combination of indicator- and decomposition-based algorithmic components. Finally, we allow designers to use different preference relations for each of the main components of MOEAs, namely mating and enviromental selection, as well as archive truncation.

   The second abstraction level concerns the conceptual separation between algorithmic components responsible for dealing with multi-objective aspects of the search such as dominance, convergence, and diversity (MO components), and the underlying EA with which they are coupled. In more detail, evolutionary computation comprises a wide family of algorithms, some of which are jointly called evolutionary algorithms. In multi-objective optimization, MOEAs do not always use the same underlying EA, a fact that hinders directly the fairness of comparisons between them. Even more serious is the fact that researchers have concentrated much of their efforts solely on devising MO components, without paying the due attention to the interactions between these components and the different underlying EAs one could use. As a result, the literature investigating the effectiveness of underlying EAs or variation operators for multi-objective optimization is rather small. The abstraction we adopt in our framework is instrumental in this direction, since effective underlying EAs from single-objective domains can now be freely combined with the most relevant MO components from the MOEA literature.

2. **The empirical demonstration that automatically component-wise designed MOEAs can outperform manually designed MOEAs:** In an initial feasibility study, we automatically design several MOEAs (AutoMOEA's) that outperform the traditional MOEAs used for assembling our framework. Concretely, we define a design space based on our template, and use an automatic algorithm configurator to search this design space. We demonstrate the feasibility of the proposed approach on four variants of the MO-PFSP, differing as to number and nature of their objectives, as well as for two benchmark sets of continuous optimization problems with two, three, and five objectives. In all scenarios considered, the AutoMOEA algorithms are both more effective and more robust than the MOEAs from which we gathered the components used to assemble our

framework were gathered. These results are further strengthened by the additional factors we consider in each investigation. In the case of the MO-PFSP, we address the possibility of designing an AutoMOEA general enough for any variant of this problem. Results demonstrate that this is a feasible line of research, with the more general AutoMOEA outperforming existing MOEAs. However, the performance of this algorithm is more strongly affected by the differences from tuning and test instance sets, and so it is ultimately matched or surpassed by variant-specific AutoMOEA's. Regarding continuous optimization, we consider different stopping criterion and cross-benchmark effects to respectively demonstrate the flexibility and robustness of the proposed approach. In the former setup, for instance, we see how the performance of some of the best-performing existing MOEAs is greatly affected, and also how the design of the AutoMOEA algorithms adapt to the different setup. More importantly, this is compelling evidence that an automatic, component-wise approach to MOEA design is a promising research direction.

3. **A comprehensive performance analysis of multi- and many-objective EAs, a concrete first step towards defining a state-of-the-art in MOEAs for continuous optimization:** When attempting to design state-of-the-art AutoMOEAs for continuous optimization, it became clear that this primary application field for MOEAs lacked a recent, comprehensive, and rigorous performance assessment that could identify the state-of-the-art MOEAs for both multi- and many-objective optimization. The analysis we conduct is the broadest currently available in this field, and is further enriched by the investigation of the several experimental factors we consider. In particular, we assess MOEAs in a wide set of experimental scenarios varying in the number of objectives and the stopping criterion used, and take additional precautions, such as using a set of diverse performance metrics, a reasonable number of problem sizes, and properly tuning numerical parameters presented by the algorithms for each experimental scenario. Even more importantly, in this work we adopt the conceptual separation between high-level multi-objective components (MO) and the underlying EA with which they are coupled, increasing both the fairness and the extent of our assessment. In total, our analysis encompasses 15 independently proposed MOEAs, 12 experimental scenarios, and a large set of hypothesis that we investigate. In fact, the overwhelming experimental data produced in this work is too large for a single research group to mine and is made available for the MOEA community to facilitate further investigations.

4. **The automatic design of state-of-the-art multi- and many-objective EAs for continuous optimization:** Besides demonstrating the feasibility of the automatic component-wise MOEA design approach proposed in this work, we demonstrate that it is also possible to automatically design MOEAs with state-of-the-art performance for a given application domain. In particular, we select as target application the primary domain for which MOEAs are proposed and on which they have been traditionally evaluated, namely continuous optimization. We then enrich our framework with a set of algorithmic components that have been proposed specifically for this domain, such as differential evolution [1, 2, 143, 166, 201, 219], or for the context of many-objective optimization, such as some decomposition-based components [59, 124]. Furthermore, the flexibility of our template allows all these components to be freely combined with the components that are domain-independent from our framework. Using a wide set of experimental setups, we design and evaluate a set of AutoMOEA+ algorithms, MOEAs that are able to consistently outperform the best-performing MOEAs identified in the literature for all scenarios considered. In particular, we propose a refinement of the automatic multi-objective algorithm design methodology that is instrumental for this and is one of the most significant contributions of our work. Specifically, we propose a multi-objective formulation of the tuning problem that is able to alleviate the contradictions between performance metrics, and allow us to automatically design MOEAs that are considered effective by all metrics we adopt for their evaluation.

5. **A set of automated experimental analyses concerning the effectiveness and interactions of individual MOEA components on a set of relevant benchmark problems:** One of the most direct benefits from the component-wise approach to MOEAs is the possibility to deconstruct an algorithm into its components and investigate their effectiveness and interactions. In particular, our component-wise approach is specially suited for an automated iterative analysis approach named *ablation analysis* [77] that we adapt for the context of automatic algorithm design. In this kind of analysis, one navigates a design space comprising the components of given source and target algorithms. At each step of the procedure, atomic algorithmic component changes are tested, and an ablation path is formed by iteratively selecting the atomic change that most radically alters the performance of the current intermediate algorithm. This way, it is possible to trace the performance evolution of atomic individual changes, understanding interactions between components and, in some cases, even to discover an intermediate configuration that outperforms both the source and target algorithmic designs.

   Clearly, our component-wise framework is intrinsically suited for this kind of analysis. We conduct two sets of investigations of this kind concerning relevant application domains for MOEAs, respectively the multi-objective permutation flowshop problem (MO-PFSP) and multi-objective continuous optimization problems. The first investigation focuses on the experimental setup factors that concern ablation, and specially the flexibility level of the abstractions adopted when ablating. We demonstrate, for instance, that the ability to use customized preference relations for each of the main components of a MOEA is instrumental to their effectiveness. In addition, we show that it is possible to find designs combining components from different MOEAs that outperform these algorithms. The second investigation focuses on the interactions between algorithmic components, experimental factors, and performance metrics. Among the most important insights drawn, we demonstrate how a design choice can lead to improvements according to a given metric but worsen the performance of the algorithm according to another, competing metric. In addition, we observe interactions between sets of algorithm components, reinforcing our claim about the importance of understanding component interactions when proposing a novel MOEA.

The remainder of this introduction provides a structured outline of this thesis. In particular, we delimit and further detail its structure and the publications related to this research.

1.2 Organization

This thesis is organized in six chapters, structured as follows. The most relevant background concepts required for understanding the contents of the thesis are reviewed in Chapter 2. In particular, we review the core concepts from multi-objective optimization, detail the most important characteristics of the application problems we consider in this thesis, and review the theoretical basis used for the performance assessment of multi-objective algorithms. A second topic covered by this chapter concerns evolutionary algorithms and their adaptation to multi- and many-objective optimization. We pay special attention to details that are critical for a proper understanding of the algorithms that are later deconstructed to provide the algorithmic components that we use in the framework proposed in the core chapters of the thesis. Finally, we discuss the most important topics concerning automatic algorithm design, in particular the nuances of the tuning problem, the different existing algorithm configurators, and a set of related topics that are closely related to automatic design. We remark that a reader familiar with all of those topics may proceed directly to the core chapters.

Chapter 3 presents an initial feasibility study where we empirically demonstrate the effectiveness of our component-wise approach coupled with the automatic algorithm design methodology. In particular, we first present the unified template we propose from which a set of relevant, existing MOEAs can be

instantiated, along with many other combinations that represent alternative MOEA designs. We then automatically generate a series of MOEAs (AutoMOEAs) for continuous and combinatorial optimization problems, demonstrating that these AutoMOEAs outperform the original MOEAs on nearly all experimental scenarios we consider, and are robust to stopping criterion changes and cross-validation experiments.

Our focused work on the field of continuous optimization is addressed by Chapters 4 and 5. In Chapter 4, we describe the comprehensive performance assessment of multi- and many-objective algorithms as a concrete first step to help clearly identify the state of the art in MOEAs for this domain. More importantly, this chapter presents the discussion of the most relevant insights we observe from this analysis, which comprise a major contribution to the design of better performing MOEAs.

The automatic design of state-of-the-art MOEAs for multi- and many-objective optimization is then the topic of Chapter 5. After an initial discussion of how we augment our template to encompass techniques specific to many-objective optimization and to continuous optimization, we demonstrate the effectiveness of the novel set of AutoMOEAs, which consistently outperform existing MOEAs. More importantly, we also discuss our efforts to refine the automatic algorithm design methodology, specially its ability to design algorithms that excel according to a set of diverse (and sometimes contradicting) performance metrics. Finally, we conduct a detailed follow-up investigation to assess the importance of specific components. More precisely, we deconstruct the AutoMOEAs through ablation analysis to correlate algorithmic components, structural problem characteristics, and performance metrics.

Finally, an extensive discussion about the conclusions drawn from this thesis are the goal of Chapter 6. More importantly, this research work opens a number of interesting avenues for future work that we discuss in detail, aimed at stirring the MOEA community towards such topics.

1.3 Publications

The research reported in this thesis has been the subject of collective work between the author and his supervisors. In particular, the core chapters of this work are based on conference and journal papers that have been published during the course of this research. A few of these works are still under revision or are yet to be submitted, as follows:

- The feasibility study described in Chapter 3 is based on the papers listed below. In particular, the initial studies on the MO-PFSP were published on selected conferences (1–2). A significantly extended version focusing on continuous optimization and considering a diverse set of tuning scenarios was later published in a high-impact journal (3).

1. Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. **Automatic design of evolutionary algorithms for multi-objective combinatorial optimization.** In Thomas Bartz-Beielstein et al, editors, *Parallel Problem Solving from Nature* (PPSN XIII), volume 8672 of LNCS, pages 508-517. Springer International Publishing, 2014
2. Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. **Deconstructing multi-objective evolutionary algorithms: an iterative analysis on the permutation flowshop problem.** In Panos M. Pardalos et al., editors, *Learning and Intelligent Optimization* (LION 2014), volume 8426 of LNCS, pages 157-172. Springer International Publishing, 2014.
3. Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. **Automatic component-wise design of multi-objective evolutionary algorithms.** *IEEE Transactions on Evolutionary Computation*, 2016. In press.

- The comprehensive performance assessment targeting multi- and many-objective evolutionary algorithms described in Chapter 4 is based on the papers listed below. In particular, the initial investigations published at a specialized conference concern: (4) the interactions between high-level multi-objective components and underlying evolutionary algorithms, and; (5) a preliminary comparison of selected MOEAs from different design paradigms on a restricted experimental scenario. The comprehensive version of this investigation will be submitted after the private defense to a high-impact journal (6).

    4. Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. **Comparing decomposition-based and automatically component-wise designed multi-objective evolutionary algorithms.** In A. Gaspar-Cunha et al., editors, *Evolutionary Multi-Criterion Optimization* (EMO 2015), volume 9018 of LNCS, pages 396-410. Springer International Publishing, 2015.
    5. Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. **To DE or not to DE? Multi-objective differential evolution revisited from a component-wise perspective.** In A. Gaspar-Cunha et al., editors, *Evolutionary Multi-Criterion Optimization* (EMO 2015), volume 9018 of LNCS, pages 48-63. Springer International Publishing, 2015.
    6. Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. **A performance assessment of tuned multi- and many-objective evolutionary algorithms.** To be submitted to *Evolutionary Computation* (2016).

- The augmented MOEA template, the state-of-the-art AutoMOEAs, and the ablation analysis described in Chapter 5 will be submitted to a high-impact journal after the thesis defense.

    7. Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. **Automatically designing and understanding effective evolutionary algorithms for multi- and many-objective optimization.** To be submitted.

Other publications concerning related topics are listed below, published during the course of this research work either as a collective work between the author and his supervisors (8 and 10), or as collaboration between the author and other research groups (9). However, they do not relate to any chapter as their subject does not fall within the scope of the thesis. In particular, the first two of these papers concern multi-objective ant colony optimization (MOACO) algorithms, addressing the extension of a MOACO framework and the automatic design of MOACO algorithms for the bi-objective bidimensional knapsack problem (8), and a component-wise analysis of MOACO components for the point-to-point multi-objective shortest path problem (9). The bi-objective bidimensional knapsack problem is also the subject of the last paper (10), where we conduct a component-wise analysis of a bi-objective stochastic local search method (Pareto local search).

8. Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. **Automatic generation of multi-objective ACO algorithms for the bi-objective knapsack**. In M. Dorigo et al., editors, *Swarm Intelligence* (ANTS 2012), volume 7461 of LNCS, pages 37-48. Springer, Heidelberd, Germany, 2012.

9. Leonardo C. T. Bezerra, Elizabeth F. G. Goldbarg, Luciana S. Buriol, and Marco C. Goldbarg. **Analyzing the impact of MOACO components: An algorithmic study on the multi-objective shortest path problem.** *Expert Systems with Applications*, 40:345-355, 2013.

10. Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. **An analysis of local search for the bi-objective bidimensional knapsack problem.** In Martin Middendorf and Christian Blum, editors, *Evolutionary Computation in Combinatorial Optimization* (EvoCOP 2013), volume 7832 of LNCS, pages 85-96. Springer, Heidelberg, Germany, 2013.

Finally, we remark that several other papers are planned to be submitted as follow-up works from the two journal papers that yet have not been published. We hope the reader appreciates the effort dedicated to this research.

CHAPTER 2

Background

As previously discussed, this thesis covers a wide range of topics ranging from multi-objective optimization to automatic algorithm configuration. In this chapter, we review the most important theoretical foundations for understanding the remaining of the thesis. In particular, we start by reviewing the definition and basic concepts of *multi-objective optimization* in Section 2.1. Specifically, we address the formulation of a multi-objective problem, the most important issues regarding preference relations (particularly Pareto dominance), and the grounds for the performance assessment of multi-objective optimization algorithms. In addition, we detail the benchmark problems we use in this work, namely the MO-PFSP and the continuous functions we adopt. Section 2.2 focuses on metaheuristics and their adaptation to multi-objective optimization. In particular, we first provide a general overview of metaheuristics, as well as the most challenging issues to be considered for the design of effective multi-objective metaheuristics. Next, we detail *evolutionary algorithms* and their multi-objective design evolution, ranging from the first EA applications to multi-objective problems to contemporary topics such as many-objective optimization. Finally, the goal of Section 2.3 is to familiarize the reader with *automated algorithm engineering* research. In particular, we pay special attention to the existing work on automatic algorithm configuration and design, highlighting the works on multi-objective optimization that inspired this thesis. The reader well versed in these subjects may proceed to the following chapters.

2.1 Multi-objective optimization

Multi-objective optimization problems (MOPs) are generalizations of single-objective problems where several objective functions can be simultaneously considered. Examples of MOPs arise everywhere in our daily lives, ranging from business to science, from industrial to sporting applications. Consider, for instance, a company that has clients spread around the world. While it is imperative that consultants reach out to these clients, travel costs must be minimized to ensure profit. In addition, if time windows are considered, minimizing the waiting time of each client is critical for customer satisfaction. By his turn, a salesman entitled to commissions per sale will also try to maximize his total earnings. In fact, when one analyzes MOPs in general, it is easy to profile the most commonly seen objectives. First, nearly all MOPs present an objective related to effectiveness. In business and industry, this is typically profit, whereas in sports it is a performance measure such as time. A second recurring objective (usually conflicting)

concerns the consumption of any resource one deems valuable. In auto sports, high accelerations must be traded off against tyre consumption. In chemical plants, the total products of a reaction must be maximized, while the consumption of reagents needs to be minimized. Finally, other types of objectives may be related to reliability, safety, robustness, or other secondary quality measures of this nature. For instance, high-profit stock portfolios must be assessed as to their risk and, in the Internet, minimizing the route length between package origins and destinations must also account for alternate routes in case links are dropped.

The importance of multi-objective problems in real-world scenarios had been recognized already in the 1960s by the field of multi-criteria decision making (MCDM) [18]. This particular area of operations research deals with the decision process that has to be made when solving an MOP. In particular, the *decision maker* (DM) is a key player in multi-objective optimization because he is ultimately responsible for selecting the solution to be used. Three different decision making approaches may be adopted when solving a multi-objective problem. In an *a priori* approach, the DM defines a preference relation that can be used to compare solutions, usually based on a ranking or aggregation of the different objectives. Conversely, in an *a posteriori* approach a solver first finds the optimal solution subset (or an approximation of this set) according to a given objective-neutral preference relation, and the DM is responsible for selecting from this set the solution(s) that best fit his preferences. More precisely, *dominance relations* are preference relations where no a priori preference between the different objectives is specified, allowing in theory the search to be conducted evenly across the objective space. Finally, an *interactive* approach is a hybrid alternative to the previous approaches in which a dynamic, evolving preference relation is used during the run of the algorithm. Concretely, at intermediate stages of the run the DM is prompted to refine the preference relation used, either via selecting from a subset of the current solution set or by reassessing the importance of the different objectives.

2.1.1 Core definitions

Formally, an MOP can be defined as follows.

Definition 1 (MOP) *A multi-objective optimization problem (MOP) is a tuple* $\Pi = (S, f)$*, where* $f\colon S \to \mathbb{R}^M$ *is an* M*-dimensional objective function,* $M \geq 2$*, that maps* S*, the set of candidate solutions for* Π*, also called* decision space*, to* $\mathbb{R}^M$*. The image of* S *is also called* objective space*.*

A MOP (as optimization problems in general) may vary as to the domain and possible constraints of the variables that define it. Concerning the variable domain, *continuous* problems present real-valued decision variables, whereas the domains of variables in *combinatorial* problems are discrete. Regarding constraints, the decision variables of an *unconstrained* problem may assume any values (from their domains). By contrast, a *constrained* optimization problem presents constraints that restrict the values that some (or all) variables may assume. As a consequence, the interest of optimizers dealing with constrained problems concerns the subset *feasible*(S) comprising only *feasible* solutions, i.e., solutions that do not violate constraints.

Since the dimensionality of f is greater than one, each feasible solution has an associated *objective vector*, i.e., its image in $\mathbb{R}^M$ (M is also called the number of objectives). Comparing solutions in an MOP is then a task of comparing their objective vectors. Not surprisingly, the often contradicting nature of objectives usually leads to the situation where it is impossible to clearly decide between two objective vectors. In particular, when the optimization problem considered presents an evaluation function with $M = 1$, the algebraic relations $\leq$ and $\geq$ establish natural total orders over the solution quality domain $\mathbb{R}$, with a few different solution subsets presenting the same quality. Nonetheless, single-objective optimization problems always present a single globally optimal point in the objective space, even if several solutions map that same quality value. Analogously, in order to tackle an MOP one must first choose a

preference relation to order the objective space $\mathbb{R}^M$, even if a neutral one such as a dominance relation. We then define two important concepts that derive from using a preference relation.

Definition 2 (Ψ-efficient objective vector and solution) *Given a preference relation Ψ and an objective vector $\mathbf{v}$ that is the image of a feasible solution $s \in feasible(S)$, $\mathbf{v}$ is called a* Ψ-efficient *objective vector if it is minimal (maximal) according to Ψ. In addition, a solution that maps to a Ψ-efficient vector is called a* Ψ-efficient *solution.*

As in single-objective optimization, it is possible to have subsets of solutions that are equivalent according to Ψ, called Ψ-*indifferent* solutions. In addition, in multi-objective optimization one also deals with the notion of Ψ-*incomparability*, i.e., vectors that are neither better, worse, nor indifferent to each other according to Ψ. As a result, there usually exists a set of Ψ-incomparable efficient objective vectors in an MOP, and a possibly even larger set of Ψ-incomparable efficient solutions. The goal of a multi-objective optimizer is hence to retrieve a Ψ-optimal set or front for a given preference relation Ψ, as follows.

Definition 3 (Ψ-optimal set) *The Ψ-optimal set $\Psi_{set} \subseteq S$ is the set of Ψ-efficient solutions.*

Definition 4 (Ψ-optimal front) *The Ψ-optimal front $\Psi_{front} \subseteq \mathbb{R}^M$ is the set of Ψ-efficient vectors.*

In general, multi-objective optimization problems are at least NP-hard, even when the single-objective problems from which they have been generalized are polynomially solvable [209]. An evidence of this increased complexity concerns single-objective projections, or *scalarizations*, a technique that aggregates the different objectives and is often employed in the literature of multi-criteria decision making. For a particular type of aggregation techniques, the optimal solution for a single-objective projection of an MOP is also dominance-optimal for the original MOP. In fact, the literature establishes a separation between solutions that can be retrieved via linear scalarizations of the MOP and solutions that cannot, respectively referred to as *supported* and *non-supported*. In other words, a supported efficient solution of Π is a solution that belongs to its convex hull. For a particular class of MOPs that originates from the generalization of polynomially solvable optimization problems, these can be retrieved in polynomial time. In this type of MOPs, the goal of the multi-objective optimizer is made easier since supported efficient solutions might suffice for a DM to choose from. For MOPs in general, however, even the task of finding supported solutions is NP-hard. As a result, multi-objective optimization algorithms are often approximative, not ensuring the optimality of the solutions contained in the set/front they return.

Definition 5 (Approximation set) *An* approximation set *$A^{\Psi}_{set} \subseteq S$ is a set of solutions (decision vectors) returned as the output*[1] *of a multi-objective approximative (heuristic) algorithm meant to optimize an MOP according to a given preference relation Ψ.*

Definition 6 (Approximation front) *An* approximation front *$A^{\Psi}_{front} = f(A^{\Psi}_{set})$ is the image (objective vectors) of a given approximation set A^{Ψ}_{set}.*

Overall, it becomes clear that the role of preference relations in solution comparison is critical for the design of multi-objective optimization algorithms. Next, we discuss the most relevant categories of preferences used in the literature.

2.1.2 Preferences and solution comparison

The highest-level separation between preference relations concern the importance given to objectives: while *dominance relations* consider objectives equally important, the remaining preference relations establish some order of priority over them[2]. Among the latter, the most basic form of ordering solutions

[1] The set of solutions maintained by a population-based algorithm during its execution is also called an approximation set.

[2] It is however possible to have a special-case of *scalarization-based* preference relations where objectives are considered equally important, as we will later discuss.

in an MOP is lexicographically, and it is of particular interest for exact algorithms. For the multi-objective shortest path (MSP) problem, for instance, a generalized version of Dijkstra's algorithm has been devised and showed to be efficient [169]. The proposed algorithm works by creating/expanding labels representing partial solutions on the graph nodes. However, the optimality of the complete solutions found is only ensured if all of the labels generated by the algorithm are expanded in lexicographical order of their associated objective vectors.

Next we discuss in more detail the two most relevant families of preference relations, namely *aggregation* and *dominance* relations.

Aggregation relations

Besides the natural approach of lexicographic ordering, another set of common preference relations in MCDM that establishes a preference over objectives comprises *weighted aggregations*, *utility functions*, and *boundary intersection*. All these approaches have in common the advantage of converting the original MOP to single-objective projections. Among their benefits, one can highlight the reduced computational complexity of the problem, meaning a much lower computational effort to find high-quality solutions. In addition, by using multiple weight vectors one can retrieve multiple solutions from a single run when using population-based metaheuristics, for instance. However, a major drawback related to this kind of approach is that not all search directions will necessarily lead to optimal solutions. As a consequence, much effort might be wasted by the algorithm, hindering its overall performance. We then further detail each of them:

- **Weighted aggregations** use scalar values to aggregate the information concerning each objective[3], the most common example being linear weighted sum [180]. Given a weight vector λ, the linear weighted sum δ_{lws} of an objective vector $\mathbf{v}$ is given by:

$$\delta_{lws}(\mathbf{v}, \lambda) = \sum_{i=1}^{M} v_i \cdot \lambda_i, \quad \text{subject to} \quad \lambda_i \in [0,1], \quad \sum_{i=1}^{M} \lambda_i = 1 \tag{2.1}$$

- **Utility functions** are more elaborate mathematical functions used to evaluate the preferability of solutions. For example, the Tchebycheff function [177] uses both a weight vector λ and a reference point $\mathbf{r}$ to measure the distance between an objective vector $\mathbf{v}$ and a given region of interest of the objective space:

$$\delta_{Tch}(\mathbf{v}, \lambda, \mathbf{r}) = \max\{\lambda_i \cdot (v_i - r_i)\}, \quad \text{subject to} \quad \lambda_i \in [0,1], \quad \sum_{i=1}^{M} \lambda_i = 1 \tag{2.2}$$

- **Boundary intersection** is an approach based on the fact that the intersections between reference lines defined by a set of weights and the boundaries of the attainable objective space represent optimal solutions. This way, the search process of an algorithm is directed towards minimizing the distance between the current solution and the boundary intersection point, defined by the weight vector used. However, to prevent solutions from converging to the same point in the objective space one may use a penalty factor ξ, as in the *penalty-based boundary intersection* (PBI) approach [228]:

$$\begin{gathered}\delta_{pbi}(\mathbf{v}, \lambda, \mathbf{r}, \xi) = d_1 + \xi \cdot d_2, \quad d_1 = \frac{||(\mathbf{r}-\mathbf{v})^T \cdot \lambda||}{||\lambda||}, \quad d_2 = ||\mathbf{v} - (\mathbf{r} - d_1\lambda)|| \\ \text{subject to} \quad \lambda_i \in [0,1], \quad \textstyle\sum_{i=1}^{M} \lambda_i = 1\end{gathered} \tag{2.3}$$

[3] In multi-criteria decision making, this technique is often used to express the preferences of the decision maker, where weights refer to the relative importance of the different objectives.

where, $\mathbf{v}$ and $\mathbf{r}$ are a reference point and an objective vector, respectively, and d_1 and d_2 respectively represent the distance between $\mathbf{v}$ and the boundary intersection, and between $\mathbf{v}$ and the reference line defined by λ.

Dominance relations

Concerning dominance relations, no distinction between objectives is made, as previously discussed. The most commonly used are Pareto [84] and Lorenz [179] dominance, although clearly the works using the former far outnumber the latter. In fact, several variants of the original Pareto dominance relation have been proposed, although they escape the scope of this thesis. Concerning Pareto and Lorenz dominance, they are both illustrated in Fig. 2.1.

Definition 7 *(Pareto dominance) Given two different candidate solutions s_1 and s_2 of an MOP, the Pareto dominance relation states that s_1 dominates s_2 if, and only if, s_1 is not worse than s_2 for all objectives and strictly better for at least one. Formally, for a minimization problem (without loss of generality),*

$$s_1 \prec s_2 \iff \forall m = 1, \ldots, M, \quad f^m(s_1) \leq f^m(s_2), \text{ and } \exists j \text{ such that } f^j(s_1) < f^j(s_2)$$

The geometrical interpretation of Pareto dominance can be seen on Figure 2.1 (left). Given two solutions s_A and s_B of a minimization problem, any other solution $s_C \neq s_A \neq s_B$ located within the gray area of the plot will be dominated either by s_A, by s_B, or by both.

By contrast, Lorenz dominance compares solutions not based on their objective vectors, but on their *generalized Lorenz* vectors.

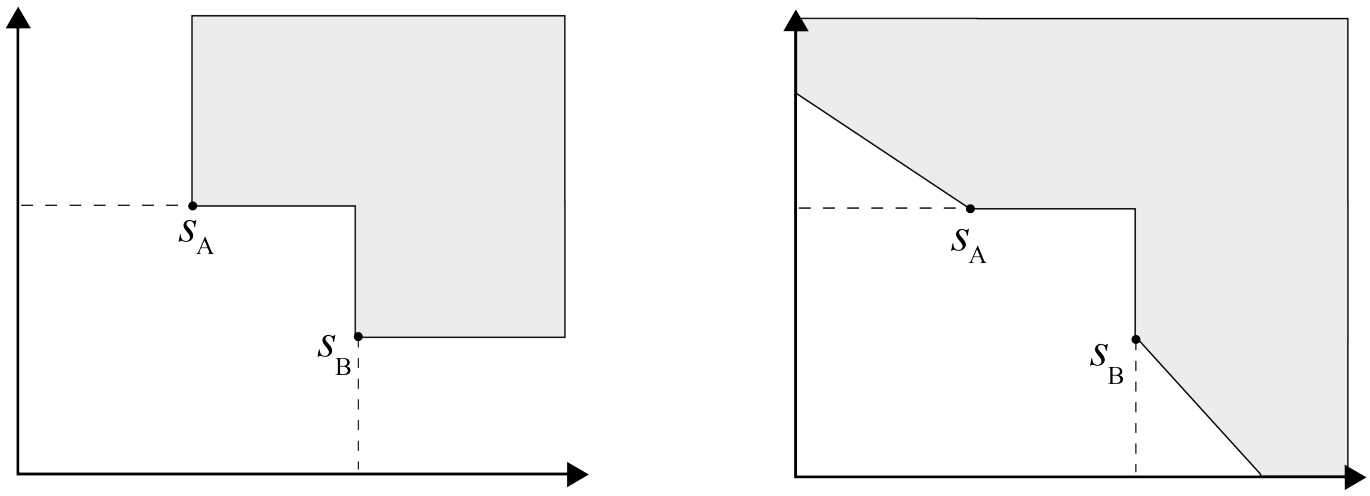
Definition 8 (Generalized Lorenz vectors) *Given an objective vector $\mathbf{v}$ and its i-th largest component $\theta_i(\mathbf{v})$, the generalized Lorenz vector $\overline{\Theta}(\mathbf{v})$ is defined as*

$$\overline{\Theta}(\mathbf{v}) = (\overline{\theta}_1(\mathbf{v}), \overline{\theta}_2(\mathbf{v}), ..., \overline{\theta}_M(\mathbf{v})) \quad \text{with} \quad \overline{\theta}_i(\mathbf{v}) = \sum_{j=1}^{i} \theta_j(\mathbf{v}) \tag{2.4}$$

The geometrical interpretation of Lorenz dominance can be seen on Figure 2.1 (right). In particular, the region dominated by a solution s according to Pareto dominance is a subspace of the region dominated by s according to Lorenz dominance. As a result, Lorenz dominance is considered more strict than Pareto dominance, although it is clear that solutions that are Lorenz-incomparable will also be Pareto-incomparable. For this reason, a recent empirical investigation has compared the effects of Lorenz versus Pareto dominance w.r.t the increase in the number of objectives and as an additional source of convergence pressure to algorithms [181]. Although the cardinality of the Lorenz fronts remains reasonable, their spread is considerably lower than when Pareto dominance is adopted. Since this is still a rather incipient research direction, we adopt Pareto dominance in the remainder of this thesis, which we further detail next.

2.1.3 Pareto dominance

Besides the previously discussed relations between solutions, Pareto dominance presents additional nuances for solution comparison, namely *strict* and *weak* dominance, which we formally detail in Table 2.1. In particular, the weak Pareto dominance relation only assures that a solution is not worse than another. All Pareto dominance solution comparison relations are generalized for the comparison of solution sets. The baseline Pareto dominance relation for set comparison can be defined as follows.



(a) Pareto (b) Lorenz

Figure 2.1: Difference between the dominated subspaces according to the dominance relation adopted.

Table 2.1: Pareto dominance relations among solutions [140].

Relation		**Interpretation in the objective space**
Strict dominance	$s_1 \prec\prec s_2$	s_1 is better than s_2 for all objectives
Dominance	$s_1 \prec s_2$	s_1 is not worse than s_2 for all objectives and is better for at least one
Weak dominance	$s_1 \preceq s_2$	s_1 is not worse than s_2 for all objectives
Incomparability	$s_1 \mid\mid s_2$	Neither $s_1 \preceq s_2$ nor $s_2 \preceq s_1$
Indifference	$s_1 \sim s_2$	s_1 has the same value as s_2 for all objectives

Definition 9 (Pareto set dominance relation) *Given two sets of solutions A and B, A is said to dominate B ($A \prec B$) if, and only if, for every solution $s_b \in B$ there exists a solution $s_a \in A$ such that $s_a \prec s_b$.*

The definitions of set strict-, weak-, and non-dominance are analogous to the baseline definition and are formally defined in Table 2.2. Moreover, an extra relation has been defined for comparing sets, namely the *better* ($\lhd$) relation[4].

Definition 10 (Pareto better dominance relation) *Given two sets of solutions A and B, A is said to be better than B ($A \lhd B$) if, and only if, for every solution $s_b \in B$ there exists a solution $s_a \in A$ such that $s_a \preceq s_b$, but A is not identical to B.*

The goal in MOPs that are tackled according to Pareto dominance is then to identify the *Pareto-optimal set* and its associated Pareto-optimal front. Notice that the cardinality of the Pareto front may be smaller than the cardinality of the Pareto set, since different solutions may map to the same objective vector. For this reason, algorithms generally aim at retrieving the Pareto front rather than the Pareto set, even though different Ψ-indifferent solutions could be of interest to decision makers. In practice, however, the previously discussed MOP computational complexity means that this goal is relaxed towards finding an as good approximation front as possible, although we will next discuss that this is a complex subject.

[4]We remark that the symbols $\prec\prec$, $\prec$, $\preceq$, $\lhd$ are interchangeably used in the literature with their symmetric symbols $\succ\succ$, $\succ$, $\succeq$, $\rhd$. In this work, we adopt the convention that these symbols should be used as generalizations of the $<$ and $>$ symbols, and without loss of generality assume minimization for the remainder of the thesis ($\prec\prec$, $\prec$, $\preceq$, $\lhd$).

Table 2.2: Pareto set dominance relations [140].

Relation		**Interpretation in the objective space**
Strict dominance	$A \prec\prec B$	Every $s_2 \in B$ is strictly dominated by at least one $s_1 \in A$
Dominance	$A \prec B$	Every $s_2 \in B$ is dominated by at least one $s_1 \in A$
Better	$A \lhd B$	Every $s_2 \in B$ is dominated by at least one $s_1 \in A$ and $A \nsim B$
Weak dominance	$A \preceq B$	Every $s_2 \in B$ is weakly dominated by at least one $s_1 \in A$
Incomparability	$A \mid\mid B$	Neither $A \preceq B$ nor $B \preceq A$
Indifference	$A \sim B$	$A \preceq B$ and $B \preceq A$

2.1.4 Performance assessment

The concept of high-quality approximation fronts is not straightforward. In the best-case scenario, every front generated by one optimizer Φ_1 strictly dominates every front generated by another optimizer Φ_2. In practice, however, this is rarely the case. In order to evaluate approximation fronts, several methodologies can be used, such as *dominance rankings* [140], quality metrics (or *indicators*) [123, 129, 206, 235], and *empirical attainment functions* (EAFs) [82, 158].

Dominance rankings

A rigorous comparison between the approximation fronts produced by different MOEAs is to use dominance rankings. In more detail, let Φ_1 and Φ_2 be two algorithms one wants to compare and $A^1_{\Phi_1}, A^2_{\Phi_1}, \ldots, A^r_{\Phi_1}$ and $A^1_{\Phi_2}, A^2_{\Phi_2}, \ldots, A^r_{\Phi_2}$ the approximation fronts they respectively produce over a series of r runs, comprising a collection $\mathbb{C}$. Each of these fronts is assigned a dominance ranking depicting how many fronts from $\mathbb{C}$ are better ($\lhd$) than it. In this way, both algorithms can now be evaluated based on the ranking values their approximation fronts achieve. In particular, statistical analysis can investigate whether this transformed sample for Φ_1 is significantly different from the sample for Φ_2, and post-hoc tests can indicate which algorithm produces better quality fronts if difference is observed [140].

As previously mentioned, this is a fairly rigorous approach since it is only possible to discriminate between algorithms that present very different performance. Therefore, to assess whether an approximation front is better than other, a common approach is to use quality indicators, as we shall see next.

Quality indicators

Measuring the quality of an approximation front is a complex task. One approach that has drawn significant attention from the research community is the use of quality indicators (or performance metrics). Concretely, these metrics either (i) analytically measure a given (set of) characteristic(s) a high-quality front should present, or (ii) analytically assess the difference between two fronts, in which case it is possible to evaluate how well a front approximates the Pareto front. In the former case, a metric is said to be unary, whereas in the latter it is said to be binary, as we formally defined next.

Definition 11 (Unary and binary quality indicators) *Let Ω be the set of all possible approximation fronts of an MOP Π. A* unary quality indicator *$\mathcal{I}$ is a function $\mathcal{I}\colon \Omega \to \mathbb{R}$ that measures a particular characteristic of approximation fronts and assigns them scalar values. A* binary quality indicator *is a function $\mathcal{I}\colon (\Omega, \Omega) \to \mathbb{R}$ that analytically measures the similarity between two approximation fronts.*

As we will see later, some binary quality indicators can be used to construct as unary quality indicators. More importantly, a requirement that should be observed by indicators concerns their agreement with Pareto dominance, formally defined as follows.

Definition 12 (Pareto compliance) *Let $\mathcal{I}\colon \Omega \to \mathbb{R}$ be a quality indicator which is to be maximized. $\mathcal{I}$ is said to be Pareto-compliant if, and only if, for every pair of approximation fronts $(A, B) \in \Omega$ for which $I(A) \geq I(B)$, it also holds that $B \not\preceq A$.*

As discussed by Zitzler et al. [235], true unary quality indicators display a limited potential for comparing sets of solutions while respecting Pareto dominance. By contrast, some binary indicators can deliver Pareto compliance, but the amount of data produced when analyzing a set of algorithms with multiple runs using binary indicators can be overwhelming. For these reasons, the most appropriate approaches to quality indicators are either to (i) use binary indicators as an auxiliary method for computing dominance rankings, or; (ii) reformulate binary indicators as unary indicators considering the comparison between an approximation front and a reference front. For instance, consider the additive binary ϵ-indicator ($I_{\epsilon+}$) [235] which we will shortly explain. Being a binary indicator, the $I_{\epsilon+}$ takes two input approximation fronts to compare. However, it is possible to use it as an unary indicator as $I^1_{\epsilon+}(A) = I_{\epsilon+}(A, R)$, where R is a reference front [140].

The ideal reference fronts to be used by quality indicators are true Pareto fronts. However, in combinatorial optimization it is often the case that one cannot compute these directly given the NP-hardness of the original single-objective problems. For continuous problems, the artificially designed problems typically considered for multi-objective optimization present backdoors that allow Pareto optimal solutions to be easily generated. However, these Pareto fronts can be too large for practical purposes depending on the tolerance level used and the correlation between the different objectives. Whenever Pareto fronts are not available, reference sets can be assembled by merging all approximation fronts found by all optimizers being assessed. At this point, one can either (i) filter these supersets to leave only nondominated solutions or (ii) generate "average fronts" via different methods. Although this approach is far from ideal, reference sets can become rich information sources as long as they are continuously refined by adding solutions found by high-performing algorithms. Next, we detail and discuss the most important indicator proposals observed in the literature:

1. **The hypervolume indicators** [231, 235]: In its original unary version (I_H), the hypervolume indicator measures the volume of the subspace dominated by a given approximation front bounded by a reference point. The larger the hypervolume, the better the approximation front is. Given the drawbacks already discussed for unary indicators, different binary versions of this metric have been adopted in the literature, as follows:

- *The hypervolume difference* (I^-_H or I_{HD}) [231]: given two approximation fronts A and B, the I_{HD} computes the difference between the hypervolume of the subspace bounded by a reference point $\mathbf{r}$ that is dominated by A and but not by B. Used as an unary indicator with a reference front R as B, we have the following minimization metric:

$$I^1_{HD}(A) = I_{HD}(R) - I_{HD}(A) \tag{2.5}$$

- *The relative hypervolume* ($I_{H\%}$) [231]: given two approximation fronts A and B, the $I_{H\%}$ computes the ratio between the hypervolume of the subspace dominated by A and the hypervolume of the subspace dominated by B. Used as an unary indicator, we have the following minimization metric:

$$I^1_{H\%}(A) = \frac{I_H(A)}{I_H(R)} \tag{2.6}$$

- *The hypervolume relative percentage deviation* (I_{HR}) [32]: given two approximation fronts A and B, the I_{HR} computes the relative percentage deviation between the hypervolume of the subspace dominated by A and the hypervolume of the subspace dominated by B. Used as an

unary indicator, we have the following minimization metric:

$$I_H^{rpd}(A) = \frac{I_H(R) - I_H(A)}{I_H(R)} \tag{2.7}$$

The hypervolume-based indicators are the most used in the literature due to their Pareto-compliance, as long as the reference point used is strictly dominated by the approximation front [140]. In fact, the hypervolume is the only indicator that has been shown to present this property[5]. On the other hand, the most significant challenge concerning hypervolume-based indicators is the computational complexity of computing the hypervolume for an approximation front, which is exponential in the number of objectives [20]. However, significant work towards reducing the overhead imposed by this indicator has been able to produce much improved implementations [20, 183, 223, 224]. Finally, an additional challenge is evaluating the region of interest for these indicators as the number of objectives grows. For instance, hypervolume implementations have to deal with overflows when addressing large numbers of objectives, such is the increase in the hypervolumes. Consequently, the range of values that the unary hypervolume metrics display becomes considerably small.

2. **The ϵ-indicators** (I_ϵ and $I_{\epsilon+}$) [235]: originally defined as binary metrics, the additive ($I_{\epsilon+}$) and multiplicative (I_ϵ) ϵ-indicators measure the translation (additive) or scaling (multiplicative) that would be required for an approximation front A to become weakly dominated by another front B. Used as unary indicators, we have the following minimization metrics [140]:

$$I_\epsilon^1(A) = I_\epsilon(A, R) \qquad\qquad I_{\epsilon+}^1(A) = I_{\epsilon+}(A, R) \tag{2.8}$$

   The binary ϵ-indicators are able to assess whether an approximation front is better ($\lhd$) than another, being weakly Pareto-compliant [235]. In addition, used as unary indicators they are able to assess whether an approximation front has converged to the reference front with ν tolerance. However, it is not possible to determine whether an $I_{\epsilon+}^1 > \nu$ ($I_\epsilon^1 > \nu$) means a lack of convergence or diversity. Furthermore, compared to the hypervolume indicator the computational overhead posed by the ϵ-indicators is minimal.

3. **The generational distance indicators** (I_{GD}, I_{IGD}, and I_{IGD+}) [49, 123, 220]: the first proposed member of this family of binary indicators was the generational distance indicator (I_{GD}) [220], a minimizing convergence metric that measures the average Euclidean distance between each point of an approximation front A and the closest point to it from another front B. This metric strictly concerns convergence, and is generally used as an unary metric comparing the distance between an approximation front and a reference front. Later, the *inverted generational distance* (I_{IGD}) [49] was proposed as a unary metric that accounts for both convergence and diversity. Concretely, the I_{IGD} is defined in function of the I_{GD} indicator:

$$I_{IGD}(A) = I_{GD}(R, A) \tag{2.9}$$

   In other words, the inverted generational distance of a given front A w.r.t. an approximation front R is actually the generational distance between R and A. Similarly to the ϵ-indicators, the computational overhead required for computing I_{IGD} is minimal when compared to the effort required for the hypervolume computation. However, although this indicator is the most widely adopted in the field of *many-objective optimization*, it is not Pareto compliant [123]. Recently, a Pareto-compliant version of the I_{IGD} was proposed (I_{IGD+}, [123]), differing by a specific feature of

[5] We will later see that the $I_{\epsilon+}$ indicator has also been shown to be Pareto-compliant, but only weakly.

the distance computation. Given an approximation front A one wants to evaluate and a reference front R, a modified Euclidean distance calculation between points $a \in A$ and $z \in R$ is used, which considers only the objective vector components for which a is dominated by z. Formally, the distance between a and z for a given objective i is computed as follows (considering a minimization problem without loss of generality):

$$d_i^+(z, a) = max\{a_i - z_i, 0\}, \quad i = 1, 2, \ldots, M \tag{2.10}$$

As discussed above, each metric is a different approach to measure the quality of an approximation front. Given that different metrics may favor different characteristics, it is not surprising that they eventually disagree, as an empirical investigation has recently shown [129]. In order to understand the consistencies and contradictions between metrics, we first define more precisely the characteristics most commonly used to analyze approximation fronts:

Convergence: refers to the optimality of solutions. A front is said to have converged if all of its solutions are Pareto-optimal.

Spread: refers to the extent of the front, more specifically the distance between the extreme solutions of a front. It can be used to assess whether an algorithm faces difficulties in a given objective, like in combinatorial optimization problems where the objectives may differ considerably as to their nature.

Distribution: refers to the evenness of the front, more specifically to the uniformity of the distances between pairs of adjacent solutions. In particular, analyzing distribution is a way of detecting gaps in the fronts, meaning that the algorithm had difficulties to optimize some combination of the objectives.

We then review the most important findings of the empirical investigation that reported the contradictions between performance metrics [129]. In particular, the authors used a set of different metrics to evaluate two different Pareto optimal fronts, differing between each other because the second had been post-processed to improve its I_H value. We remark that the findings were observed for continuous problems where fronts have smooth shapes, and hence it is not clear how they generalize to the tricky fronts of some combinatorial problems. In addition, we restrict our review to the metrics that were both present in that study and that we have previously explained (I_H, $I_{\epsilon+}$, I_{IGD}).

The key features behind the interactions between metrics concern the geometry of the Pareto fronts and the correlations between objectives ($0.1 \leq \rho \leq 3.0$), as follows:

Convex problems: The I_H and the I_{IGD} agree in these problems, valuing distribution over spread. By contrast, $I_{\epsilon+}$ increasingly disagrees with the remaining indicators as the number of objectives grows. For $M = 3$, $I_{\epsilon+}$ favors spread over distribution for half of the correlation values considered ($0.1 \leq \rho < 0.5$). Moreover, already for $M = 4$, $I_{\epsilon+}$ disagrees with the remaining metrics for most correlation values considered.

Concave problems: The I_H and the $I_{\epsilon+}$ value spread over distribution. Conversely, the I_{IGD} disagrees with these metrics for all correlation values considered whatever the number of objectives.

Empirical attainment functions

Besides quality indicators, another essential methodology for comparing multi-objective optimizers is the use of *empirical attainment functions* (EAFs) [158, 235]. To understand this concept, one must first understand the concepts of *attainment functions* and *empirical attainment surfaces*. *Attainment*

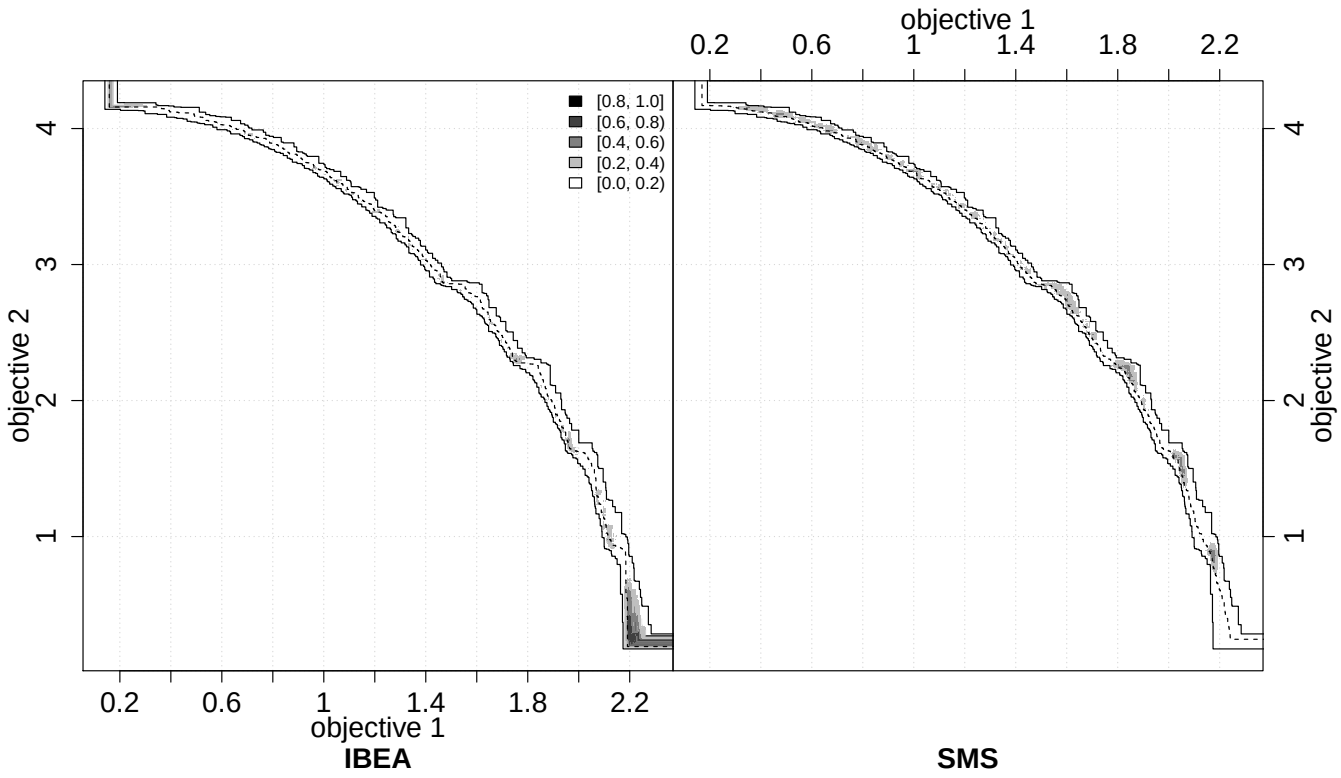


Figure 2.2: EAF difference plots comparing two multi-objective optimization algorithms. Darker areas on the left indicate that the fronts from the algorithm depicted on the left have better spread more often than the fronts from the algorithm depicted on the right.

functions measure the percentage of times an algorithm generates fronts that dominate each vector of the objective space. Since it is not feasible to compute exact attainment functions, in practice one estimates empirical attainment functions from the available approximation fronts. This information is generally used for comparing multiple optimizers through statistical tests [140].

Having calculated the empirical attainment functions for the data available from a set of optimizers, one can compute their *empirical attainment surfaces*. Attainment surfaces are used to delimit which parts of the objective space have been attained up to a frequency rate k. For example, a 10%-attainment surface delimits which parts of the objective space have been attained by at least 10% of the runs of a given algorithm. Computing several of these attainment surfaces, one can use this data for graphical visualization.

The most important graphical visualization methodology currently using empirical attainment surfaces are the EAF difference plots [158]. This type of plot is the first choice when directly comparing two algorithms because they resemble density distribution differences showing which algorithm has attained more often each region of the objective space. An example is shown in Figure 2.2. Each side of the figure shows the resulting EAF difference for each algorithm being compared. Darker areas on the left box indicate that the fronts from the algorithm depicted on the left have better spread more often than the fronts from the algorithm depicted on the right. Additional information can also be retrieved from this type of plots. The inner and outer continuous lines on both sides of the figure respectively represent the worst-case and the best-case attainment surfaces for both algorithms. The dashed line represents the median attainment surface of each algorithm. Despite the large amount of information they can depict, EAF difference plots face a major conceptual limitation. So far, these tools have only been of practical use for bi-objective optimization.

2.1.5 Application MOPs

We next formally define the MOPs we consider as application problems in this thesis. We start by discussing the most important concepts concerning continuous optimization and by reviewing the characteristics of the existing artificial problem benchmarks used in the literature for this field. Concerning combinatorial optimization, we select as application the multi-objective permutation flowshop problem (MO-PFSP). As we will discuss, a number of multi-objective metaheuristics has been applied to the MO-PFSP due to the particular characteristics presented by the most commonly used objective functions of this problem.

Continuous optimization

The domain of continuous optimization involves all application problems that can be modeled as a mathematical function dependent on real-valued variables. Although confusing, a multi-objective continuous problem is simply referred to as an MOP in the literature. Formally, an MOP with M objectives and n_{var} variables is defined as[6]:

$$\min f(\mathbf{s}) = (f_1(\mathbf{s}), f_2(\mathbf{s}), \ldots, f_M(\mathbf{s})) \tag{2.11}$$

where $\mathbf{s} \in \mathit{feasible}(S)$ is a solution (a decision vector) and each $f_i : \mathit{feasible}(S) \rightarrow \mathbb{R}$ is an objective function to be optimized.

Although this is a field rich in real-world applications, most studies rely on artificial benchmark problem sets. Before reviewing the most relevant sets observed in the literature, we discuss the most recurring characteristics present in real-world objective functions and problems that these sets try to capture [61, 112].

1. **Separability**: separable objective functions allow the optimization task to be done iteratively by fixing a few variables and optimizing others. In multi-objective optimization, it is also common to have a separation between (i) variables that affect the optimality of the problem, thus leading to novel dominating solutions (*distance*-related variables), and; (ii) variables that when modified can only lead to incomparable solutions, and hence affect only the search diversity (*position*-related variables).

2. **Local Pareto-optimal fronts**: a locally optimal front is a surface that tricks algorithms into believing that they have found the true Pareto front of an MOP, slowing down their progress. When a problem presents a large number of local Pareto-optimal fronts, algorithms may even be unable to converge to the true front.

3. **Modality**: unimodal objective functions present a single, globally optimal objective vector. Conversely, multi-modal functions are more difficult to optimize because they present a series of local optima that can trick an algorithm or at least slow down its convergence. In addition, multi-modal functions are more recurring in the real-world than unimodal ones. Finally, a special type of multi-modality is known as *deceptiveness*. A deceptive objective presents only two optima, and the majority of the search space favors the local one, deceiving algorithms in their search for the global optimum.

4. **Density bias**: objective space bias means that there is a shift in the density distribution of the objective space meant to test algorithms for their ability to keep the search well-spread. In particular, the biased regions of the objective space may act as attractive poles at any point during the search process and, hence, this feature constitutes an important challenge for optimizing MOPs.

[6] Notice we assume minimization for all objectives without loss of generality.

MOP	Separability (1)	Local fronts (2)	Modality (3)	Bias (4)	Geometry (5)
DTLZ1	✓*	✓	M	–	linear
DTLZ2	✓*	–	U	–	concave
DTLZ3	✓*	✓	M	–	concave
DTLZ4	✓*	–	U	polynomial	concave
DTLZ5	✓*	–	U	–	degenerate?
DTLZ6	✓*	✓	U	–	degenerate?
DTLZ7	✓	–	U,M	–	disconnected
WFG1	✓	–	U	polynomial	convex, mixed
WFG2	–	–	U,M	–	convex, disconnected
WFG3	–	–	U	–	linear, degenerate
WFG4	✓	–	M	–	concave
WFG5	✓	–	D	–	concave
WFG6	–	–	U	–	concave
WFG7	✓	–	U	parameter dependent	concave
WFG8	–	–	U	parameter dependent	concave
WFG9	–	–	M,D	parameter dependent	concave

Table 2.3: Summarized description of two problem benchmark sets commonly used in the literature of multi-objective continuous optimization: DTLZ [61] and WFG [112].

5. **Pareto front geometry**: a series of different Pareto front geometries can be observed in real-world problems, namely convex, mixed, linear, degenerate, and disconnected. It is also possible to have combinations of these geometries, and thus evaluate the ability of algorithms to simultaneously deal with them. In particular, we briefly discuss the concepts of degenerate and disconnected fronts, as they are not as obvious as the others.

   - A *degenerate* front presents a dimensionality that is, at least, two dimensions lower than the dimensionality of the problem considered. For instance, a degenerate front of a three-objective problem would be a line;
   - A *disconnected* front presents pools of Pareto-optimal solutions, and tests the ability of algorithms to simultaneously approximate multiple trade-off regions of the objective space.

6. **Correlation between the objectives**: objectives that present high correlation are easier to optimize because there is a high probability that when one objective is improved, the positively correlated one(s) is (are) also improved. By contrast, when a problem presents conflicting objectives converging towards the Pareto front requires being able to simultaneously optimize them.

A number of benchmark problem sets has been proposed in the literature of continuous optimization [12, 61, 112, 229], and an even larger number of independent benchmark problems [112]. In this chapter, we restrict our discussion to two benchmarks that we select because they are scalable as to the number of variables (n_{var}) and objectives (M), namely the DTLZ [61] and the WFG [112] benchmark sets. Both benchmark sets present problems that are box-constrained, i.e., their variables can only take values within a bounded real interval. A summarized detail of the problems contained in these sets is given in Table 2.3, following the analysis of Huband et al. [112]. In particular, we remark that computing correlations between objectives over a range of different M values would not fit this section due to the amount of pairwise analysis required by this approach. The most important overall remarks about the selected benchmark problems sets are discussed next:

The DTLZ benchmark: this problem benchmark represents an important milestone when one analyzes the multi-objective optimization performance assessment literature. In particular, this was the first benchmark problem set simultaneously scalable w.r.t. M and n_{var}, although increasing

n_{var} only increases the number of distance-related variables. In addition, the large variety of problem characteristics assessed by this benchmark set has been instrumental for helping design robust algorithms. However, a few disadvantages have to be remarked:

- *Extremal optima* – for many DTLZ problems, Pareto optimal solutions are found when distance-related parameters are set to the lower bound of the domain of values variables can take. Truncation procedures used by algorithms when variables violate bounds can then lead to good results simply due to this characteristic.
- *Separability* – functions are considered separable from a practical point of view since it is possible to find at least one global optimum when optimizing variables independently.
- *Geometry* – the DTLZ5–6 problems present design issues that affect their analysis [112], and hence when $M > 3$ it is not possible to assess their true geometry.
- *Flexibility* – fiddling with the DTLZ functions to regulate the number of Pareto-local optima or other problem characteristics is not a straightforward task.
- *Problem sizes* – although this benchmark was proposed to be scalable as to n_{var}, very few works have considered problem sizes different from the ones suggested by the DTLZ proposers, which can be very low for practical purposes ($n_{\text{var}} < 10$).

The WFG benchmark: proposed a few years later than the DTLZ, the WFG set was proposed as an illustrative example of a flexible toolkit designed to allow researchers to freely construct or modify MOPs. In particular, a WFG problem is the result of a series of composable transformations that define the geometry, bias, and other characteristics the problem presents. In practice, however, the use of the WFG toolkit has been limited, and researchers have attained to the illustrative benchmark problem set defined by the WFG proposers. Even more so, the major drawback with this problem set is precisely the few algorithms that have been benchmarked on it, reducing the available knowledge as to the true difficulty posed by this set.

The multi-objective permutation flowshop problem

The MO-PFSP is a generalization of the permutation flowshop problem, a workshop scheduling problem that arises in any industrial setting where a set of n jobs has to be executed by m different machines in a given machine order. The profit of chemical, steel, or ceramic tile production companies depends directly on the efficiency of the job scheduling order, since each execution takes a different amount of time. An instance Π_{PFSP} of the PFSP is a tuple $\Pi_{\mathsf{PFSP}} = (n, m, P)$, where P is a $n \times m$ matrix of processing times p_{ij} depicting the time required for the execution of job i on machine j.

Although for more than three machines optimal solutions may be non-permutation schedules, a common restriction in flowshop scheduling research is to consider only permutation schedules, that is, schedules where the job order on all machines in the same. Here, we follow this tradition. In addition, we adopt also the usual assumptions taken in many permutation flowshop scheduling problems such as job availability at zero and that operations may not be interrupted. A solution to Π_{PFSP} is, hence, a permutation π specifying the order in which jobs are to be processed. The completion times of all jobs on all machines are defined as

$$\begin{aligned} C_{\pi_0,j} &= 0, \quad j = 1, ..., m, & C_{\pi_i,0} &= 0, \quad i = 1, ..., n, \\ C_{\pi_i,j} &= \max\{C_{\pi_{i-1},j}, C_{\pi_i,j-1}\}, & i &= 1, ..., n, \quad j = 1, ..., m \end{aligned} \tag{2.12}$$

where π_i is the job at position i in the permutation, π_0 is a dummy job, and machine 0 is a dummy machine. Although many variants of the PFSP have been proposed in the literature, in this thesis we

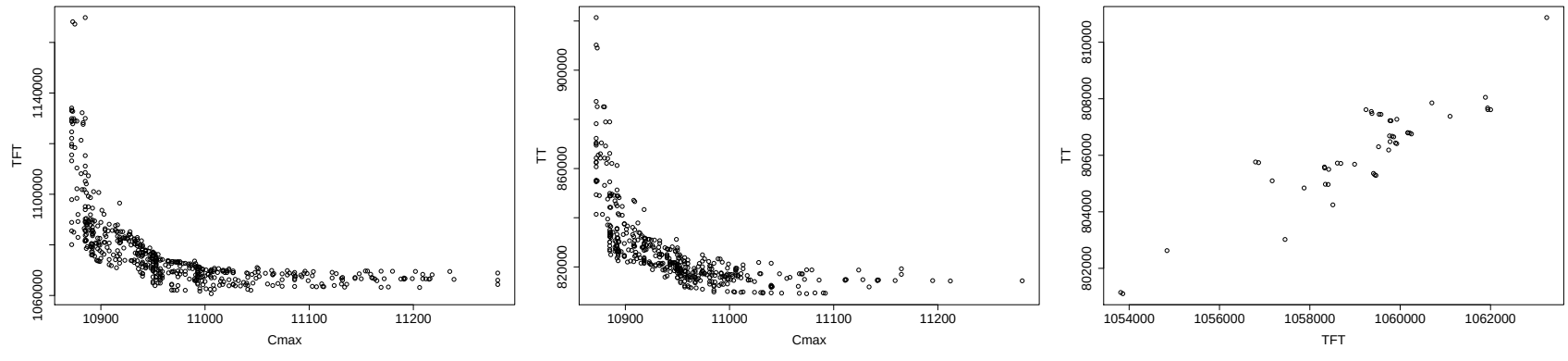


Figure 2.3: Approximation fronts retrieved by a state-of-the-art optimizer [69] for the MO-PFSP on different bi-objective variants depicting the correlation between the objectives.

consider only the original variant described above. Moreover, we restrict our review to the analysis of the most common objectives used in the literature, as follows:

- **Makespan** ($\mathsf{C_{max}}$): minimizing the completion time of the last job on the last machine, i.e., $C_{\pi_n,m}$. Formally,

$$\mathsf{C_{max}}(\pi) = C_{\pi_n,m} \tag{2.13}$$

- **Total flow time** (TFT): minimizing the sum of the completion times $C_{\pi_i,m}$ of each job $i = 1, 2, \ldots, n$ on the last machine. Formally

$$\mathsf{TFT}(\pi) = \sum_{i=1}^{n} C_{\pi_i,m} \tag{2.14}$$

- **Total tardiness** (TT): minimizing the difference between the completion times $C_{\pi_i,m}$ of all jobs in the last machine and their due dates d_{π_i}, which need to be provided. More precisely,

$$\mathsf{TT}(\pi) = \sum_{i=1}^{n} \max\{C_{\pi_i,m} - d_{\pi_i}, 0\} \tag{2.15}$$

- **Weighted tardiness** (WT): a more general version of the total tardiness objective, where weights w_i assessing the importance of jobs are provided. Formally,

$$\mathsf{WT}(\pi) = \sum_{i=1}^{n} w_{\pi_i} \cdot T_{\pi_i} \qquad T_{\pi_i} = \max\{C_{\pi_i,m} - d_{\pi_i}, 0\} \tag{2.16}$$

In this thesis, we consider a three-objective variant of the PFSP with the tuple of objectives ($\mathsf{C_{max}}$,TFT,TT), as well as the bi-objective variants comprising their pairwise combinations, namely ($\mathsf{C_{max}}$,TFT), ($\mathsf{C_{max}}$,TT), and (TFT,TT). Since the original PFSP problems are NP-hard, Pareto fronts are not available. Furthermore, we illustrate the shapes of high-quality approximation fronts for each of the bi-objective variants in Figure 2.3. In particular, each plot shows the aggregation of 25 approximation fronts returned by a state-of-the-art optimizer for the MO-PFSP when ran with different seeds. The roughly convex shapes of the fronts found for the $\text{PFSP}_{\mathsf{C_{max}}\text{-}\mathsf{TFT}}$ (left) and for the $\text{PFSP}_{\mathsf{C_{max}}\text{-}\mathsf{TT}}$ (center) contrast with the shape for the $\text{PFSP}_{\mathsf{TFT}\text{-}\mathsf{TT}}$ (right). In fact, this latter problem presents a particular characteristic of increasingly correlated objectives as the number of jobs grows [68].

Algorithm 1 Constructive metaheuristics

```
repeat
    Construct(pop)
    Learn(pop)
    s* ← ExtractBest(pop)
until termination condition is met
```

Output: $s^\star$

2.2 Multi-objective metaheuristics

Except for a particular family of problems for which polynomial algorithms have been devised, the complexity of MOPs made it impossible to use the *a posteriori* approach in the early years of MCDM. However, while the *a priori* approach reduces the difficulty of MOPs, any changes to those preferences may require re-executing the optimizer from scratch, which can be a computationally costly task depending on the given MOP. A surrogate approach used at early MCDM days was to employ multiple runs of the optimizer using different sets of preferences, providing the DM with a set of solutions that could be readily available in the case of preference changes. The advent of population-based metaheuristic algorithms (*metaheuristics*, for short) later enabled the adoption of the *a posteriori* approach from a single algorithmic run. Since defining preferences was never a straightforward task, efforts in applying metaheuristics to MOPs arose throughout the literature of operations research. Promising results have shown that it is indeed possible to devise high-performing algorithms able to retrieve large numbers of solutions at a small computational cost. By now, a number of such proposals can be identified forming a rich literature of *multi-objective metaheuristics* [50, 157, 199].

2.2.1 General overview of metaheuristics

Metaheuristics have been proposed as problem-independent heuristic algorithmic frameworks. On a higher abstraction level, metaheuristics work by constructing and/or refining solutions for a given optimization problem, in an iterative process that is held until a given termination criterion has been reached. At a lower level, the methods used either for constructing or for refining solutions make use of problem-specific knowledge, ensuring that the search is biased towards promising regions of the objective space. The flexibility of metaheuristic algorithms coupled with the good performance they have shown for many applications to important optimization problems has led to a rich literature on the subject [65, 74, 90, 92–94, 103, 106, 109, 127, 131, 134, 197, 213]. Metaheuristics have initially been proposed for single-objective optimization and later adapted for problems with multiple objectives. For this reason, we start our review by the original, single-objective versions of metaheuristics. However, a detailed description of the most relevant approaches observed in the literature would not fit this section, such is the extent of this field. By contrast, we choose to explain metaheuristics by focusing on the most relevant high-level features that characterize them.

To this date, many taxonomies for classifying metaheuristics have been proposed, but so far this task remains complex. The most common structural feature distinguishing metaheuristics is the way solution quality is improved, i.e., how the algorithm learns to identify the most promising regions of the search space. In common, metaheuristics create solutions by means of problem-specific heuristic procedures during each iteration. However, while some construct solutions from scratch, others generate solutions by modifying existing ones.

Algorithm 2 Population-based refinement metaheuristics

```
pop ← InitialSolutions()
repeat
    Refine(pop)
    s* ← ExtractBest(pop)
until termination condition is met
```

Output: $s^\star$

Algorithm 3 Local search metaheuristics

```
s ← InitialSolution()
repeat
    Perturb(s)
    LocalSearch(s)
until termination condition is met
```

Output: s

Constructive metaheuristics

Algorithms that stochastically build solutions from scratch at each iteration are called *constructive*. To increase the probability of generating high-quality solutions, metaheuristics such as *ant colony optimization* (ACO, [65]) and *estimation of distribution algorithms* (EDA, [144]) learn from the correlations between solution components and solution quality. In ACO, learning is modeled by means of positive and negative feedback. Analogously, EDAs iteratively refine the statistical distribution modeling of solution component correlations to improve the quality of sampled solutions. A high-level pseudocode for constructive metaheuristics is given in Algorithm 1. In particular, at each iteration an algorithm may keep either a single solution or a population of solutions, meant to increase the variability of the constructive procedure.

Refinement metaheuristics

Algorithms that start from a (set of) solution(s) and iteratively try to improve them are called *refinement metaheuristics*. Two main classes of metaheuristics can be identified on this group. The first maintain a population of solutions and produces novel solutions from their interactions. The other considers a single candidate solution and refines it by the systematic application of perturbations, i.e., modifications to a solution done in a heuristic way.

Population-based metaheuristics. The intelligence in population-based metaheuristics comes from (i) the set of potential solutions they keep and (ii) a heuristic way to use their interaction in an improving way. *Evolutionary algorithms* (EAs, [106, 197, 213]), for instance, use problem-specific variation operators such as crossover and mutation to modify existing solutions in hopes of improving them. Similarly to crossover operators from EAs, *scatter search* [94] algorithms reason that combining solution components from multiple solutions can lead to novel improved solutions, and do so by applying the *path-relinking* [94] operator. Another example of population-based metaheuristic is *particle swam optimization* (PSO, [74]), where solutions are modeled as particles that "fly" through the search space, modifying their path in function of the other particles from the population. Algorithm 2 is a high-level pseudocode for population-based metaheuristics.

Local search metaheuristics. These metaheuristics are heavily based on the use of local search procedures, and Algorithm 3 presents a high-level pseudocode for these algorithms. In more detail, given a perturbation pattern (*neighborhood operator*) σ and a candidate solution s, a local search

metaheuristic	approach	population based	local search	inspiration
evolutionary algorithms [106, 197, 213]	refinement	yes	optional	Darwinian evolution
ant colony optimization [65]	constructive	yes	optional	ant foraging behavior
particle swarm optimization [74]	refinement	yes	optional	bird flocks
artificial bee colony [131]	refinement/LS	yes	native	bee colonies
simulated annealing [134]	local search	no	native	metallurgy
tabu search [92, 93]	local search	no	native	gaming
GRASP [78]	both	no	native	none
iterated greedy [127]	constructive/LS	no	native	none
variable neighborhood search [103]	local search	no	native	none
scatter search [94]	refinement/LS	yes	native	none

Table 2.4: Non-exhaustive descriptive list of metaheuristics. LS stands for local search.

procedure starts by a systematic evaluation of the solutions s' that can be produced by applying σ to different solution component subsets of s (its σ-*neighborhood*). A selected improving neighbor replaces s, and the local search procedure is applied iteratively until a σ-local optimum solution is found. Local search metaheuristics are algorithms that use local search procedures as underlying search engine, but provide intelligent approaches to escape local optima. *Tabu search* (TS, [93]), for instance, allows the local search procedure to accept a non-improving σ-neighbor of s, but keeps a memory of previous moves to prevent cycling (the tabu list). *Variable neighborhood search* (VNS, [103]), by contrast, uses a set of different σ_i neighborhood operators of increasing perturbation size to allow an algorithm to escape a σ_i-local optimum through a σ_j neighbor, with $j > i$. Additionally, a local search metaheuristic can restart the local search procedure from a different point in the search space when no further refinement can be achieved, which is generally done by applying a randomized perturbation to the current candidate solution.

It is important to remark that, in practice, the separation between metaheuristics is not so blunt. For instance, local search procedures have been incorporated by many metaheuristics due to their effectiveness on particular application domains such as combinatorial optimization. In addition, some metaheuristics such as GRASP [78] and *iterated greedy* [127] combine a stochastic constructive approach with local search procedures at each iteration. Even more interesting is the hybridization of different metaheuristics, an ongoing research field with many relevant contributions [40].

A relevant sample of metaheuristics is summarized in Table 2.4 according to the features described in this section. We remark that the taxonomy we use in this review is merely structural. Other features could be considered that would reveal different categories of metaheuristics. For instance, some metaheuristic proposals are called *inspired* because the intelligent mechanisms they use derive from observations of real-world intelligence. A few examples of inspiration metaphors for metaheuristics are shown under the last column of Table 2.4. For further information on specific metaheuristics, the reader is referred to Hoos and Stützle [110] and to Gendreau and Potvin [91].

2.2.2 Adaptation to multi-objective optimization

It is often the case that the adaptations of metaheuristics to multi-objective optimization face critical design questions that eventually lead to a number of possible algorithmic structures. Some of these questions are general and apply to all metaheuristics, whereas others concern specific features of given metaheuristics.

General design challenges

We start our discussion by the more general issues that affect multi-objective metaheuristics. The first concerns how to maintain a search on multiple regions of the objective space at once. The most commonly used solutions are distinct for population-based and non-population-based metaheuristics.

Population-based metaheuristics: One naturally assumes this type of metaheuristics is intrinsically suited for the parallel optimization of MOPs, but the first studies on multi-objective evolutionary algorithms [204] showed that a population can suffer from: (i) *speciation*, when the population isolatedly converges to the extremes of the objective space, or; (ii) *stagnation*, when the population fails to get closer to the true Pareto front. Different solutions were proposed for specific metaheuristics to ensure a well-spread and effective search. In general, EAs use dominance-based preference relations [57]. By contrast, PSO algorithms have primarily relied on archives and their inherent concept of neighborhoods to ensure a diversified population [199].

Multiple weighted aggregations: Decomposing the original MOP into a set of subproblems via scalarizations is an approach used by many different multi-objective metaheuristics. In this case, the most important design question regards the methods for generating weight sets. In a uniform distribution, the number of weights grows significantly with the number of objectives. To a given extent, one can reduce the number of weights by increasing the spacing between the weights, but this approach leaves ever growing gaps in the search directions. Other approaches consider randomizing search directions, at the risk of not searching entire regions of the objective space. We remark that population-based metaheuristics can benefit from weighted aggregations, and in their case there is the extra design issue of how to promote cooperation between the different search directions. The solution is typically the use of neighborhoods w.r.t. search directions, based on the idea that the exploration of similar subproblems should consider solutions that are also similar. In practice, different metaheuristics use this concept in different ways [157, 186, 228].

Another example of a practical design question related to the adaptation of metaheuristics to solve MOPs concerns *external archives*. This algorithmic component is an auxiliary set used to store nondominated solutions found during the run of an algorithm. External archives are essential to non population-based algorithms because they optimize a single solution per iteration. To population-based algorithms, these archives are useful for two main reasons: (i) when the number of nondominated solutions of a given MOP is too large to be contained in a population, and; (ii) because of two theoretical properties that a bounded-size approximation front A such as a population may not present, namely *limit-stability* and *limit-optimality* [160], which are defined as follows.

Definition 13 (Limit-stability) *For any sequence consisting of points drawn indefinitely with a strictly positive probability from a finite set, there exists a point t such that $\forall\, v > t, A_t = A_v$. That is, the set A converges to a stable set in finite time.*

Definition 14 (Optimal approximation front of bounded size) *If $A \subseteq \mathbb{R}^M$ is a nondominated set, $|A| \leq N$, $N \in \mathbb{N}^+$, and $\nexists\, B \subseteq \mathbb{R}^M$, $|B| \leq N$, $B \lhd A$, then A is an optimal approximation front of bounded size N.*

Definition 15 (Limit-optimality) *For any sequence consisting of points drawn indefinitely with a strictly positive probability from a finite set, the set will converge to an optimal bounded front.*

Several questions derive from the use of external archives [160]: (i) which solutions are to be accepted in the archive; (ii) should the archive size be bounded, and; (iii) if bounded, how to select the solutions that are kept and that are removed when the archive size surpasses this limit.

Acceptance criterion: The traditional acceptance criteria used by external archives in the literature are related to Pareto dominance. When *nondominated* solutions are considered acceptable, the total number of solutions contained in the archive may rapidly increase. As a consequence, the performance of the algorithm may be impaired since the number of comparisons required for determining if a new solution will be inserted or not in the archive also grows. On the other hand, if only solutions that *dominate* solutions from the archive are accepted, such a performance issue is not a problem. However, this acceptance method opens the possibility of archive stagnation, i.e., not being able to add new solutions, even if they are not dominated by any solution in the current archive. The direct consequence of such situation is the presence of gaps in the output approximation fronts.

Bounded archives: A possible design alternative for dealing with the acceptance issues previously described is the use of *bounded archives*. Typically, bounded archives accept any nondominated solution but limit the archive size to prevent slowing the optimization process down. Several archive bounding techniques have been proposed so far, each presenting different strategies for determining which solutions are deleted from the archive every time a new solution is accepted. A theoretical and experimental analysis of this subject was performed by López-Ibáñez et al. [160]. The authors show that these bounding techniques can even lead the archive to a cycling situation, and that a careful analysis is required before selecting the proper archiving technique, with no particular archive being clearly consistently better than the remaining ones. We will resume this discussion when reviewing the MOEA literature, since most external archivers were proposed in this context.

2.2.3 Evolutionary algorithms

As discussed above, design questions are numerous when adapting a metaheuristic to deal with multi-objective optimization. Not surprisingly, the number of different MOEA proposals in the literature is large and an exhaustive review would go beyond the scope of this thesis. More importantly, in order to grasp the peculiarities of the different MOEA proposals, a precise definition of the original single-objective EAs is required.

Evolutionary algorithms are, in fact, a family of distinct algorithm subclasses [55]. In common, EAs share the characteristic of maintaining a population of individuals subject to an evolutionary process. This evolution is operationalized by means of (i) *variation* operators that produce new *offspring* individuals from *parent* individuals of the current population, and (ii) using selective pressure to maintain the population size constant throughout iterations. By combining these two procedures, EA designers expect their algorithms to converge towards high-quality solutions. A general pseudo-code of EAs can be seen in Algorithm 4. First, a population of solutions is generated, typically randomly or by means of problem-specific constructive heuristics. Then, the evolutionary process takes place and when a given termination criterion is met, such as a number of iterations, a number of solution evaluations, or a number of solutions generated, the best solution $s^\star$ of the population is returned. In particular, the Reduce procedure is used by some algorithms to select solutions that will survive on the next iteration. In others, the population size is kept constant by *online replacement*, meaning offspring solutions replace parent solution as soon as they are created.

Although all EAs share the same overall structure, they present significant differences which we detail next.

Genetic algorithms

The best known type of EAs, *genetic algorithms* (GAs) [106] are among the first proposals in the field of evolutionary computation. Using the biological terminology, GAs represent candidate solutions as evolving *chromosomes*. In its original proposal, instead of optimizing decision variables directly GAs

Algorithm 4 Evolutionary algorithms

```
pop ← GenerateInitialSolutions()
repeat
    Variation(pop)
    Reduce(pop)
    s* ← ExtractBest(pop)
until termination condition is met
```

Output: $s^\star$

generally made use of *indirect encoding*. Concretely, the *genotype* of the chromosome maps an actual solution to the problem – its *phenotype*. *Binary encoding*, for instance, maps the ranges of the decision variables to binary representation. Although indirect representation is still used nowadays on specific scenarios, *direct encoding* has become the most common encoding used by current GAs.

The GA chromosome population is evaluated at every generation (iteration). The fitness computation of chromosomes is an important component of GAs because every step of the evolutionary process tries to maximize the overall fitness of the population. More precisely, before applying variation operators, a GA first builds a *mating pool* of parent solutions. After the application of the operators, GAs perform a *population reduction* known as *environmental selection*. For both selection scenarios, the fitness of the individuals plays a central role as high-quality individuals are favored in hopes of improving the overall quality of the population. Concerning mating, many selection operators can be used in GAs, the most common being: (i) *roulette-wheel*, where the probability of selecting an individual is proportional to its fitness, and; (iii) *n-ary tournament*, where n individuals are randomly selected and pairwisely compared according to their fitness until only one is selected (like a champion in sports playoffs). Once the mating pool is built, genetic operators are applied to generate offspring solutions.

The standard variation operators used by GAs are crossover and mutation. Crossover operators require (at least) two parents and mix their information to generate offspring solutions. Mutation operators are unary operators that apply perturbations to chromosomes, and are complementary to crossover. Several operators of each kind have been proposed for the various domains of optimization, and later we will discuss the most used for the target problems we consider in this thesis. Once the offspring population has been created, GAs apply an environmental selection procedure to reduce the population size back to its standard value. The first proposal of GAs originally used *generational replacement*, i.e., maintaining only the individuals created in the current generation. Later, *elitism* was introduced, where algorithms allow the best solutions of the previous generation to persist in the new population. The number of elitist solutions to be preserved is a critical parameter of GAs, which needs to be defined on an application-basis. For further information on GAs, the reader is referred to Reeves [198].

Evolution strategies

One of the earliest proposals of metaheuristics, evolution strategies (ES) [197] are also the conceptually simplest approach among EAs. By contrast to GAs, ES always represent solutions through direct encoding. The set of decision variables is known as *object parameter set*, and the individuals as *object parameter vectors*. These individuals are subject to variation operators as in other evolutionary algorithms, but the mating selection is typically performed randomly. Similarly to GAs, once the offspring population is produced, an environmental selection strategy is employed. Formally, an ES algorithm is defined using the $(\mu/\rho \overset{+}{,} \lambda)$-ES notation, where μ stands for the population size, ρ stands for the number of parents used by the crossover operator, and λ stands for the number of offspring produced in one generation of the ES. The *comma* and *plus* refer to the environmental selection strategy. In a comma strategy, only the λ newly generated offspring comprise the selection pool. In other words, the new population will be exclusively formed by $\mu < \lambda$ offspring elements, equivalent to the generational replacement from GAs.

Conversely, plus strategies add the μ solutions from the parent population to the selection pool, thus allowing elitism in the algorithm. Originally, ES proposals took the form of $(1+1)$-ES algorithms, i.e., the population size set to one, and so the number of offspring. In addition, only mutation operators were used [197]. Later, *steady-state* ES were proposed, taking the form $(\mu/2+1)$-ES. In these algorithms, two parents are randomly selected and subject to crossover producing a single offspring[7], which is then subject to mutation. The newly generated offspring then replaces the worst individual of the population. Over the years, the $(\mu/\rho+\lambda)$-ES and $(\mu/\rho,\lambda)$-ES were proposed, thus giving birth to the $(\mu/\rho\,\dot{,}^{+}\lambda)$-ES notation [21].

From a high-level perspective, one notices the similarities between GA and ES. In fact, the separation between these two EAs has become ever blurrier, with algorithms being simply referred to as EAs. Nonetheless, there is one particular feature of ES that makes it unique with regard to GAs, namely the self-adjustment of its numerical parameters. In particular, some ES algorithms use co-evolution to simultaneously evolve the object parameter set and its numerical parameter values. The current most successful example of an evolution strategy that self-adapts some of its numerical parameters concerns the field of continuous optimization. CMA-ES (*covariance adaptive matrix evolutionary strategy*) [102] is a $(\mu/\mu,\lambda)$-ES algorithm that samples object parameter vectors using statical distributions regulated by a covariance matrix. At every iteration, the covariance matrix is adapted to learn from the newly produced individuals. Due to its popularity, many variants of this algorithm have been proposed, among which we highlight two: (i) a variant that is the current state-of-the-art for single-objective continuous optimization [11], and; (ii) a variant that has been proposed for multi-objective optimization, which we will later further detail [119]. For further reference on ES, the reader is referred to Beyer and Schwefel [21].

Differential evolution

Differential evolution (DE) is an EA approach proposed for the field of continuous optimization several decades after the first proposals of GA and ES [213]. The underlying concept of DE algorithms is calculating difference vectors between individuals to guide their movement. Similarly to ES, DE algorithms use direct encoding by means of parameter vectors, but in the literature of DE these individuals are also referred to as chromosomes. Traditionally, the initial population in DE algorithms is generated randomly. The evolutionary process makes use of a three-stage combined operator. First, a *target* vector is selected. Second, a *donor* vector is created by means of a differential mutation operation. Finally, a *trial* vector is produced by the crossover between the target and donor vectors. The trial vector created by this differential operation replaces the target vector in the following generation if it is not worse than it. The basic structure of DE algorithms is then defined as schemes DE/x/y/z, where x stands for the target vector selection method, y for the number of donor vectors used by each target vector, and z is the type of crossover used. Concerning mutation, five standard schemes have been proposed by Storn and Price [213], which we describe next.

- The first two DE schemes use a single donor, varying only as to the target selection mechanism: *DE/rand/1* (2.17) selects target vectors randomly, whereas *DE/best/1* (2.18) always uses the best individual as its target. Formally, given randomly generated mutually exclusive indices r_j^i, a crossover operator represented by the $+$ symbol, and a scale factor $F \in [0,1]$, the i-th trial vector produced at generation G, $\vec{V}_{i,G}$, is defined as:

$$\vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}) \tag{2.17}$$

[7] In case two offspring are produced by the crossover operator, one is selected at random and discarded before being evaluated.

$$\vec{V}_{i,G} = \vec{X}_{best,G} + F \cdot (\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}) \tag{2.18}$$

- Other two standard schemes are obtained from the first two by the addition of a second donor vector, thus giving rise to schemes *DE/rand/2* (2.19) and *DE/best/2* (2.20):

$$\vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}) + F \cdot (\vec{X}_{r_4^i,G} - \vec{X}_{r_5^i,G}) \tag{2.19}$$

$$\vec{V}_{i,G} = \vec{X}_{best,G} + F \cdot (\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}) + F \cdot (\vec{X}_{r_3^i,G} - \vec{X}_{r_4^i,G}) \tag{2.20}$$

- The last standard scheme, *DE/target-to-best/2* (2.21), uses a random target individual which is recombined with two donors. The first is the output of the differential mutation operation between itself and the best vector of the population. The second, the output of the differential mutation between two randomly selected vectors.

$$\vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot (\vec{X}_{r_{best}^i,G} - \vec{X}_{r_1^i,G}) + F \cdot (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}) \tag{2.21}$$

Concerning crossover operators, the traditional proposals in the DE literature are the *exponential* and the *binomial* operators. Exponential crossover uses two randomly generated numbers: a starting position r and the number of elements L that will come from the donor vector. Given a problem with n decision variables, a donor vector $\vec{V}$ and a target vector $\vec{X}$, the trial vector $\vec{U}$ is defined as $\vec{U} = (x_1, ..., x_{r-1}, v_r, ..., v_{r+L}, x_{r+L+1}, ..., x_n)$, with all indices subject to *modulus* n. By contrast, binomial crossover randomly selects for each position of the parameter vector if the information will come from the target or from the donor vectors. For further information on differential evolution, the reader is referred to Das and Suganthan [54].

2.2.4 MOEA design evolution

The historical evolution of MOEA design to deal with the adaptation issues from the generalization to multi-objective optimization can be split into three main epochs.

Initial years

In the first years, MOEA researchers were mainly concerned with modeling MO components to allow EAs to search and retrieve multiple Pareto optimal solutions in a single run, to overcome the speciation issue observed in the first ever MOEA, **VEGA** [204]. The most instrumental ideas from this period were the proposal of tackling convergence and diversity through different preference metrics to be used in mating selection. The first suggestion of such a proposal described the ideas for nondominated sorting (convergence) and fitness sharing (diversity) [96], although very briefly. A few years later, three of the most relevant algorithms from this epoch were proposed in a very short span of time, namely MOGA [80], NSGA [212], and NPGA [111]. In fact, these algorithms are responsible for stirring the studies on MOEAs, as they were the first to successfully demonstrate their potential for simultaneously optimizing multiple solutions from different regions of the objective space. In general, all these algorithms used fitness sharing for diversity. For brevity, we only review the nondominated sorting approaches implemented by MOGA and NSGA, as follows.

Nondominated sorting

- **MOGA** [80] evaluates its population based on their *dominance rankings*. More precisely, the dominance ranking of a solution equals one plus the number of other solutions from the population

that dominate it. Solutions with lower rankings are favored, and solutions with unitary ranking comprise a nondominated subset within the population.

- **NSGA** [212] authors argued that dominance rankings were a misleading approach, since solutions with the same ranking could dominate each other in the case of rankings greater than one. As an alternative, *dominance depth* divides the population into nondominated fronts, ensuring that (i) the members of the first front are not dominated by any individual of the population; (ii) the members of front i are only dominated by members of fronts $j < i$, and; (iii) members of a given front are nondominated w.r.t. each other.

Fitness sharing: the idea of fitness sharing is to promote convergence within localized regions of the objective space, while maintaining several niches spread along the objective space. Although algorithms present slight variations as to the implementation of this metric, the general idea is to analyze the density of the niche with radius σ_{share} centered in the solution under analysis. This way, if the algorithm must select between two solutions (either for mating or replacement), the one with fewer solutions in its niche should be favored, potentially preventing the search from missing a region of the objective space. In practice, this algorithmic component was used in different ways by the proposed MOEAs, with some computing niches in the decision space (NSGA) and others in the objective space (NPGA and MOGA). Finally, we remark that the most significant drawback with this approach is setting σ_{share} appropriately, since this parameter may be affected by factors such as problem dimensionality, objective ranges, and stage of the run.

Maturing years

With the promising results from the first epoch, the second epoch in MOEA design focused on the development of effective methods to explore the potential demonstrated by the first MOEAs. Important concepts proposed in this epoch include MO elitism [60, 137, 233, 234], external archivers [137, 147, 160], the integration of quality indicators into the search [19, 232], the decomposition of the original MOP in subproblems for parallel optimization [150, 228], and the use of EAs other than genetic algorithms [19, 119, 137, 143, 201, 219].

MO elitism: Although the first MOEA proposals demonstrated that it was possible to optimize multiple solutions at once, an experimental analysis conducted by the authors of SPEA [233] revealed that *elitism* was critical for improving the convergence ability of MOEAs. In addition, authors proposed a new diversity approach and a refined version of dominance rankings, namely *dominance strength*, as follows.

- When computing dominance rankings, **SPEA** takes into account the number of solutions each individual dominates, i.e., its *dominance strength*. This way, the refined ranking of a solution equals one plus the strength sum of the individuals that dominate it.
- To preserve diversity, **SPEA** clusters solutions and selects one individual to represent each cluster. In particular, *clustering* is done based on the average distance between solutions and the number of clusters equals the number of surviving population members, and hence no numerical parameters are required.

In particular, it is important to remark that SPEA used dominance strength solely for mating selection, whereas clustering was used only for environmental selection. A series of elitist MOEAs were proposed in the following years, among which we remark PAES [137], NSGA-II [60], and a refined version of SPEA, namely SPEA2 [234]. We remark that the elitism proposed for PAES is directly related to its external archive, and hence for now we restrict our presentation to NSGA-II and SPEA2.

- To promote elitism, NSGA-II [60] uses dominance depth and *crowding diversity* both for mating and environmental selection, in this order of priority. In particular, diversity based on crowding is a parameter-less approach where individuals are evaluated based on their distances to their neighbors. The notion of solution neighborhood in this proposal is defined objective-wise. More precisely, solutions are sorted for each objective and consecutive solutions are considered neighbors. The crowding distance of a solution is then the average distance between the pairs of neighbors of a solution from each objective. The main benefit of this approach is that no numerical parameter needs to be set, and that for two-objective problems the notion of neighborhood is precise. By contrast, at least three drawbacks deserve to be mentioned. First, selecting neighboring solutions based on objective-wise sorting is an approach that does not scale well with the number of objectives. Second, since extreme solutions have a single neighbor, their values are artificially enlarged to ensure they are not discarded. However, an extreme solution that is only slightly improving for one objective but is particularly poor for the others will still be favored in selection because of this artificial crowding. Finally, the crowding distance of a solution for a given objective i does not vary with modifications to its i-th objective as long as it remains within the bounding box delimited by the i-th objective values from its neighbors.

- Similarly to NSGA-II, **SPEA2** [234] uses convergence and diversity metrics in this priority order both for mating and environmental selection. However, SPEA2 is the first algorithm to consider diversity metrics with different behaviors for each of these procedures. In particular, although both metrics are based on *k-th nearest neighbor* computation, mating selection uses a pre-defined formula for computing k as a function of the population size. By contrast, the environmental selection procedure considers $k = 1$, and increasing k in the case of ties.

The proposal of elitist MOEA approaches can be clustered according to the number of times fitness metrics are computed for the environmental selection. *One-shot removal* [13] computes metrics once and discards the worst solutions altogether. *Sequential* (or *iterative* [13]) *removal* [160] discards one solution at a time and recomputes metrics before the next solution is discarded. Although the information provided by the sequential removal policy is more accurate, this policy is computationally more demanding, which may compromise its performance in time-constrained scenarios. Among the approaches we discussed, NSGA-II adopts the one-shot removal [60], whereas SPEA2 uses sequential removal [234].

External archives: Although elitism improved MOEA convergence properties, Laumanns et al. [147] demonstrated that approximating a possibly limitless Pareto front using a bounded population was a task that the then existing MOEAs were unable to accomplish. In particular, diversity-preserving selection approaches as the ones used by NSGA-II and SPEA2 lack both limit-stability and limit-optimality. By contrast, the proposal of an auxiliary, external solution set (archive) to ensure convergence was firstly proposed for PAES [137] and later novel stand-alone archiving proposals followed [135, 145, 147, 160]. Below we summarize their characteristics following the theoretical and empirical analysis of López-Ibáñez et al. [160]. In particular, besides the concepts of limit-stability and limit-optimality, the authors also considered the concepts of Pareto-monotonicity and of $\lhd$-monotonicity, respectively meaning that (i) a vector present in an archiver cannot be dominated by a vector previously discarded, and; (ii) there does not exist a previous state of the archiver A_t such that $A_t \lhd A_c$ (A_t is better than A_c), A_c being the current state of the archiver.

- **The adaptive grid archiver (AGA)** [137]: this archiver divides the objective space into a hypergrid with identical volume grid cells. The size of the grid cell is defined adaptively as a function of the objective ranges of the extreme solutions found so far in the search and of

a numerical parameter l that establishes the number of divisions per objective. Since only a single solution per grid cell is allowed, this archiver ensures diversity. However, similarly to the diversity-preserving approach from NSGA-II and SPEA2, this archiver originally proposed for **PAES** [137] is unable to ensure convergence, being neither monotone, $\lhd$-monotone, limit-stable, nor limit-optimal.

- **The adaptive ϵ-Pareto archiver** [147]: this archiver is very similar to the AGA, but defines the size of its grid cells as a function of an ϵ value. In particular, the objective-wise size of each grid cell increases exponentially as a function of ϵ, meaning that near the Pareto front the granularity of this archiver is much more refined than for distant regions. Similarly to the AGA, this archive is diversity-preserving, but neither monotone, $\lhd$-monotone, limit-stable, nor limit-optimal.
- **The hypervolume archiver ($\mathbf{AA}_s$)** [135, 139]: this archiver accepts a novel nondominated vector by discarding the *least hypervolume contributor* of the current archive. More precisely, the least hypervolume contributor of an approximation front is the vector $\mathbf{v} \in A$ that least contributes to the hypervolume of A. With this approach, the AA_s ensures that I_H of the archive never decreases, inheriting its properties, i.e., this archiver is both diversity-preserving and $\lhd$-monotone. In addition, since a maximum I_H exists for any given MOP, this archiver is also both limit-stable and limit-optimal. The only property not displayed by this archiver is monotonicity.
- **The multi-level grid archiver (MGA)** [145]: this grid-based archiver uses a surrogate dominance concept to ensure convergence, called *box dominance*. More precisely, the *box index vector* of a solution comprises its box indices $r^{(i)}$ for each objective $i = 1, 2, \ldots, M$, where $r^{(i)} = \lfloor v_i \cdot 2^{-i} \rfloor$. The box dominance relations ensure that a solution is only accepted if it belongs to a grid cell that is closer to the true Pareto front, i.e., if it *box-dominates* another solution from the archiver. In particular, this archiver has been shown to be diversity-preserving, $\lhd$-monotonic, limit-stable, an limit-optimal. However, similarly to the AA_s it is not Pareto-monotonic.

As discussed above, the analysis conducted by López-Ibáñez et al. [160] demonstrated that only the AA_s and the MGA are simultaneously limit-stable and limit-optimal. However, while both archivers are shown to be monotone w.r.t. the better ($\lhd$) relation, it is possible that a objective vector present at the end of a run be dominated by an objective vector discarded along the search. It is important to remark that the algorithms and test cases used by the authors were both artificial, but one overall concludes the importance of external archives and that this research field requires further targeted efforts.

Indicator-driven search: With the proposal of Pareto-compliant quality indicators, some authors envisioned using these indicators within MOEAs to direct their search. In particular, the convergence pressure provided by some quality indicators is not directly influenced by the number of objectives. In addition, some indicators also account for diversity, keeping the population well-spread along the objective space. The drawback of indicator-based approaches is the computational complexity of some quality indicators such as the hypervolume, which is exponential in the number of objectives in the worst case [20]. The firstly proposed indicator-based MOEAs were **IBEA** [232] and **SMS** [19], which we next detail.

- **IBEA** [232] uses a refined version of dominance rankings based on *binary quality indicators*. Concretely, IBEA computes the pairwise values of a given binary quality indicator for the whole population. Then, the preference of each individual is given by the aggregation of its binary values w.r.t. to the rest of the population. Since only Pareto-compliant binary

indicators are used, the solutions that dominate a given solution s contribute negatively to the score of s, whereas the solutions that s dominates contribute positively to its score. By favoring solutions with larger scores, IBEA is able to promote convergence and diversity. In addition, given that pairwise comparisons between solutions using binary indicators are cheap to compute, IBEA has very little computational overhead when compared to other indicator-based MOEAs. The most used binary indicators for IBEA are the additive epsilon indicator ($I_{\epsilon+}$) and the hypervolume difference (I_H^-) [235]. In addition, IBEA is an elitist MOEA that uses sequential removal for the environmental selection.

- **SMS** [19] uses a set of metrics for its environmental replacement, the most distinguishing one being the *exclusive hypervolume contribution* (I_H^1). More precisely, SMS was proposed as a hypervolume-based MOEA, but the I_H^1 is a computationally demanding indicator. For this reason, SMS-EMOA combines the I_H^1 with two other convergence metrics to reduce the number of times this indicator is employed. Concretely, the primary convergence metric used by SMS is dominance depth. In case multiple nondominated fronts are identified in the population, dominance ranking is used to distinguish dominated and nondominated solutions. This way, the exclusive hypervolume contribution is only used when the population is comprised solely of nondominated solutions. Furthermore, to reduce the overall computational complexity of the search, SMS uses steady-state replacement, as we will later further describe. By generating a single offspring per generation, the environmental selection procedure of SMS reduces to the computation of the *least hypervolume contributor* as in AA_s, and recent available implementations have been able to considerably reduce this overhead for larger numbers of objectives [183, 223].

MOP decomposition: An alternative research direction already during the first epoch of MOEA design was to decompose the original MOP into simpler subproblems [100, 128], typically by means of scalarizations. This paradigm, however, did not become mainstream for many years for two main reasons. First, in the initial years MOEA researchers were more interested on developing approaches that could deal with MOPs from a dominance perspective, since VEGA [204] had demonstrated that a population can suffer from speciation. Second, the performance of the decomposition-based MOEA considered in the experimental comparison conducted by Zitzler and Thiele [233], **HLGA** [100], was worse than the performance of the MOEAs that used nondominated sorting and fitness sharing. In particular, the most commonly believed explanations for the poor performance of HLGA were the theoretical issues previously discussed for scalarization-based approaches. A few other decomposition-based MOEAs were proposed for continuous [124, 125] and combinatorial optimization [122, 128], but it was not until the proposal of the **MOEA/D** algorithms [150, 228] that the interest from the MOEA community on decomposition was stirred. Next, we detail the two most relevant MOEA/D algorithms.

- The original **MOEA/D** algorithm uses a set of uniformly distributed weight vectors to simultaneously optimize different subproblems. As aggregation method, the authors of MOEA/D consider all the options we previously discussed in Section 2.1.2, namely linear weighted sum, Tchebycheff utility, and penalty-based boundary intersection (PBI). To promote cooperation around diversified regions of the objective space, the mating selection of MOEA/D considers local neighborhoods for each subproblem. Concretely, these local neighborhoods are defined based on the Euclidean distance between the weight vectors used to define each subproblem. Variation operators are applied as in a regular EA, and the newly generated solution is compared to the best solutions found so far for each subproblem. In particular, MOEA/D implements a strong version of elitism via a cloning mechanism that allows the same solution to become the new best solution for multiple subproblems.

- **MOEA/D-DRA** was proposed a few years later, with a number of modified algorithm components w.r.t. the original MOEA/D. The most significant novelty of MOEA/D-DRA concerns the *dynamic resource allocation* (DRA) procedure, an online adaptive strategy proposed to reduce the computational budget wasted on non-promising search directions. Initially, each of the μ weight vectors is given the same utility value. At each iteration, MOEA/D-DRA selects a subset μ/ν to explore via tournament selection based on the utility values of the weights. Once the weights have been selected, variation is applied to each search direction. In this version, however, a parameter δ regulates whether mating selection will consider the local neighborhood or the whole population. Finally, a subset of the selection set (local neighborhood or population) is used to update the search reference point for the current weight. The size of this subset is regulated by an additional parameter ϕ. At intermediary stages of the search, the utility values of the weights are recomputed. Despite the large set of numerical parameters, MOEA/D-DRA gained popularity after winning the IEEE CEC 2009 competition on unconstrained multi-objective continuous optimization [150].

Alternative EAs: In general, MOEA authors have focused their efforts on devising effective components to deal with the multi-objective aspects of the search, while simply reusing the same EA and variation operator choices traditionally adopted in the literature. In particular, nearly all the MOEAs we reviewed so far were GAs with deterministic tournament selection, SBX crossover and polynomial mutation. The only MOEAs we have discussed so far that were based on different EAs are (i) SMS, a steady-state $(\mu + \lambda)$-ES that uses random mating selection but adopts the same variation operators used by the other MOEAs, and; (ii) MOEA/D-DRA, which uses DE variation operators.

Overall, the literature exploring different EAs is limited, and the role variation operators of the underlying EAs have on performance is still rather poorly understood for both multi- and many-objective optimization. The most relevant MOEA proposals based on different EAs or using different variation operators during this epoch were **DEMO** [143, 201][8] and **MO-CMA-ES** [119], which we discuss below.

- **DEMO** stands for differential evolution (DE) for multi-objective optimization. Following an initial research on the adaptation of DE for MOPs [1, 166], the authors of DEMO proposed an algorithm that combines the online replacement from single-objective DE with nondominance sorting and diversity preservation mechanisms from NSGA-II. For mating, DEMO uses the DE/rand/1/bin strategy, and no other strategy has been reported by the authors. The positive results achieved by DEMO later motivated its authors to test other existing environmental selection metrics, resulting in different DEMO versions [219]. In particular, authors demonstrated that DE could improve the overall performance of existing MOEAs such as SPEA2 and IBEA.

- **MO-CMA-ES** is an adaptation of the successful *covariance matrix adaptation evolution strategy* (CMA-ES) used in continuous single-objective optimization [101]. The basic MO-CMA-ES algorithm [119] works as a number of parallel executions of the CMA-ES algorithm starting from different initial solutions. For environmental selection, dominance depth and the exclusive hypervolume contribution (I_H^1) are used in this priority order. Variants of this algorithm were proposed later, allowing different quality indicators, selection for survival, and learning strategies for the underlying CMA-ES algorithm [221].

[8] DEMO [201] and GDE3 [143] are two independently proposed MOEAs that are identical in nearly all of its algorithmic components. We choose to explain DEMO since it was proposed shortly before GDE3.

Contemporary years

The third (and current) epoch in MOEA design is characterized by an increased interest on so called *many-objective optimization* problems (MaOPs) [79, 195]. Originally, MaOPs did not draw much attention of MOEA researchers for two main reasons. First, researchers deemed that algorithms that performed well on problems with two and three objectives would naturally scale to deal with any number of objectives. Second, the practical applicability of such models was seen as reduced, since an *a posteriori* decision making process would have too many incomparable solutions to deal with. Both these premises have recently been re-evaluated. From an application perspective, Fleming et al. [79] demonstrated that existing engineering problems could be remodeled by considering constraints as objective functions, coining the term many-objective optimization. From the perspective of algorithm performance, two works were instrumental. The first one [132] demonstrated that MOEAs (in particular NSGA-II, SPEA2, and PESA [52]) faced scalability issues w.r.t convergence, diversity, and/or running time when solving DTLZ problems with up to ten objectives. In fact, even doubling the already extremely large number of function evaluations given to MOEAs was not enough for them to reach the actual Pareto fronts. The second work [195] demonstrated how existing MOEA algorithmic components were unable to scale in face of *dominance resistance*. More precisely, in the presence of a large number of conflicting objectives, the proportion of the feasible space that is incomparable becomes too large for traditional MOEAs to handle. Already at an early execution stage, the entire MOEA population becomes nondominated and spread, making it difficult for MOEAs to progress through the evolutionary operators commonly used.

Altogether, these factors increased the application demand and reinforced the need for improving existing MOEA approaches. To overcome these limitations, several approaches have been considered, as identified by Li et al. [149]. Below we detail the most relevant approaches grouped in two main categories, as follows.

Dimensionality analysis. The first category considers analytical approaches to easy the task of dealing with multiple objectives. Concretely, these dimensionality reduction approaches look for objectives with high correlation that could be further represented by a single objective [203], or use feature-based selection to extract only the objectives that are most significant [161]. In general, this kind of analysis is conducted offline, meaning it does not incur in computational overhead for algorithms and can possibly be reused by different optimizers. Nonetheless, a few online approaches have also been proposed [149].

Algorithmic approaches. The second category comprises new algorithmic concepts, such as using reference-based search [59, 148, 192], or revisiting existing ideas, such as quality indicators [62, 174, 202, 222], and alternative dominance definitions [4, 98, 99, 145, 181, 227]. We further discuss each of these approaches:

- **Reference-based search**: these approaches focus specifically on the diversity challenge posed by the large number of objectives. In particular, to ensure that the search is conducted in a balanced way, reference elements are used as attraction sources. In some proposals, these elements are actual solutions, like an archive focused on diversity to ensure the spread of the search [148, 192]. In other, abstract elements such as reference lines [59] or target vectors [124, 126] are used. We remark that reference solutions can also be used to drive an algorithm towards convergence, as MOEA/D does with the best elements for each subproblem [228].
- **Quality indicators**: given the contrast between the desirable properties and the computational complexity of the hypervolume, some algorithms have considered: (i) novel indicators such as generational distance indicators [62, 174, 175], Δ_p [202], and α [43, 222], or; (ii) a refined version of the hypervolume coupled with less computationally costly methods for its approximation [13].

- **Alternative dominance definitions**: since Pareto dominance is known to become less restrictive as the number of objectives increases, these approaches either (i) rely on existing dominance alternatives to Pareto dominance such as the previously discussed ϵ-dominance [98, 99], grid dominance [227], box-dominance [145], and Lorenz dominance [181], or; (ii) propose novel relations such as space partitioning [4].

It is important to remark that many of the components discussed above are not conceptually novel, having been revisited from their original proposals. We then select two algorithms to further detail them, namely the indicator-based HypE [13] and the reference-based NSGA-III.

HypE [13] is an indicator-based MOEA that searches the solution space guided by the *shared hypervolume contribution* (I_H^h) of the individuals. This quality indicator measures the volume of the subspace an individual exclusively dominates, plus shares of the volumes that it jointly dominates with up to other h individuals of the population. For mating selection, HypE uses the I_H^h indicator with $h = \mu$, the population size. For environmental selection, HypE uses dominance depth as a primary convergence metric, and uses the I_H^h contribution to distinguish between solutions from the first non-fitting nondominated front. In addition, the elitist environmental selection from HypE uses sequential removal. When recomputing the I_H^h contributions, h equals the number of solutions from the non-fitting front that still have to be discarded. Due to its computational overhead, HypE uses a Monte Carlo simulation to estimate I_H^h for problems with more than three objectives.

NSGA-III [59] is the most recent algorithm in the NSGA series. Similarly to NSGA-II, the environmental selection uses dominance depth as primary fitness metric. However, the crowding distance operator from NSGA-II is replaced in NSGA-III by a diversity mechanism that uses weight-based reference lines. More precisely, each solution is initially associated with its closest reference line. The algorithm then selects a representative from the population for each line, ensuring that the population remains spread over the objective space. More importantly, NSGA-III proposes a two layer weight generation methodology to reduce the total amount of weights required by large numbers of objectives. The elitism procedure of NSGA-III follows a one-shot approach, since solutions are associated to reference lines only before the solutions to be discarded are selected.

2.2.5 MOEA frameworks

Given the large number of MOEAs proposed in the literature, several algorithmic framework initiatives have prompted to encourage practitioners wanting to use MOEAs on their application domains. In particular, these frameworks focus on reusability from a problem perspective, meaning MOEAs are provided as classes and an end user needs only to implement (or select an available implementation) of the solution encoding and evolutionary operators that suit his needs best. Below we detail the most relevant MOEA frameworks from the literature, since a better understanding of these frameworks is instrumental to grasp the originality of the framework we propose in this thesis. We remark that we only include compilable frameworks in our discussion since frameworks implemented in interpreted language present a natural runtime handicap that could interfere in experimental assessments[9].

Shark [120] is a machine learning framework that implements a few MOEAs such as SMS-EMOA and MO-CMA-ES. Since the scope of Shark is broad, it is not surprising that a few MOEAs are provided by default, although implementing a novel MOEA using this framework is not a challenging task. A few, classical evolutionary operators are provided, and so are a few MO

[9] We do not include PISA [39] in our review for a similar reason. More precisely, although compilable PISA is implemented as a state machine that uses file-based communication. This architecture greatly hinders its efficiency and makes it unpractical for experimental analyses.

components such as dominance depth and the hypervolume indicator, but for the latter we remark that the computational complexity of the implementation provided is far from efficient.

PaGMO [38] is a framework for parallel global optimization that provides a few MOEA implementations such as NSGA-II, SPEA2, and SMS-EMOA. In particular, PaGMO implements one of most efficient hypervolume (contribution) algorithms. Since the main focus of PaGMO is parallel optimization, implementing a novel MOEA is not as easy as reusing parallel computation models, for instance, although similarly to Shark it cannot be considered a challenge per se. By contrast, PaGMO offers implementations of other multi-objective metaheuristics such as particle swarm optimization (PSO) and artificial bee colony (ABC), which can prove useful in given circumstances such as hybridization studies.

ParadisEO-MOEO [153] builds upon the ParadisEO framework, adding the functionalities required for EAs to deal with multi-objective optimization. Besides being problem-flexible, ParadisEO-MOEO is the only framework we identify in the literature that allows MOEA components to be reused in a flexible way. More precisely, practitioners are given the option of implementing MOEAs from scratch or as an instantiation of a template that takes algorithmic components as input. This way, implementing a novel MOEA can be made simpler since only the lacking algorithmic components need to be implemented. However, the major drawback with this framework is precisely that the template used for defining a MOEA offers little representativeness. In particular, this template is based on the early works where MOEAs were mostly based on fitness/diversity-specific components. In addition, it is not possible to use a single component with different behaviors according to the type of selection it is used for (mating or environmental), as often seen in the literature. Two facts further evidence the representativeness limitations we address. First, only a few MOEA implementations are provided as default instantiations of this template, since implementing other approaches would require considerable changes to the structure of the template. Second, the SPEA2 template instantiation provided requires a specific archiver implementation, although we show in this thesis that the truncation procedure proposed for archivers by SPEA2 is not in essence different from its mating selection preferences.

As it can be seen, frameworks are a relevant research effort observed in the literature, and have greatly helped disseminate the use of MOEAs by practitioners with little expertise in multi-objective metaheuristics. Nonetheless, it is important to remark that their main motivation is MOEA reusability rather than MOEA algorithmic component reusability, being fundamentally different from the approach we propose in this thesis.

2.3 Automated algorithm engineering

As shown in the previous section, the number of different MOEAs available for a practitioner to consider when dealing with a novel multi-objective application problem is beyond what practically can be considered. In addition, since these algorithms have numerical parameters, wrong conclusions can easily be drawn simply due to an algorithm configuration that is suboptimal for the experimental setup under consideration. More importantly, although our discussion considers MOEAs as case study, any given heuristic optimization technique with a set of different parameterized algorithmic proposals would also be a representative scenario for this problem. In this section, we discuss recent research that deals with the automated selection, configuration, and/or design of algorithms. In particular, we first formally define the algorithm selection (AS) and configuration (AC) tasks, which underlie the different approaches in automating algorithm engineering. Next, we review the literature on the two best-established related research areas, namely *automatic algorithm configuration* and *automatic algorithm design*, the focus of this thesis.

2.3.1 Algorithm selection and configuration

The algorithm selection task was first formally defined by Rice [200] in 1976. In particular, he identified a recurring pattern of having a set of available algorithms from which one has to select the algorithm that optimizes a given performance metric on a target application problem. Framed in this way, an effective abstract approach to this issue would probably be applicable to any concrete example of this selection task. A formal definition is provided as follows.

Definition 16 (Algorithm selection) *Given a set of instances $\Pi_{\subset}$ of a given problem Π and a set of algorithms Ψ_{Π} available for solving $\Pi_{\subset}$, select the best algorithm $\psi_{best} \in \Psi_{\Pi}$ for solving $\Pi_{\subset}$ according to a given set of performance metrics $\hat{\mathcal{C}}$.*

In particular, the algorithm selection formulation is flexible w.r.t. considering subsets of $\Pi_{\subset}$, Ψ, and/or $\mathcal{C}$. For instance, in general a single performance metric is considered. In addition, the major focus of the research on algorithm selection concerns *per-instance algorithm selection*, the goal of which is to identify a mapping between instances and algorithms, that is, a best algorithm $A^{\pi_i}_{best} \in \Psi_{\Pi}$ for each instance $\pi_i \in \Pi_{\subset}$. Despite its flexibility, the general AS formulation defines algorithms $\psi_j \in \Psi_{\pi}$ with no regard for their eventual numerical parameters, assuming that they are optimally configured. A problem that is complementary to the algorithm selection task is then the algorithm configuration task, which can be formally defined building on the following definition.

Definition 17 (Configuration and configuration space) *Let ψ be a parameterized algorithm with a set of parameters $\Phi_{\psi} = \{\phi_i\}$, each with domain $\mathcal{D}_{\phi_i}$. The* configuration space Θ_{ψ} *of ψ is given by the cross-product of all domains D_{ϕ_i}, and comprises all possible* configurations $\theta_{\psi} = \{\theta_{\phi_i}\}$*, i.e., vectors comprising a value $\theta_{\phi_i} \in D_{\phi_i}$ for each parameter $\phi_i \in \Phi_{\psi}$.*

Definition 18 (Algorithm configuration) *Given a set of instances $\Pi_{\subset}$ of a given problem Π, an algorithm ψ with a set of parameters Φ_{ψ} and configuration space Θ_{ψ}, find the configuration $\theta_{\psi} \in \Theta_{\psi}$ that optimizes a given performance metric $\hat{c}(\psi, \theta_{\psi}, \Pi_{\subset})$, i.e., running ψ on $\Pi_{\subset}$ using configuration θ_{ψ}.*

In the literature, the algorithm configuration task is also called the *tuning task*, since a properly set parameter configuration can also be referred to as *tuned settings*. Although complementary, algorithm selection and configuration differ in their nature, and so the first is traditionally approached as a feature-based machine learning prediction problem, whereas the latter can be the subject of both machine learning and heuristic optimization approaches, as we briefly introduce below:

1. **Portfolio-based AS (PbAS)**[9, 88, 184, 194, 225]: the goal of this research approach is to devise algorithm *portfolios* comprising instance-specific, high-performing algorithms and thus maximizing the effectiveness of the portfolio on an instance benchmark. In particular, this approach combines offline and online strategies. Offline, machine learning techniques taking instance features as input are devised to identify effective algorithms for given instances[10]. Hence, a subset of Ψ_{Π} that is considered effective for different instance classes is chosen to comprise a portfolio. Online, the selector defines which algorithm(s) of the portfolio will be run based on the features of the target instance.

2. **Automatic AC** [8, 17, 37, 113, 115, 130]: this research approach aims to automate the AC process. Given a configuration budget, typically the number of experiments or a maximum runtime, approaches search the configuration space to retrieve a high-performing parameter setting for a given instance benchmark set. It is also possible to consider per-instance automatic algorithm configuration, and in this case the configurator (tuner) returns a set of configurations $\theta_{\psi,\Pi}$ to be used by algorithm ψ for subsets of input instances that share structural features [130].

[10] In general, parameters are not considered part of the feature space used to create models.

Next, we further detail the latter research line discussed above, since it is tightly related to the research on automatic algorithm design, which we present right after.

2.3.2 Automatic algorithm configuration

A significant research effort has been devoted over the past decade to automating the algorithm configuration process. In particular, a number of *automatic algorithm configuration* tools, or *configurators* for short, have been proposed [75, 107]. The most emblematic example is probably the commercial mixed-integer programming solver IBM-ILOG-CPLEX, which now ships with an integrated configurator to help end users fine-tune the nearly hundred relevant parameters it presents [118]. In fact, this initiative is a result of the significant improvements demonstrated by the automatic configuration community on the runtime required by CPLEX for solving particular problems once properly tuned [114], sometimes surpassing 50-fold speedups over the default settings previously recommended by the CPLEX team. Another direct benefit of the automatic algorithm configuration methodology is encouraging developers to expose parameters that were previously hard-wired into the code, but that can be handled more appropriately by applying automatic configuration to the target domains, as advocated by the proponents of a software design approach know as *programming by optimization* [108]. In its most advanced version, this design paradigm gives rise to the augmented automatic configuration approaches in automatic algorithm design, as we will later discuss in this section.

Ironically, the fast rise in the number of configurator proposals has created a recursive problem. Since the computational overhead for a rigorous experimental assessment of a configurator is considerable (and without large computing clusters, unfeasible), most configurators have only been compared against one or two other configurators, whereas others have been directly tested on application problems. Although it is not our intention to discuss the empirical performance of any configurator, it is clear that a practitioner needing to select a configurator to tune an algorithm is now faced with the problem of selecting an appropriate tool without knowing the practical (dis)advantages they offer. In some cases, this lack of clearly defined practical guidelines will simply incur in a greater overhead for setting up the configurator. In most extreme situations, however, the practitioner might (i) have to transform his data to fit the requirements of a given configurator (e.g., discretizing real-valued parameters), or (ii) not be able to conduct post-configuration analysis about why specific parameter values have been selected. More importantly, a poor separation between training and testing instance sets might lead to an overfitted configuration being selected [35].

In this section, we present a conceptual analysis of existing configurators that is complementary to the existing automatic algorithm configuration surveys [75, 107] because it is at the same time practical, revelant, and succint. More importantly, this analysis helps us identify the most suited configurator for our purposes, and ensures the experimental soundness of our work.

Conceptual definitions

Since automatic algorithm configuration is a field derived from several research areas, we incrementally address the concepts originating from each of these fields, namely algorithm configuration and the related research on experimental design and artificial intelligence. Next, we detail concepts particular to automatic configuration, which are instrumental for our analysis and guidelines.

Algorithm configuration. Overall, the parameters of an algorithm can be of two primitive types:

- **Numerical:** a parameter ϕ_i is said to be *numerical* if its domain is $\mathcal{D}_{\phi_i} \subset \mathbb{R}$ (a *real*-valued parameter) or $\mathcal{D}_{\phi_i} \subset \mathbb{Z}$ (an *integer*-valued parameter). Examples are the variation probabilities in evolutionary algorithms or the temperature parameter in simulated annealing.

- **Categorical**[11]: a parameter ϕ_i is said to be *categorical* if its domain $\mathcal{D}_{\phi_i} = \{\nu_i\}$, where each ν_i is a discrete option and no ordering relation can be used for $\mathcal{D}_{\phi_i}$. For instance, a set of different neighborhood operators can be available for a local search algorithm.

It is also important to distinguish between the two similar concepts below:

- **Parameter interaction:** two (or more) parameters are said to interact when the effect of simultaneous changes to these parameters differs from the effects of individually changing them. For instance, crossover and mutation rates in evolutionary algorithms jointly regulate the balance between intensification and diversification. Parameters that interact cannot be configured independently.
- **Parameter dependency:** some parameters are only used when specific values for other parameter(s) is (are) selected. For instance, choosing k-means as a clustering approach requires specifying another parameter k. Parameters such as k are known as *conditional parameters*.

Experimental design. Given the configuration space Θ_ψ of an algorithm ψ, the experimental design literature has several different proposals for its analysis. The most important concepts related to this field are define below.

- **Latin hypercube sampling (LHS)** [165] is an example of *partially factorial design*, selecting a subset of the configuration space to be evaluated while trying to maximize its representativeness. Specifically, a factorial design amounts to the exhaustive[12] evaluation of the configuration space, which is generally infeasible in practice even for a moderate number of parameters. By contrast, LHS is a sampling technique based on the properties of latin hypercubes. More precisely, a latin hypercube is a generalization of latin squares, i.e., square grids where there is only one point in each row and in each column. In LHS, given n parameters ϕ_i and a numerical value m used to divide each domain $\mathcal{D}_{\phi_i}$ in equal intervals, a set of k configurations is generated while ensuring that no parameter value is repeated across different configurations. Clearly, k and m define both the computational complexity and the representativeness of this approach.
- **Response surface models (RSMs)** [41] are an important algorithmic approach used as underlying model in some algorithm selection and configuration techniques. In a nutshell, these approximative models learn from the performance patterns of a given algorithm on a set of instances with particular features, and are able to predict how the algorithm will perform on an unseen instance. Its practical applicability is large, ranging from algorithm selection and configuration to understanding what features of a given problem pose extra hardness for algorithms in general. The most important elements for devising an RSM are (i) feature extraction and selection; (ii) regression approaches, and; (iii) the performance metric one wants to predict.

One important remark about the analysis of heuristic optimizers is that these algorithms are generally stochastic, and hence any experimental design approach must consider *repetitions*, i.e., multiple runs using different random seeds. While some approaches are naturally equipped to deal with repetitions, in some cases aggregative approaches are used, such as mean, quartiles, median, or rank sums [51].

Artificial intelligence. Algorithm configuration shares characteristics of both an optimization problem and a machine learning (ML) problem. From an optimization approach, the parameters of an algorithm are the decision variables of a problem, the configuration space is the decision space, and the

[11] Categorical parameters have also been called *symbolical* in the literature [211].

[12] Since it is impossible to evaluate all possible values for continuous-valued parameters, a discretized configuration space is often used instead.

performance metric is the optimization function. From an ML perspective, tuning benefits from modeling feature-performance interactions, such as instance-configuration correlations. The most relevant concepts originated from the AI community are listed below.

- **Training/test set separation:** machine learning algorithms are subject to overfitting, i.e., being too specific to the previously observed training data and thus being little generalizable to unseen data. To prevent this, the problem instances one has at hand are typically divided into a training set and a test set [35]. The training instances are only used for learning, whereas the test set is used to evaluate how generalizable the learning of the algorithm is. In addition, it is also possible to evaluate the prediction performance of an algorithm during training by using cross-fold validation, i.e., partitioning the training set into subsets that are interchangeably used as in the overall training/test separation.
- **Racing** [168] is an approach used to reduce the computational cost of running configurations on instances. Specifically, a set of configurations is run on an incremental instance set, and poor-performing solutions are discarded along the race. In general, statistical tests are used to determine whether configurations can be considered to show worse performance than others. Since racing is based on discarding poor configurations, it is a safer approach than trying to lower the computational overhead by reducing the instance set size.

As we will later see, all of the above concepts are central in AI-based algorithmic configuration, regardless of using a search optimization or a machine learning approach. It is also important to remark that tackling algorithm configuration as a meta-optimization problem as done by the AI community establishes a recursive pattern of having configurators with their own parameters to be configured, with no base case. Nonetheless, using configurators rather than traditional optimizers is interesting because configurators (i) are custom-tailored to the algorithm configuration experimental setups, landscape characteristics, and parameter types, and (ii) present less parameters than most heuristic optimizers, with some having been shown to be robust to different parameter settings [117, 190].

Additional concepts. The most important concepts that have originated as a result of the research on automatic algorithm configuration are listed below, and are instrumental to our later analysis.

- **Cardinality of $\Pi_{\subset}$:** the clear goal of automatic algorithm configuration is to maximize the performance of an algorithm on an instance set. This can be accomplished in a general way, when a single configuration is returned to be used for the whole instance set (*multi-instance*), or on a *per-instance* basis, when the configurator returns a configuration for each instance subset it identifies.
- **Model use:** configurators that are search-based or that rely on clustering are called *model-free*. By contrast, some configurators use *model-based* search. In this case, some use models as a surrogate configuration evaluation technique, whereas others use models to bias the search in the direction of promising configurations. Finally, model-based configurators can either build (i) a single, *global* model, like in RSM-based configurators, or; (ii) a set of local models that only concern individual candidate configurations, like in EDA-based configurators.
- **Instance-parameter correlations:** per-instance and some multi-instance configurators are able to account for instance-parameter correlations. This is particularly important for further analysis conducted post-tuning, since one can try to understand the effects of specific instance features/classes in the selected parameter ranges.
- **Sharpening:** since configurators are iterative learning approaches, configurations sampled in later iterations tend to be better performing than configurations produced earlier. For this reason, a

more accurate comparison methodology is required at later iterations to distinguish between configurations. Sharpening can be implemented in different ways, such as increasing the number of (i) instances or (ii) repetitions used to test configurations in later iterations. In addition, configurators that adopt racing naturally enforce sharpening as surviving candidates are tested on more instances than configurations discarded early.

- **Capping:** when runtime is used as performance metric, it is not interesting to wait for poor performing candidates to finish their run, potentially wasting tuning budget. Capping is a mechanism used to terminate runs from candidates that extrapolate a given runtime. This capping runtime is usually either (i) a pre-defined configurator parameter, or (ii) dynamically defined as a function of the runtime required by the best-performing candidate for the given instance.

- **Tuning budget:** an automatic algorithm configuration process is generally terminated based on pre-defined stopping criteria. In general, this means exhausting a *tuning budget*, i.e., a maximum runtime the configurator is allowed to run for or a maximum number of experiments it is allowed to perform. In some configurators, though, it is also possible to terminate when either no improvements are observed over a series of iterations or when a minimal number of configurations can no longer be distinguished between.

Practical guidelines

Based on the concepts from the previous section, we detail a set of practical guidelines that help end users select among different configurators. In particular, we base our configurator choice for the remainder of this thesis upon these guidelines.

As a rule of thumb, the first consideration a practitioner needs to have in mind when selecting a configurator is instance set cardinality. Defining between a single parameter configuration or a set of instance-class configurations depends directly on the application domain of the end user. In academia, for instance, per-instance parameter settings are not an acceptable standard when comparing a novel algorithm to existing ones if existing algorithms have been tuned on a multi-instance scenario. In real-world applications, however, such concern is not a problem, with customization being desirable as long as it produces better results.

The second most important consideration concerns the goal of the automatic algorithm configuration process. More precisely, some practitioners are only interested in ad-hoc configurations regardless of understanding why such parameter values have been selected. This can be the case, for instance, when comparing a novel algorithm to existing ones, since one assumes a detailed analysis has already been conducted by the proposers of existing algorithms. Conversely, when one is more interested in understanding parameter effects, e.g., when using tuning within the algorithm engineering process, model-based approaches can be instrumental, or at least certainly more straightforward. In addition, this kind of analysis can greatly benefit from models that encompass instance features. It is also possible to use a per-instance approach and for each cluster use a model-based approach.

A third important consideration regards the type of parameters the problem requires. Discretizing numerical parameters for categorical-only configurators is an approach that discards potentially valuable information from the configuration space. In addition, it is also important to consider whether configurators that accept numerical parameters allow different tolerance levels for each parameter, as this can vary considerably depending on the given parameter.

Finally, racing, sharpening, and capping are desirable features configurators should present since they have the potential of maximizing the effectiveness (or at least the efficency) of an automatic algorithm configuration campaign. In particular, capping can make the difference between two high-performing configurators when using runtime as performance metric.

configurator	$\Pi_{\subset}$ cardinality	numer.	catego.	model use	parameter interaction	inst.-param. correlation	racing	sharp.	capping
ParamILS [113]	*multi*	✗	✓	*free*	✗	✗	✓	✓	✓
GGA [8]	*multi*	✓	✓	*free*	✗	✗	✓	✓	✓
REVAC [182, 211]	*single*	✓	✗	*local*	✗	✗	✓	✓	✗
irace [14, 159]	*multi*	✓	✓	*local*	✗	✗	✓	✓	✗
SPO [16, 17]	*single*	✓	✗	*global*	✗	✗	✗	✓	✗
SMAC [115]	*multi*	✓	✓	*global*	✓	✓	✓	✓	✗
GGA++ [10]	*multi*	✓	✓	*global*	✗	✗	✓	✓	✓
ISAC [130]	*per-instance*	*	*	*	*	✓	*	*	*

Table 2.5: Non-exhaustive description of configurators. We refer to the text for an explanation on ISAC.

Configurator overview

A broader summary of configurators and their characteristics according to the conceptual definitions introduced in the previous sections are provided in Table 2.5. However, an in-depth analysis of all automatic algorithm configuration tools available would not fit this section and escapes the goal of this thesis, and so the reader is referred to [75, 107] for this purpose. In this section, we review the configurators that best fit the context of this thesis according to the previously discussed guidelines, being either the most recent or the most cited from the automatic configuration literature.

ParamILS [113] is a local-search based algorithm that automates the sequential trial-and-error method. For ParamILS, the quality of a parameter configuration ϕ is given directly by an estimate measure $\hat{c}(\phi)$, such as mean runtime or median solution quality. Starting from an initial parameter configuration, ParamILS iteratively alters the current incumbent solution by modifying only one of its parameter values at a time. The modified parameter configuration ϕ' is considered better than the original configuration ϕ if $\hat{c}(\phi') \leq \hat{c}(\phi)$ (considering a measure to be minimized). Since it is a local-search based algorithm, ParamILS uses a *perturbation* procedure to prevent getting trapped in local optima, consisting of simultaneously altering several parameter values of the incumbent solution. Moreover, a restart mechanism ensures the algorithm explores different regions of the parameter search space. The main advantages of ParamILS are that it uses racing, sharpening, and capping. As drawbacks, real-valued parameters need to be discretized and, since models are not used, it is not straightforward to analyze parameter interactions nor correlations between specific instances and parameter configurations.

irace [14, 159] is an estimation of distribution algorithm (EDA) that encodes its learning as probability distributions that are used to sample configurations and race them. More specifically, a solution in irace comprises a configuration and a set of parameter-wise probability distributions. At each iteration, offspring configurations are sampled as follows. Given a parent and its parameter-wise probability distributions, a configuration is sampled and inherits the distributions from its parent. These offspring configurations are then tested on a subset of instances by means of racing. Although any racing procedure could be used by irace, F-Race is the typical approach adopted. Statistical methods adopted are Friedman's non-parametric statistical test or Student's t-test, to be selected by the user according to the characteristics of the data [51]. Once racing is concluded, irace learns: given a surviving solution s, it updates the probability distributions of s to add bias in favor of the parameter values presented by $\phi(s)$, its associated configuration. Effectively, the parameter-wise distributions of s are biased towards the regions of the configuration space where the performance of the configuration associated to s according to $\hat{c}$ is maximized. If a new iteration is to be started, a new set of offspring is sampled based on these updated probability distributions. Else, the configurations of the surviving candidates are returned.

The major advantages of irace is dealing with multiple instances and with all parameter types. In

addition, racing and sharpening are used, and a number of application works have demonstrated its effectiveness in comparison to manual tuning. By contrast, the main drawback of irace are that it does not (i) implement capping, since its primary use is for solution quality tuning scenarios, and; (ii) explicitly model instance-configuration correlations.

GGA [8] is a gender-based genetic algorithm that uses an AND/OR tree to encode parameters as chromosomes, depicting the dependencies they display. The population comprises individuals from two different genders: (i) a *competitive* one that is raced on instances to evaluate their quality, and (ii) a *non-competitive* one that is only used for diversity preservation. Specifically, the mating selection procedure of GGA selects the best competitive individuals and mates them with non-competitive individuals. The environmental selection is *age-based*, i.e., individuals are removed from the population within a given number of iterations after their creation. The main advantages of GGA are that its tree-based representation and operators can deal with numerical and categorical parameters in a straightforward way. In addition, racing, sharpening, and capping are used. The major drawbacks of GGA are that it does not explicitly account for parameter interactions nor for instance-configuration correlations.

SMAC [115] uses the idea of sequential model-based optimization (SMBO), in which RSMs are constructed/refined at each iteration. More precisely, SMBO approaches initially sample configurations using a given method (LHS, for instance) and fit an RSM. During consecutive iterations, novel configurations are sampled (using LHS, for instance) and evaluated according to the RSM. Selected configurations expected to be high-performing are raced against the best-so-far configuration found. At the beginning of each iteration, the RSM is refined to learn from the performance of the novel configurations. By the end of the iterative process, a high-performing configuration is returned.

SMAC extends previous SMBO approaches by allowing (i) different machine learning methods to fit the RSM, (ii) the inclusion of categorical parameters, and; (iii) using instance sets instead of a single instance. Concerning machine learning methods, the authors proposed adaptations to use random forests or Gaussian processes. The first deals naturally with categorical parameters, whereas the latter is adapted using a Hamming distance-based kernel. To be applicable to instance sets, SMAC includes instance features as part of the model fitting procedure. Besides these generalizations, the most significant difference between SMAC and previous SMBO approaches is the intensification process used by authors to sample configurations. First, SMAC maintains a single incumbent solution all along the algorithm configuration process, and races it against a novelly sampled candidate solution on the same instances on which the incumbent solution has been evaluated to reduce variance. Second, rather than simply using randomized sampling of candidates at each iteration, SMAC employs a local search to retrieve a set of high-performing configurations for the current model. To avoid getting trapped in local optima, the algorithm intertwines high-quality and randomly sampled candidate solutions for every instance considered in the intensification procedure.

The main advantages of SMAC are its ability to (i) deal with numerical and categorical parameters, and (ii) explicitly account for instance features and parameter interactions. In addition, SMAC uses racing, sharpening, and has been tested with several different machine learning methods and shown to work well on most scenarios [117]. As a drawback, the major bulk of the research on SMAC is restricted to runtime as performance metric, and its effectiveness is based on the quality of instance features, not readily available for all NP-hard problems.

GGA++ [10] is a model-based version of GGA, where the crossover operator is guided by a random forest model to target promising regions of the configuration space. More precisely, authors proposed a random forest model focused on splitting the configuration space between high- and

poor-performing parameter configurations. This way, offspring individuals are generated with more chances of being high-performing. The main advantages of GGA++ are the same previously discussed for GGA. The major drawbacks of GGA, however, were not addressed by GGA++, namely explicitly modeling the possible interactions between parameters or instance-configurations.

Experimental analyses

Two types of experimental evaluations are of interest to automatic algorithm configuration, namely (i) evaluating the efficacy of a configurator when compared to manual tuning, and (ii) comparing different configurators on similar benchmark problems. In the former case, the large number of citations each of the different configurators considered in this work has received over a short span of time confirms their effectiveness. More importantly, the total number of configurator citations evidences the practical demand for high-performing automatic configuration tools.

Works of the second type are scarce for two main reasons. Firstly, a representative analysis of this kind would require a set of considerably different benchmark problems and algorithms. An initiative in this direction is AClib [116], a benchmark library recently proposed for evaluating algorithm configurators. A second challenge is to consider different scenarios, such as (i) multi- and per-instance setups, (ii) benchmarks comprising different parameter type combinations, (iii) different performance metrics to be optimized, and (iv) using manually-tuned configurations as initial candidates. The last and most significant challenge to an experimental assessment of tuners is the computational cost of such an undertaking. In particular, we strongly believe that this is the kind of research work that will require a significant collaboration from several research groups, and require massive cluster or cloud computation.

In addition, before discussing experimental analyses, we remark an issue that is critical for any automatic configuration campaign, namely the proper separation between training and test set. In particular, this division should try to generate sets that are representative of each other concerning factors such as homogeneity and instance structural characteristics. For instance, tuning only for easy or hard problems may lead to a configuration that is poor performing on the other type. One particular concern is the situation where few instances are available. In this case, the cross-fold validation approach is recommendable since it can effectively multiply the seen/unseen data ratio.

Below, we individually address the experimental works we identify in the literature:

- **SMAC versus GGA** [115]: The authors considered two SAT solvers (SAPS and SPEAR) and IBM-ILOG-CPLEX, both to minimize runtime. Training and test set separation was respected, and experiments were split into single- and multi-instance. On multi-instance scenarios, SMAC outperformed GGA as configuration space size growed. Interestingly, authors also considered scenarios where the configuration space was discretrized, and showed that original SMAC outperformed discretized-space SMAC for several scenarios. This is further empirical evidence that confirms the importance of adopting configurators suited to the application domain parameter types.

- **GGA++ versus SMAC and GGA** [10]: The authors compared GGA++ with its predecessor GGA and with SMAC for tuning two SAT solvers[13] based on runtimes on a set of industrial SAT instances. For a small configuration space size, all configurators improve over default settings and rank as follows: GGA++, SMAC, and GGA last. For a larger configuration space size, however, neither GGA nor SMAC were able to improve over default settings, whereas GGA++ was. We remark though, that SMAC is reportedly known to be poor-performing on the industrial SAT instances as a function of the instance features currently available for this set. In addition, only two solvers were used as algorithms to be configured. We then believe that, although these results depict a possible and real scenario, they should not be generalized regarding the poor performance of SMAC.

[13] glucose 4.0 and lingeling. For solver references, we refer to [10].

As it can be seen, the experimental analyses addressing more than one configurator are few and generally restricted. In addition to the works discussed above, we also mention the analyses that have demonstrated the parameter robustness of irace [190] and SMAC [117].

Configurator choice and discussion

Overall, the analyses reviewed in the previous section evidence that little comparison between different configurators on tuning scenarios that consider solution quality as performance metric have been conducted so far. In fact, we could not identify any published experimental work of such kind, although the effectiveness of configurators in general on these scenarios is beyond question. Moreover, GGA++ was proposed when this thesis was already nearly completed, and ParamILS requires discretization of numerical parameters. For these reasons, we restrict our decision to SMAC, irace, and GGA. Concerning the choice between GGA and SMAC, we select the latter over the former given the direct experimental comparison previously discussed for runtime prediction scenarios. Regarding the choice between irace and SMAC, the latter is highly dependent on a rich feature set available for the application problems one wants to target, but no feature proposal concerning the problems we consider in this thesis can be found in the literature. In particular, we remark that we regard this as an interesting research direction in which the work described in this thesis should be extended. For this investigation, though, targeting both the feasibility of automatic MOEA design and the delimitation of problem-wise feature sets would either slow down or even hinder the practicality of our work.

2.3.3 Automatic algorithm design

Beyond the challenges of automated algorithm selection and configuration, a yet more ambitious research area concerns the *automatic algorithm design*. Although approaches in the literature differ as to the final product and the procedure to design this product, the overall goal of automatic design is to automate the design process of algorithms, particularly when dealing with specific problems or experimental setups. Obviously, a completely human-free algorithm design is still out of reach, with automatic algorithm design approaches automating specific parts of the design process and in many cases relying on existing human knowledge. However, given the numerous potential applications of automatic design, the challenge it poses is currently being addressed by an ever-growing number of researchers. In fact, there has been an incipient but steadily developing research effort on automatically designing complex algorithms, such as SAT solvers [133], multi-objective meta-heuristics [70, 157], machine learning tools [217], or even algorithm portfolios [208, 226] and selectors [154]. Overall, results are promising yet demonstrate that the effectiveness of automatic algorithm design is tightly coupled with making proper use of human-crafted knowledge. More precisely, from a high-level perspective automatic design approaches can be categorized as (i) *bottom-up*, where heuristics are crafted using little human insights and heavily relying on automatically-discovered knowledge, or (ii) *top-down*, where human knowledge provides a structural basis (e.g., a template or a grammar) and the automated design process attempts to design the best possible algorithm based on this structure. While a number of *bottom-up* approaches exist with *genetic programming* [141] being the most consolidated bottom-up automatic design research field, the scope of that research has been traditionally restricted to the design of heuristics. By contrast, top-down approaches have been increasingly proven effective whether for designing simple heuristics or complex algorithm portfolios.

Since automatic algorithm design is a primary focus of this thesis, we review the most important subfields within this research area. In particular, we start our discussion with the research on a field of genetic programming that has been used as a top-down automatic design approach to hyper-heuristics, namely *grammar-based genetic programming*. Although works of this type are generally restricted to the design of heuristics, the notion of grammar-based automatic design is important for the main discussion

of this section, which we detail next. Specifically, we review and discuss the most promising top-down automatic algorithm design approaches found in the literature, categorized in this work as *augmented automatic configuration*.

Grammar-based GP

Although hyper-heuristic genetic programming (GP) algorithms are inherently bottom-up, one particular type of GP algorithm can be considered top-down, namely *grammar-based* GP (gGP [173], for short). More specifically, gGP algorithms define a grammar to determine how the genotype of a solution will be translated into its phenotype. In tree-based gGP, for instance, the genotype of a solution is a derivation (tree) from the grammar, whereas its phenotype is the expression-tree one obtains when only terminal symbols from this derivation are considered. The major benefits of using grammars on hyper-heuristic GP algorithms are that they (i) embed human knowledge, and (ii) allow more complex heuristics/algorithms to be created. More generally, the major advantage of grammars is providing a structure way to navigate the search space size while maintaining a high expressivity, allowing, for instance, recursive rules. By contrast, it is also possible that this advantage becomes a drawback, either if the optimal solution cannot be represented using the grammar adopted, or if recursion is allowed to go too deep.

Notwithstanding the potential benefits of grammars, tree-based gGPs still present the same indirect encoding drawbacks from tree-based GPs. For this reason, linear gGPs were proposed a few years later. Within this context, grammatical evolution (GE) [185] has stirred a few works and has been adapted for the context of generating heuristics. In principle, GE algorithms differ from tree-based gGPs in that grammar derivations are represented as variable-length chromosomes, where each gene is an integer depicting a derivation choice, referred to as a *codon*. In particular, a grammar derivation in a GE algorithm starts from the top-most derivation rule and explores symbols from left to right. When a non-terminal symbol with a derivation rule presenting multiple choices is found, a codon is consumed. More precisely, given k alternatives for expanding a non-terminal symbol and a codon i to be consumed, alternative $k \,\%\, i$ is selected (where $\%$ represents the integer modulo operation). A few remarks concerning this solution representation deserve special attention. First, the number of codons required to conclude a derivation is not ensured to match the solution size. In the case when a derivation is concluded but codons are still available, GE algorithms traditionally disregard the remaining ones. By contrast, when codons have all been consumed and the derivation has not yet been concluded, *wrapping* is generally allowed, i.e., reusing codons from the start of the solution. Clearly, the coupling between grammars and this encoding is loose, one of the major disadvantages of GE. Another example of this loose coupling is the integer modulo operation required for choosing derivation alternatives, since there is no clear connection between the integer range and the number of derivation choices available. In fact, a single codon may be used for the expansion of several different non-terminal symbols in case wrapping is adopted. Despite these potential drawbacks, several gGP hyper-heuristics works can be identified in the literature [45]. More importantly, the notion of grammar-based automatic algorithm design is central to the next set of approaches we discuss.

Augmented algorithm configuration

Applying automatic configuration tools to the context of automatic design is a natural consequence of exposing parameters that were previously hardwired into the code [108]. In particular, an augmented algorithm configuration approach expands the configuration space to be searched by configurators such that design choices can also be considered. More precisely, the *design space* of an augmented algorithm configuration approach is defined with the help of a human-designed structural pattern, e.g. a template or a grammar, delimiting how low-level components can be combined to produce reasonable algorithmic designs. It is also important to remark the blurry difference between categorical parameters and design

choices. On one hand, the latter is generally represented as the former, and the traditional examples of categorical parameters such as the choice of a local search or a crossover operator can be considered design choices. On the other hand, augmented automatic configuration approaches propose a much more high-level perspective to existing algorithms from a given field. In particular, the process of crafting a template or a grammar depends on finding design patterns in the existing literature on a given topic, and proposing flexible ways of recombining these components in human-reasonable ways. Often, these works propose novel unified models, revealing the equivalence and interchangeability of components that had been independently proposed. For instance, GAs and DEs are traditionally seen as intrinsically different algorithms, but in this thesis we show how to consider these two approaches as categorical design choices for a configurator to select between.

Since template- and grammar-based approaches differ considerably as to their nature, we next review the main insights and proposals from each group individually.

Template-based approaches comprise the union of configurators with flexible, template-based algorithmic frameworks. Specifically, it is implemented by adding the configurable algorithmic components of the framework to the parameter space to be searched by the configurator. In a template-based approach, a design choice derives from deconstructing existing algorithms into algorithmic component patterns, thus providing different categorical choices to be selected by the configurator. A few applications of template-based approaches deserve to be mentioned due either to their historical relevance or to their connection to this thesis, which we review next.

- **SATenstein** [133] was the first proposal of template-based augmented algorithm configuration, used for automatically building a stochastic local search (SLS) SAT solver. In particular, authors identified that the literature on SLS SAT solvers comprised four major algorithmic groups, three of which were used during the deconstruction stage to assemble the SATenstein-LS framework. The template comprising the design choices provided by SATenstein-LS was used to define the augmented configuration space (design space) to be searched by the configurator (ParamILS [113]). Results were remarkable, with the automatically produced SLS SAT solvers outperforming all the other 11 SAT solvers considered, selected due to their good performance on SAT competitions.
- **The MOACO framework** [157] was the first proposal of template-based augmented algorithm configuration applied to a multi-objective optimization scenario. In particular, authors expanded an existing *ant colony optimization* (ACO) framework [214] to deal with a bi-objective optimization problem, namely the TSP. More importantly, the different design choices used to assemble this framework were gathered by deconstructing the most relevant multi-objetive ACO (MOACO) proposals from the literature. The augmented configuration space was defined based on the MOACO template that underlies this framework. In addition, this was also the first work to consider a separation between multi-objective components and underlying algorithms, with the two most used ACO algorithms from the literature begin available as design choices[14]. Results were once again remarkable, with the automatically designed MOACO algorithms outperforming by a large margin the MOACO algorithms from which the framework components were gathered. Later, this work was extended to tackle combinatorial problems differing in nature to the TSP [22]. In Appendix D of this thesis, we describe such a work applied to the bi-objective bidimensional knapsack problem.
- **The TP+PLS framework** [70] was the first proposal of template-based augmented algorithm configuration that considered hybrid metaheuristics. In particular, the two most

[14] Ant colony system (ACS) and $\mathcal{MAX}$-$\mathcal{MIN}$ ant system ($\mathcal{MM}$AS). For further reference on ACO, the reader is referred to Dorigo and Stützle [65].

commonly adopted SLS methods for bi-objective optimization were selected[15], and the most relevant proposals for each method were identified and deconstructed to provide components for a hybrid framework. We remark that, although algorithm design was done automatically, the hybridization between metaheuristics was an *a priori* human-designed stage, represented by the given template. Authors considered the bi-objective PFSP as application problem and irace as configurator. Results once again showed the improved performance of automatically designed algorithms when compared to manually designed ones. In Appendix E of this thesis, we describe a work that was inspired by this one, in particular the automatic design of PLS algorithms applied to the bi-objective bidimensional knapsack problem.

- **AutoFolio** [155] uses the previously discussed CLASPFOLIO 2 framework to automatically design algorithm portfolio selectors[16]. More precisely, authors have identified a general pattern used by algorithm selectors and have assembled a framework from these constituent algorithmic components. Hence, given a set of training instances and a set of algorithms to comprise a portfolio, AUTOFOLIO automatically designs algorithm portfolio selectors, and evaluates them based on the algorithm portfolios they produce. The experimental evaluation conducted by the authors considered several decision problems such as SAT, ASP, CSP, and QBF. Overall, the performance of AUTOFOLIO was significantly better than the performance of all other algorithm selectors considered, even given their already good performances. In addition, AutoFolio also ranked first in an algorithm selection competition, reinforcing the quality of the automatically designed algorithm selectors.

- **Auto-WEKA** [217] uses the well-known WEKA machine learning toolkit as framework and considers its parameter space as a design space for automatically designing effective machine learning classification algorithms. Specifically, WEKA offers a set of different classification algorithms, both atomic and ensemble. In addition, many classifiers present their own parameter settings that must be properly set. Authors than considered this augmented configuration space and two different model-based configurators. Given a large classification dataset, authors showed that the produced Auto-WEKAs consistently outperformed the existing classifiers considered, demonstrating the effectiveness of augmented algorithm configuration on this application domain that is intrinsically very different from algorithms for NP-hard problems.

Grammar-based approaches While template-based approaches directly provide a set of algorithmic components that can be used to define the augmented configuration space, it can be argued that templates lack expressivity when one considers concepts such as recursive design components. An alternative to template-based approaches relies on grammars to provide this increased expressivity power. More precisely, in grammar-based augmented algorithm configuration the design space searched by the configurator is defined in function of a (context-free) grammar. However, instead of using genetic programming or any sort of evolutionary approach to grammars, the automatic algorithm design is tackled directly by configurators as follows. First, the exhaustive set of possible derivations allowed by the grammar is computed and then translated into a parametrical space in an automated way. Therefore, any parameter instantiation selected by the configurator corresponds to a valid grammar derivation and can be evaluated on the target problem. Overall, grammar-based approaches provide both benefits and drawbacks. As major advantage, algorithm designers are given enhanced expressivity, being able to produce complex algorithmic designs based on recursive rules, the most prominent being hybridization. By contrast, the maximum height of the (implicit) derivation trees produced from the selected grammar must be kept small to prevent the augmented

[15] Pareto local search (PLS) and Two-phase local search (TPLS). For further reference on SLS methods for bi-objective optimization, the reader is referred to Paquete and Stützle [188].

[16] We remark that the goal of AUTOFOLIO is not to design the algorithms used within the portfolio, but rather the algorithm selector that assembles a portfolio when given a set on existing algorithms.

parameter space from growing beyond practicality. In other words, the maximum recursion level allowed by a grammar is an important parameter that should be set accordingly.

We next review the most relevant grammar-based augmented algorithm configuration approaches identified in the literature.

- **Recursive IG algorithms** [170, 172]: authors proposed a grammar for defining recursive iterated greedy (IG) algorithms, i.e., allowing an IG algorithm to use another IG algorithm within it. To assess the effectiveness of the proposed representation, as well as the performance of augmented algorithm configuration compared to GE, authors used three different methods[17] to automatically design algorithms for the one-dimensional bin packing problem and for the PFSP with weighted tardiness minimization. In most scenarios, the proposed approach outperformed GE-based approaches, confirming the effectiveness of grammar-based augmented algorithm configuration in the context considered. Concerning recursion, authors demonstrated that it indeed improved the quality of the automatically designed IG algorithms, but that increasing the maximum allowed recursion level did not improve results further.
- **Hybrid SLS algorithms** [167, 171]: authors proposed a grammar for producing both pure[18] and hybrid SLS algorithms. In particular, the grammar models the overall structure used by SLS algorithms such as tabu search, variable neighborhood search, simulated annealing, iterated greedy, and many others [110]. As a result, the configurator is allowed to produce (i) pure SLS algorithms from one of these families, or (ii) hybrid approaches that combine algorithmic components from different families. To evaluate the proposed grammar, authors have compared the effectiveness of automatically designed (i) pure SLS algorithms using the derivation rules that apply only to a given algorithmic family; (ii) pure SLS algorithms using all derivation rules, or; (iii) hybrid methods using all derivation rules. More precisely, the difference between (i) and (ii) is that in the latter the configurator must first select which pure SLS algorithm to use, and then select its components/parameters. The configurator adopted was irace, and overall results confirmed the effectiveness of automatically designed SLS algorithms, with hybrid methods always outperforming pure ones.

As demonstrated in all works described above, augmented algorithm configuration is both a feasible and effective approach to automatic design. More importantly, a number of insights are produced throughout the deconstruction, assembling, and design phases. For instance, during deconstruction it is common to identify equivalent algorithms or algorithmic components that had been independently proposed by different research groups. During assembling, it is not uncommon to envision novel applications of existing components once a more high-level template has been identified. Finally, the automatically designed algorithms are much more reasonable from a human perspective, and hence it is possible to analyze why these designs work well and how they could be further fiddled with to produce yet more insights or become more effective.

Concluding remarks

As demonstrated in this section, the research on automating algorithm engineering has led to a rich literature with numerous examples of effective approaches that can further empower algorithm engineers in the task of creating better performing algorithms. Nonetheless, many of the proposed approaches reviewed in this section have only been applied to specific problems, requiring that researchers from different areas join this effort that can change the way algorithms are engineered in a near future. This

[17] An evolutionary GE approach, a GE approach using irace, and the proposed approach. The same grammar was used by all methods.

[18] We use the term *pure* here in contrast to *hybrid* to denote an algorithm from a given SLS family that does not use components originally proposed for other SLS families.

thesis provides a number of concrete contributions to both the literature on MOEAs and on automated algorithm engineering, as we detail over the next chapters.

2.4 Summary

In this chapter, we have reviewed the core concepts and most relevant literature concerning the research areas directly connected to this thesis. To start, we have concisely reviewed the most important concepts concerning multi-objective optimization, a field that offers problem modeling with higher accuracy at the cost of increased computational complexity. In particular, we have discussed the challenges faced by the research on multi-objective optimization, ranging from simple issues as solution comparison to the more complex task of evaluating the performance of different heuristic multi-objective optimizers. In addition, we have formally defined the application problems we consider in this thesis, and have provided discussion about the characteristics of these problems.

Next, we have presented an overview of the method of choice that has been enabling the successful applications of multi-objective optimization, namely metaheuristic algorithms. Specifically, we have first provided a general overview of the metaheuristic literature focused on the concepts that aggregate or differentiate these algorithms. More importantly, we have presented the most significant challenges faced by algorithm engineers who wish to adapt metaheuristic algorithms to tackle multi-objective optimization problems, such as the difficulty to approximate a possibly infinite set of incomparable solutions by means of a finite population or archive. Finally, we have presented a comprehensive, detailed historical review of how EAs have become the predominant metaheuristic within multi-objective optimization. In particular, we have analyzed how the most relevant algorithmic concepts used in the general multi-objective metaheuristic literature were proposed, and how the recent challenges of many-objective optimization have been pushing researchers to devise new such concepts. Last but not least, we have concisely reviewed the most relevant MOEA algorithmic frameworks that have been enabling practitioners to apply MOEAs to their target domains, and over which the proposed work improves, as we detail in the next chapter.

Finally, we have discussed the field of automated algorithm engineering, a recent yet promising research field that enables algorithm designers to deal with the plethora of algorithms and algorithmic components one comes across when facing a new application domain or a much studied one such as MOEAs. Among the different automated engineering techniques, we have first presented a practical yet comprehensive review of the most important algorithm configuration concepts and tools. In particular, we have conceptually justified the configurator choice we make in this thesis. To conclude, we have reviewed the most promising approaches in algorithm design, in particular the augmented automatic algorithm configuration approaches, to which this thesis belongs.

CHAPTER 3

An initial feasibility investigation

As previously discussed, the component-wise view of MOEAs consists in identifying individual algorithmic components in different MOEAs that have the same function and, thus, could be replaced by alternative procedures taken either from different MOEAs or newly devised. Examples are the *fitness* and *diversity* components that appear in many MOEAs (see Section 2.2.4). This component-wise view has two main benefits. First, it allows algorithm designers to identify the various options available for each algorithmic component and whether a particular combination of components, i.e., an algorithm "design", has already been proposed. Second, it allows algorithm users to adapt the design of MOEAs to their particular application scenario.

One motivation for the component-wise view of MOEAs is the development of software frameworks that help practitioners apply and adapt MOEAs to their own application scenarios. As discussed in the previous chapter, the design of most MOEA frameworks focuses on applying existing MOEAs to new scenarios, rather than on flexibly combining their components to produce new designs [38, 39, 120]. Even MOEA frameworks that allow the combination of algorithmic components from different MOEAs [153] are limited to MOEAs that are structurally similar, for example, based on the traditional *fitness* and *diversity* components. More recent MOEAs that differ from this template, such as HypE [13] and SMS [19], cannot be instantiated from algorithmic components through such frameworks. We see two reasons behind this lack of flexibility. First, the analysis of MOEA components has relied on how the algorithms were described by their original authors, and only few works try to generalize functionally equivalent concepts of MOEAs into broader concepts [146, 153, 220, 236]. Second, a high degree of flexibility in a software framework may be deemed undesired, since some configurations may produce unreasonable algorithm designs or the number of possible configurations may be too large for human designers.

In this chapter, we propose a new conceptual view of MOEA components that allows instantiating, from the same algorithmic template, a larger number of MOEAs from the literature than existing MOEA frameworks. For example, we are able to instantiate at least six well-known MOEAs from the literature: MOGA [80], NSGA-II [60], SPEA2 [234], IBEA [232], HypE [13], and SMS [19]. More importantly, our framework allows to produce a large number of novel MOEA designs that are, in principle, reasonable from a human designer point of view. This is achieved by reformulating the traditional distinction between fitness and diversity components [153, 220] as preferences composed by set-partitioning, quality

and diversity metrics [236]. In addition, different preferences may be used for mating and environmental selection. Our proposal also formalizes the distinction between internal and external populations and archives, which allows us to describe, using alternative options for the same components, algorithms as different as SPEA2 and SMS. Our proposal is implemented in a software framework from which novel MOEA designs can be instantiated by properly selecting the values of the various algorithmic components and numerical parameters.

Following previous work on automatic design (see Section 2.3.3), we apply the offline automatic configuration method irace [159] to our proposed framework, considering the various algorithmic components as categorical parameters to be set. In this sense, the augmented configuration space searched by irace is actually a design space, where different MOEA components can be combined to generate unique and possibly novel MOEA designs. We automatically design several MOEAs, called here AutoMOEAs, for several application scenarios. Our scenarios were selected to provide insights on several questions about MOEA design. The first question is whether the benchmark that guides the design process has a strong influence in the performance of the resulting design. Thus, we consider continuous optimization problems, in particular, two benchmark sets, DTLZ [61] and WFG [112], with two, three, and five objectives, since MOEAs have been primarily designed for these problems. Our results indicate the best MOEA design depends strongly on which benchmark is used for the automatic design. A second question in MOEA design is the trade-off between computationally expensive components and the quality of the results. Thus, we consider two different stopping criteria for the MOEAs: maximum number of function evaluations (FEs) and maximum runtime. By using these two setups, we are able to represent problems with computationally demanding function evaluations as well as problems where the computational overhead of MOEA components is relevant. Although the former is the typical setup considered in the MOEA literature, the conclusions obtained may not apply to the latter setup. This is demonstrated by the fact that the AutoMOEAs produced for each setup show remarkable differences. In most cases, for both setups, the AutoMOEAs are able to match, and often significantly surpass, the results obtained by the MOEAs from the literature, even after tuning their numerical parameters.

Finally, we study the differences between the AutoMOEAs obtained for the continuous optimization benchmarks and those obtained for various multi-objective combinatorial optimization problems. In a preliminary version of this work [27], we considered four multi-objective variants of the *permutation flow shop problem* (PFSP), varying the number and nature of the objectives. Since MOEAs from the literature were not originally devised for such problems, it is not surprising that we were able to generate AutoMOEAs that outperformed them. Nonetheless, the best MOEA designs differ enough from what is considered the state-of-the-art in the MOEA literature that we briefly comment the results here. These results and the remarkable differences between the MOEAs designed for continuous optimization and those designed for combinatorial optimization provide further motivation for the automatic design of MOEAs.

The overall goal of this chapter is to empirically demonstrate that it is possible to find novel MOEA designs that outperform the MOEAs from the literature by means of automatically configuring the components of our proposed MOEA template. Another objective of this chapter is to reformulate diverse MOEAs into a common conceptual view that generalizes functionally-equivalent algorithmic components and describes the available design choices. A final objective is to investigate which design choices (instead of which monolithic MOEAs) are more appropriate for the various scenarios described above.

The remainder of this chapter is structured as follows. Section 3.1 presents in detail our component-wise MOEA framework. We present empirical results and discussion for continuous and combinatorial problems in Sections 3.2 and 3.3, respectively, and conclude in Section 3.4.

Algorithm 5 AutoMOEA template proposed in this work.

```
1: pop ← Initialization ()
2: if type (pop_ext) ≠ none
3:     pop_ext ← pop
4: repeat
5:     pool ← BuildMatingPool (pop)
6:     pop_new ← Variation (pool)
7:     pop_new ← Evaluation (pop_new)
8:     pop     ← Replacement (pop, pop_new)
9:     if type (pop_ext) = bounded then
10:        pop_ext ← Replacement_Ext (pop_ext, pop_new)
11:    else if type (pop_ext) = unbounded then
12:        pop_ext ← pop_ext ∪ pop
13: until termination criteria met
14: if type (pop_ext) = none
15:     return pop
16: else
17:     return pop_ext
```

3.1 A template for designing MOEAs

The AutoMOEA template we propose for instantiating and designing MOEAs is shown in Algorithm 5. As we will explain below, from this template we can not only instantiate many well-known MOEAs, but also many new ones that have never been explored so far. The proposed template is based on the view that MOEAs can be seen as extensions of traditional single-objective EAs such as genetic algorithms [13, 60, 80, 232, 234], evolution strategies [19, 137], or differential evolution [228], extended by algorithm components that deal with the multi-objective aspects. In our template, we encapsulate the lower-level procedures in components such as Variation, which applies variation operators to the mating pool (`pool`). Additional components for tackling multi-objective problems in the Pareto sense are encapsulated in the BuildMatingPool and Replacement procedures (see lines 5 and 8 of the template, respectively). In addition, MOEAs often use their internal population (`pop`) as a bounded-size approximation to the Pareto front (i.e., as an archive) and many of them add the possibility of keeping an external (bounded or unbounded) archive ($\texttt{pop}_{\texttt{ext}}$).

Next, we describe the multi-objective components, how to instantiate some well-known MOEAs from our template, and how our approach differs from existing frameworks.

3.1.1 Preference relations in mating selection and replacement

The mating and environmental selection procedures performed by MOEAs depend on ranking solutions according to a preference relation. In general, given two solutions s_1 and s_1 and a metric Ψ to be minimized (without loss of generality), a relation $\prec_\Psi$ is defined as $s_1 \prec_\Psi s_2 \iff \Psi(s_1) < \Psi(s_2)$. In our MOEA template, solutions are ranked according to *general preference relations* [236] defined as a sequence of three lower-level preference relations: a set-partitioning relation, a quality metric and a diversity metric. First, a *set-partitioning relation* ranks solutions in a Pareto-compliant way, but does not distinguish between nondominated solutions. These correspond to traditional fitness components such as dominance depth (NSGA-II) and dominance strength (SPEA2). Because of the nature of these metrics, multiple solutions are often equally ranked. Therefore, at a second step, we use refinement relations based on Pareto-compliant *quality indicators* to discriminate between equally ranked solutions. We apply these refinement relations to the equally ranked partitions, but we do not alter the cross-partition ranks. This means that if a solution θ_1 is ranked better than another solution θ_2 according to a set-partitioning relation, then a refinement relation would never contradict this. The third type of relation is based on *diversity metrics*. These metrics do not focus on Pareto dominance, but rather on allowing MOEAs to maintain a population that represents different trade-offs between the objectives.

Table 3.1: Main algorithmic components of AutoMOEA.

Component	Parameters
Preference	⟨ Set-partitioning, Quality, Diversity ⟩
BuildMatingPool	⟨ Preference$_{Mat}$, Selection ⟩
Replacement	⟨ Preference$_{Rep}$, Removal ⟩
Replacement$_{Ext}$	⟨ Preference$_{Ext}$, Removal$_{Ext}$ ⟩

Table 3.2: Algorithmic component options available for AutoMOEA.

Component	Domain
Set-partitioning	none (—) dominance count dominance rank dominance strength dominance depth dominance depth-rank (DR)
Quality	none (—) binary indicator ($I_{\epsilon+}$ or I_H^-) exclusive hypervolume contribution (I_H^1) shared hypervolume contribution (I_H^h)
Diversity	none (—) niche sharing (σ_{share}) k-th nearest neighbor (kNN) crowding distance

Component	Domain
Selection	deterministic tournament (DT) stochastic tournament (ST) random
Removal	sequential one-shot
type (pop)	{ fixed-size, bounded }
type (pop$_{\texttt{ext}}$)	{ none, bounded, unbounded }

The structure for these general preference relations is encapsulated in component Preference (Table 3.1). Additionally, any of the three components of Preference might be empty (*none*), which means that the next component takes effect. If all three components are empty, the ranking is random.

The options available for composing preference relations in our template are given in Table 3.2. This formulation of preference relations provides flexibility when designing MOEAs for different real-world optimization scenarios. For instance, set-partitioning relations may provide enough convergence given a problem with few objectives and for which the computation overhead of quality metrics may be deemed excessive. On the other extreme, given a problem with a large number of objectives, the number of incomparable candidate solutions may be too large such that set-partitioning relations do not provide enough convergence pressure. In other cases, the time required for computing the quality metrics may be negligible compared with the cost of evaluating candidate solutions. We also remark that the original proposal by Zitzler et al. [236] allows for more complex preference models (e.g. using multiple refinement relations based on quality indicators in a sequence), but our proposal here suffices to replicate most MOEAs from the literature and allows defining new preference relations in a flexible and consistent way. Furthermore, the general preference relations we adopt overcome the problems faced by existing frameworks when instantiating some recent MOEAs such as SMS and HypE. In particular, these algorithms include components that simultaneously account for convergence and diversity, and hence do not fit the traditional separation between fitness and diversity metrics [153, 220].

The mating and environmental selection procedures (BuildMatingPool and Replacement) are defined in dependence of the general preference relations described above. BuildMatingPool comprises a preference relation Preference$_{Mat}$ and a selection method Selection as shown in Table 3.1. The methods for selection we implement for this work are listed in Table 3.2. In particular, the *tournament selection* method can be used either deterministically or stochastically. While deterministic tournaments always favor the best individual according to Preference$_{Mat}$, stochastic tournaments choose, with a probability γ, the solution

Table 3.3: Different types of archives available for AutoMOEA.

Archive Type	μ_0	FS	MC	DS	EP	Replacement
type $(\texttt{pop}) =$ *fixed-size*	μ	$+$	μ	$+$	$+$	Replacement
type $(\texttt{pop}) =$ *bounded*	$\mu \cdot \mu_r$	$-$	μ	$-$	$+$	Replacement
type $(\texttt{pop}_{\texttt{ext}}) =$ *bounded*	$\lvert\texttt{pop}_0\rvert_{\prec}$	$-$	N_{ext}	$-$	$-$	Replacement_{Ext}
type $(\texttt{pop}_{\texttt{ext}}) =$ *unbounded*	$\lvert\texttt{pop}_0\rvert_{\prec}$	$-$	∞	$-$	$-$	—

FS: *fixed-size*; MC: *maximum capacity*; DS: *dominated solutions*; EP: *part of the evolutionary process*; $\lvert\texttt{pop}_0\rvert_{\prec}$: *size after removing dominated solutions*

preferable according to Preference_{Mat}. *Random selection* chooses individuals with uniform probability, and so no preference relation is used. Component Replacement is composed of a preference relation Preference_{Rep} and a removal policy Removal. Table 3.2 lists the removal policy options we implement. *Sequential* (or *iterative* [13]) *removal* [160] discards one solution at a time and recomputes the preference relation before the next solution is discarded. *One-shot removal* [13] computes preference relations once and discards the worst solutions altogether. Although the information provided by the sequential removal policy is more accurate, this policy is computationally more demanding, which may compromise its performance in time-constrained scenarios. Additionally, if the number of offspring per generation λ is set to 1 (steady-state selection), the different alternatives for component Removal become equivalent.

The ability of using different preference relations for mating and environmental selection is, in fact, another novel feature of our template over the templates implemented by existing MOEA frameworks. Although earlier MOEAs did not foresee the benefits of this design choice, more recent algorithms such as SMS and HypE already make use of it to minimize the computational overhead of quality metrics such as the hypervolume. From a more general point of view, the flexibility provided by this design choice can be used to improve the effectiveness of the algorithm in several other ways, e.g., by combining exploitative and explorative strategies.

3.1.2 Population and archives

A population is a set of individuals, dominated and nondominated alike, that are subject to the evolutionary process. By contrast, an archive is an auxiliary set used for storing nondominated solutions found during a single run of the algorithm. In our template, we model an archive as a generalized population that may (i) only keep nondominated solutions, (ii) have unbounded capacity, and/or (iii) take part in the evolutionary process. We provide two archives for MOEAs to use: an *internal* archive $\texttt{pop}$ that takes part in the evolutionary process and can be used as a regular population or as a bounded-size archive, and an *external* archive $\texttt{pop}_{\texttt{ext}}$ that does not participate in the evolutionary process.

All options implemented here for $\texttt{pop}$ and $\texttt{pop}_{\texttt{ext}}$ are listed in Table 3.2, and we present a summary of their characteristics in Table 3.3. If $\texttt{pop}$ is set to have a fixed size (*type* $(\texttt{pop}) =$ *fixed-size*), then $\texttt{pop}$ behaves like a regular population of size μ and may contain dominated solutions. Otherwise, $\texttt{pop}$ is used as a bounded internal archive (*type* $(\texttt{pop}) =$ *bounded*), accepting only nondominated solutions until its maximum capacity μ is reached. Once the maximum capacity is reached, a replacement is carried out by component Replacement mimicking an archive bounding method [138]. When used as a bounded internal archive, $\texttt{pop}$ presents two other important characteristics. First, the initial number of solutions μ_0 in $\texttt{pop}$ is controlled by a numerical parameter $\mu_r \in [0.1, 1]$, i.e., $\mu_0 = \mu_r \cdot \mu$. Second, the preference relation used by this bounded internal archive does not use set-partitioning relations, since all solutions kept by this archive are nondominated. These characteristics make $\texttt{pop}$ flexible enough to allow us instantiate archive-based algorithms such as PAES [137], as well as algorithms such as SPEA2, which are population-based but use an archive that interferes in the evolutionary process [220].

By contrast, the external archive $\texttt{pop}_{\texttt{ext}}$ can be used by MOEAs in three different ways. First, as

traditionally used in the literature, the archive can be *bounded* to a maximum capacity N_{ext} and once the maximum capacity is reached, a replacement is carried out (see also line 10 of Algorithm 5). Component $\mathsf{Replacement}_{Ext}$ is defined analogously to $\mathsf{Replacement}$, but with its own $\mathsf{Preference}_{Ext}$ and $\mathsf{Removal}_{Ext}$ options (see Table 3.1). Since all solutions kept by the archive are nondominated, $\mathsf{Preference}_{Ext}$ does not use a set-partitioning relation. For application scenarios where the number of nondominated solutions is low, MOEAs can either use an archive without capacity constraints, i.e., $\mathit{type}\,(\mathtt{pop}_{\mathtt{ext}}) = \mathit{unbounded}$, or not use an external archive at all, i.e., $\mathit{type}\,(\mathtt{pop}_{\mathtt{ext}}) = \mathit{none}$. The ability of using a different preference relation for maintaining the external archive opens a number of possibilities for MOEA designers. For example, the preference relations used for mating selection and replacement of the (internal) population could lack the limit-stable property (see Section 2.2.2) in order to promote exploration, while the external preference relation could be both limit-stable and limit-optimal such that, eventually, the external archive will converge to an optimal (bounded) archive.

3.1.3 Differences from existing frameworks

A number of MOEA frameworks can be found in the literature, as reviewed in Section 2.2.4. Together, they have made the application of MOEAs to new scenarios much easier by establishing a clear separation between problem-dependent and independent components. However, as previously discussed these software frameworks include implementations of the most popular MOEAs but their algorithmic components are often not directly inter-changeable. Thus, designing a novel MOEA by combining existing components in novel ways using most of these frameworks is not a straightforward task that can be done in an automatic manner, since they were not created with this goal in mind.

The framework that most closely resembles our proposed template is ParadisEO-MOEO [153], which provides a "unified model" for MOEAs that allows both the instantiation and the design of novel MOEAs using a template. However, the generality of the template used by ParadisEO-MOEO is limited when compared to the template we present here in at least four major aspects. First, ParadisEO-MOEO uses the traditional approach of preference relations built solely from fitness and diversity components, which is insufficient to represent complex preference relations as we do in this work. Second, these fitness and diversity components cannot be used with different behaviors for mating selection and replacement. This is highlighted by the fact that the default implementation of SPEA2 in ParadisEO-MOEO is not instantiated via their template, but requires an external archiver specifically designed for SPEA2 to work. Moreover, using the template provided by ParadisEO-MOEO, one cannot instantiate or design algorithms that use different preference relations for mating selection and replacement, such as HypE or SMS. Third, our internal population definition is a unique contribution, since it allows us to instantiate both population-based and archive-based MOEAs, such as PAES. Finally, many of the components we use in this work are not available in ParadisEO-MOEO, such as the I_H^1 and I_H^h quality indicators, or are only partially available, such as the sequential removal policy. Altogether, these aspects limit the number of MOEAs that can be represented using the template provided by ParadisEO-MOEO, and hence the number of possible designs one can instantiate through it.

The novelty of our proposal lies in the algorithmic template and the definition of its components, rather than in the software implementing them. In fact, the implementation of most of our algorithmic components is taken from the ParadisEO-MOEO [153], PISA [39], and PaGMO [38] frameworks, although we substantially modified them to work together within our algorithmic template. Nonetheless, it would be feasible to implement our proposed template within any of these frameworks, and we would like to encourage others to do so.

Table 3.4: Instantiation of MOEAs from our proposed template.

	BuildMatingPool				Replacement			
Algorithm	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal
MOGA [80]	DT	rank	—	sharing	—	—	—	generational
NSGA-II [60]	DT	depth	—	crowd.	depth	—	crowd.	one-shot
SPEA2 [234]	DT	strength	—	kNN	strength	—	kNN	sequential
IBEA [232]	DT	—	bin. indicator	—	—	bin. indicator	—	one-shot
HypE [13]	DT	—	I_H^h	—	depth	I_H^h	—	sequential
SMS-EMOA [19]	random	—	—	—	depth-rank	I_H^1	—	—

(All MOEAs above use *type* $(\texttt{pop}) =$ *fixed-size* and *type* $(\texttt{pop}_{\texttt{ext}}) =$ *none*; in addition, SMS-EMOA uses $\lambda = 1$)

3.1.4 Standard MOEAs instantiated via the AutoMOEA template

By carefully selecting the values of each algorithmic component, we can instantiate many well-known MOEAs from the literature using the proposed template. Table 3.4 shows how to instantiate the six MOEAs we consider in this work, which we have selected because of their relevance in the literature. In particular, we remark that, being one of the earliest MOEAs, MOGA does not use elitism, which can be implemented as a *generational* removal policy. Although we do not include this option for component Removal in our experiments with the framework, we use it in MOGA for fidelity to the original proposal. To ensure the correctness of our implementation, we have empirically verified that its performance matches the original implementations of the MOEAs provided by the authors (or when not available, by third-party ones). In the following experiments, we compare these six standard MOEAs with novel MOEAs instantiated from our template. Our analysis covers several scenarios, ranging from continuous to combinatorial, presented in Sections 3.2 and 3.3, respectively.

3.2 Automatically designing MOEAs for continuous problems

Our experimental investigation has two main goals. The first is to assess how automatically designed MOEAs (hereafter called AutoMOEAs) perform compared to several standard MOEAs that can be instantiated from our framework. Second, we want to investigate how much the structure of the AutoMOEAs vary depending on the benchmark and the number of objectives considered. The benchmark problems that have been considered at the design time of an algorithm may implicitly or explicitly bias the algorithm design. Here we study this effect by considering two different benchmark sets, the DTLZ set [61] and the WFG set [112], reviewed in Section 2.1.5. More precisely, we use the unconstrained DTLZ1–DTLZ7 problems from the former, and the WFG1–WFG9 problems from the latter. Following [112], we set the ratio between position and distance variables to 1/6. Each benchmark set is used with two, three and five objectives. We separate between different number of objectives as it is known before running an algorithm and, obviously, an algorithm configuration that performs well for a low number of objectives (e.g. 2 or 3) need not perform well for more objectives (e.g. 5). We then design AutoMOEAs for each of the six scenarios obtained from the combinations of benchmark set (DTLZ and WFG) and number of objectives (2, 3, and 5), as we discuss next.

3.2.1 AutoMOEA design setup

The parameter space we use for the automatic design of the MOEAs is given in Tables 3.1, 3.2 and 3.5, where p_c and $p_m \in [0, 1]$ respectively stand for the probability of applying crossover to a given pair of individuals, and the probability of applying mutation to a given individual. We use the SBX crossover operator and polynomial mutation, which have associated numerical parameters η_c and η_m (the distribution indices). Furthermore, different mutation schemes can be used by MOEAs for real-parameter

Table 3.5: Parameter space for tuning all MOEAs for continuous optimization.

Parameter	$\mu = \lvert\texttt{pop}\rvert$	$\lambda = \lvert\texttt{pop}_{\texttt{new}}\rvert$	p_c, p_m	η_c, η_m
Domain	$\{10, 20, \ldots, 100\}$	1 or $\lambda_r \cdot \mu$ $\lambda_r \in [0.1, 2]$	$[0, 1]$	$\{1, \ldots, 50\}$

Condition	**Additional parameter**	**Domain**
type $(\texttt{pop})$ = *bounded*	μ_r	$[0.1, 2]$
type $(\texttt{pop}_{\texttt{ext}})$ = *bounded*	N_{ext}	$\{100, 300, 500\}$
mutation scheme = *fixed*	p_v	$[0.01, 1]$
Selection = DT	*tournament size*	$\{2, 4, 8\}$
Selection = ST	γ	$[0.6, 0.9]$
Quality = *binary indicator*	*indicator*	$I_{\epsilon+}$, I_H^-
Diversity = *sharing*	σ_{share}	$[0.1, 1]$
Diversity = kNN (as part of Mating)	k_{method}	$\{default, k\}$ $k \in \{1, \ldots, 9\}$

optimization [58]. Here, we implement two options: (i) *bitwise*, which sets the mutation probability per variable such that on average one variable is mutated per individual chosen for mutation; and (ii) *fixed*, where the mutation probability per individual mutated is set by the user as a parameter $p_v \in [0.01, 1]$. We do not include more evolutionary operators and schemes to focus on the high-level multi-objective components that characterize MOEAs.

As the automatic offline parameter configuration tool we use irace [159], which has been adapted to handle multi-objective algorithms by using the hypervolume indicator as follows. First, the candidates generated by irace are given a maximum number of function evaluations (FE). Following [13], we set this number to 10 000 FEs. Then we assess the quality of the approximation fronts produced by each candidate by computing their hypervolume relative percentage deviation (I_H^{rpd}, see Section 2.1.4). To compute the I_H metric, we discard all solutions with objective values worse than the upper bound[1] $\mathbf{u} = [10]^M$, and use the reference point $\mathbf{r} = [11]^M$. Concerning the reference fronts we use in this work, we have initially generated 1 000 random Pareto optimal solutions for each problem size using the methodology described in the original papers where the benchmarks were proposed. However, we noticed that many of these initial sets presented poor hypervolume since solutions were not well spread. We improved these reference sets by adding nondominated solutions found by running traditional MOEAs for 100 000 FEs using their default parameters 10 times on each problem instance, for all problems where we identified this issue (WFG1–9 and DTLZ4).

Experiments are run on a single core of Intel Xeon E5410 CPUs, running at 2.33GHz with 6MB cache size under Cluster Rocks Linux version 6.0/CentOS 6.3. To keep our experiments feasible in time, we limit the maximum runtime of a single run to 10 minutes. For all configurations that correspond to the standard MOEAs, this time limit is high enough to perform all 10 000 FEs. In fact, we have empirically verified that, on average, 85% of the candidates produced by irace use all FEs allowed within this time limit. The few configurations that do not use all available FEs are typically the ones that combine many computationally costly components at once, e.g., an external archive replacement based on the shared hypervolume contribution, nearest neighbor diversity, and sequential removal. For these configurations, we assess their performance based on the approximation fronts they return when reaching the time limit. Given the large search space for designing the AutoMOEAs, we give irace a tuning budget of 20 000 runs. In our computational setup, the wall-clock time used by irace is equivalent to designing a MOEA over the weekend.

[1] The upper bound is used to prevent the effects of strong outliers.

3.2.2 Performance comparison setup

In the context of continuous optimization, benchmark sets try to be as heterogeneous as possible in order to capture as many potential features of unknown real-world problems as possible. As a consequence of how such benchmarks are designed, partitioning them into disjoint sets of functions would result in a training set for tuning that is not representative of the test set. Here, we go a step further of what is the standard in continuous optimization benchmarking, and the functions used in the tuning and test always differ in the number of variables (n_{var}). Concretely, we use $n_{\text{var}} \in \{20, \ldots, 60\} \setminus n_{\text{testing}}$ for tuning, where $n_{\text{testing}} = \{30, 40, 50\}$ are the sizes we reserve for testing. We also take the precaution of differentiating the effect of tuning the numerical parameters of MOEAs from the effect of designing novel MOEAs. Although the literature proposes default numerical parameters per benchmark [13, 19, 61, 112], we have found that major performance improvements can be achieved by tuning these numerical parameters[2]. Hence, in the remainder of the chapter, all standard MOEAs have been tuned for the corresponding scenario, using the same numerical parameter space as for AutoMOEA (Table 3.5)[3]. We prefer this approach of comparing standard MOEAs to the AutoMOEAs as there may be interactions between numerical and structural parameters that change the MOEA design and therefore transferring numerical parameter settings from one to another algorithm may bias the algorithm comparisons. For each of these tunings, we also give irace a tuning budget of 20 000 runs.

To compare different algorithms, we first run each algorithm 25 times on the testing benchmarks. In addition to the I_H^{rpd}, we also compute the additive ϵ-indicator ($I_{\epsilon+}$) of the approximation sets w.r.t. the reference fronts. The comparison is done visually by means of boxplots, and analytically through rank sums. To assess statistical significance, we adopt Friedman's non-parametric test and its associated post-hoc method at 99% confidence. For brevity, we omit the $I_{\epsilon+}$ results when they agree with the I_H^{rpd} ones. The full set of results is provided as supplementary material [28][4].

3.2.3 Results and discussion

The designs of the AutoMOEAs selected by irace for each of the scenarios we consider are shown in Table 3.6. All AutoMOEAs use replacement preference relations comprising set-partitioning and indicator-based components (very often the I_H^1), as well as large external archives. Surprisingly, the only exception to this pattern is $\text{AutoMOEA}_{\text{W5}}$, which does not use any set-partitioning metric for replacement. Concerning the external archive, the number of nondominated solutions in these problems is large, demanding an external archive, but prohibiting an unbounded one. In particular, most AutoMOEAs use a Preference_{Ext} that combines quality and diversity metrics, a combination that has been shown to work well in some cases [236]. One pattern we also observe in these external archives is that the exclusive hypervolume contribution (I_H^1) indicator is always coupled with sequential removal, while the remaining indicators are used with one shot replacement. This is likely explained by the increased computational overhead incurred by the computation of the hypervolume and our use of a maximum time limit.

Two other design choices have been frequently selected, namely steady-state replacement ($\lambda = 1$) and the BuildMatingPool component. Steady-state replacement has been shown to lead to effective results when runtime is not too limited [73]. As for BuildMatingPool, all AutoMOEAs use eight-ary deterministic tournament (except for $\text{AutoMOEA}_{\text{W3}}$ which uses four-ary tournaments), reflecting the need for convergence pressure that the problems demand. In addition, all MOEAs use crowding distance as diversity metric, the most extreme case being $\text{AutoMOEA}_{\text{D2}}$, which relies solely on this metric when selecting for mating.

---

[2]For brevity, this analysis is provided as supplementary material [28], together with the tuned configurations of all MOEAs.

[3]Configurations that would change the MOEA design as defined by Table 3.4 are not allowed when tuning numerical parameter settings.

[4]The supplementary material containing extended results of the original paper used as the basis for this chapter.

Table 3.6: Parameters selected by irace for the AutoMOEAs designed for continuous optimization problems.

	BuildMatingPool				Replacement				Replacement$_{Ext}$			Numerical							
	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	Quality	Diversity	Removal	μ	μ_r	λ	λ_r	p_c	p_m	η_c	η_v
DTLZ 2-obj	DT (8)	—	—	crowd.	DR	$I_{\epsilon+}$	sharing	—	—	crowd.	seq.	100	0.85	1	—	0.63	0.67	35	25
DTLZ 3-obj	DT (8)	DR	$I_{\epsilon+}$	kNN	rank	I_H^1	sharing	—	I_H^1	—	seq.	80	0.77	1	—	0.58	0.63	2	15
DTLZ 5-obj	DT (8)	rank	I_H^1	crowd.	depth	I_H^1	—	—	$I_{\epsilon+}$	crowd.	1-shot	40	—	1	—	0.35	0.62	42	5
WFG 2-obj	DT (8)	rank	—	crowd.	DR	I_H^1	—	—	I_H^h	crowd.	1-shot	20	—	1	—	0.11	0.33	31	11
WFG 3-obj	DT (4)	count	I_H^1	crowd.	strength	I_H^1	sharing	seq.	I_H^1	kNN	seq.	10	—	—	0.86	0.11	0.49	39	13
WFG 5-obj	DT (8)	count	I_H^h	crowd.	—	I_H^1	—	seq.	I_H^h	crowd.	1-shot	30	—	—	1.07	0.71	0.66	34	12

(All AutoMOEAs use the bitwise mutation scheme, and *type* ($\texttt{pop}_{\texttt{ext}}$) = *bounded* with $N_{\text{ext}} = 500$, except for AutoMOEA$_{\text{W2}}$, for which $N_{\text{ext}} = 300$. In addition, all but AutoMOEA$_{\text{D2}}$ and AutoMOEA$_{\text{D3}}$ use *type* (pop) = *fixed-size*, and all but AutoMOEA$_{\text{W3}}$ and AutoMOEA$_{\text{W5}}$ use steady-state replacement, i.e., $\lambda = 1$)

Table 3.7: Parameters selected by irace for the AutoMOEAs for combinatorial optimization problems. All designs use an internal archive instead of a regular population.

	BuildMatingPool				Replacement				*Numerical*					
	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	μ	μ_r	λ_r	p_c	p_{mut}	p_X
Cmax-TFT	DT (2)	—	I_H^1	crowding	—	$I_{\epsilon+}$	crowding	one-shot	80	0.3	1.5	0.38	0.82	0.71
Cmax-TT	random	—	—	—	—	I_H^h	crowding	one-shot	30	0.94	1.63	0.34	0.95	0.81
TFT-TT	ST (0.9)	—	—	crowding	—	$I_{\epsilon+}$	sharing (0.87)	sequential	70	0.94	1.47	0.77	0.99	0.63
Cmax-TFT-TT	DT (2)	—	—	crowding	—	I_H^h	crowding	one-shot	40	0.26	1.68	0.36	0.85	0.74
All variants (PFSP)	random	—	—	—	—	I_H^h	—	one-shot	60	0.73	1.53	0.17	0.76	0.40

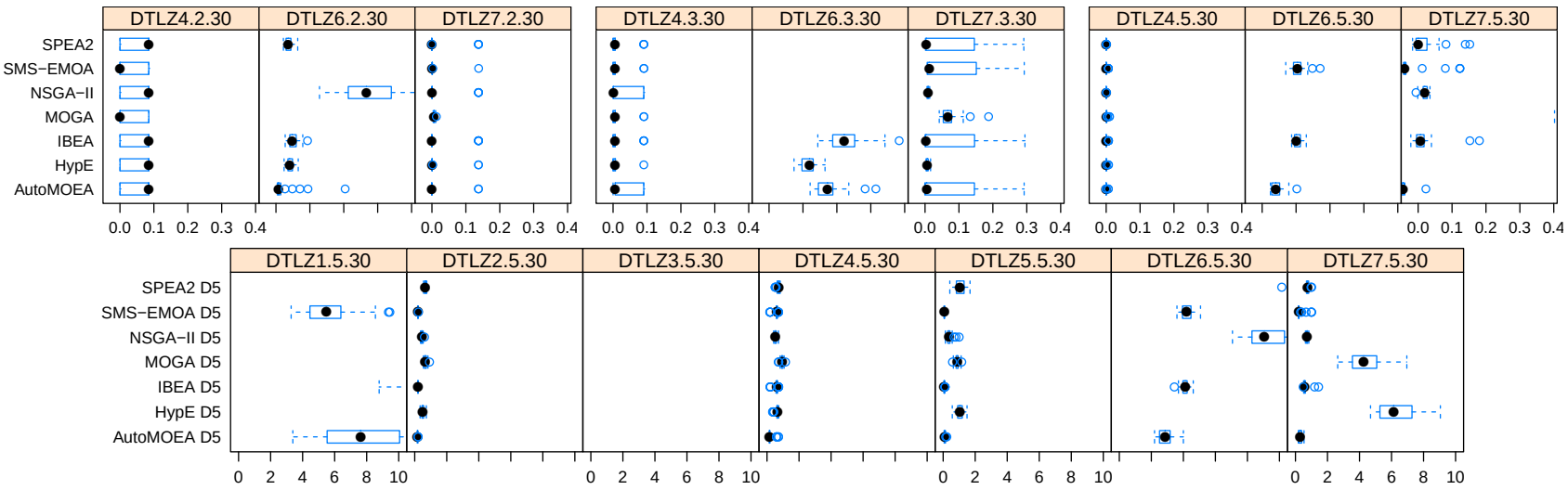


Figure 3.1: Performance boxplots for all algorithms on selected 30-variable DTLZ benchmark problems. Top: I_H^{rpd} for problems with 2, 3, and 5 objectives (from left to right, respectively). Bottom: $I_{\epsilon+}$ for 5-objective problems.

Despite these patterns, it is hard to establish general guidelines for selecting components when we consider a specific benchmark or a specific number of objectives. However, as we will discuss in more detail below, the I_H^{rpd} rank sum analysis given in Table 3.8 shows that each of these AutoMOEA variants perform very well on the scenarios for which they were designed. This result is consistent with our expectations that different scenarios should demand different components, and that the component-wise design proposed here provides enough flexibility to meet this need. Next, we discuss the performance of the algorithms for each of the benchmarks considered in detail.

DTLZ benchmark

Although this benchmark has been extensively used in the literature, most of the results can be considered novel, as the number of variables we use is larger than traditionally adopted. For DTLZ2 and DTLZ5, this increase in the number of variables is not enough to make these functions difficult, and all MOEAs find approximation sets with I_H^{rpd} very close to zero. Conversely, functions DTLZ1 and DTLZ3 become so difficult that no MOEA is able to find solutions within the bounds we set. The I_H^{rpd} of the remaining functions are shown in Figure 3.1 (top), and we examine them individually. We remark that we zoom these boxplots on the [0,0.4] range as this is the actual area of interest for this indicator. For brevity, we show only boxplots of the 30-variable functions, but we remark that the results for the other problem sizes are consistent with these ones.

DTLZ4 is a function that presents bias, and MOEAs are sometimes unable to find well-spread approximation fronts. This explains the variance we observe on the 2-objective boxplots. Still, algorithms like SMS and MOGA are able to perform well on most runs. Function DTLZ6 presents a different kind of bias, making it difficult for several MOEAs to converge to the actual fronts, specially as the number of objectives grows. This time the only MOEAs that maintain good performance in all scenarios are IBEA and the AutoMOEAs. Finally, DTLZ7 is a disconnected function that MOEAs are able to solve with two objectives, but that becomes much harder with five objectives. SMS and the AutoMOEAs are the only algorithms that present high performance in all scenarios, but once more the AutoMOEAs find approximation fronts with lower I_H^{rpd} more often than SMS. Overall, the rank sums achieved by the AutoMOEAs are much lower than those of the remaining algorithms as shown in Table 3.8, which considers all problems and all test sizes.

The results given by the $I_{\epsilon+}$ indicator are mostly consistent with the ones provided by the I_H^{rpd}, except for the 5-objective problems shown in Figure 3.1 (bottom). As discussed for the I_H^{rpd}, the performance of all MOEAs in DTLZ1 and DTLZ3 is so poor that solutions lie outside the boundaries we pre-established. Only SMS and $\text{AutoMOEA}_{\text{D5}}$ are able to reach results on DTLZ1 inside these boundaries. Concerning

Table 3.8: Sum of ranks depicting the performance of MOEAs on continuous sets according to the I_H^{rpd}. ΔR is the critical rank sum difference for Friedman's test with 99% confidence.

DTLZ			WFG		
2-obj $\Delta R = 126$	3-obj $\Delta R = 127$	5-obj $\Delta R = 107$	2-obj $\Delta R = 169$	3-obj $\Delta R = 130$	5-obj $\Delta R = 97$
AutoMOEA$_{D2}$ (1339)	**AutoMOEA$_{D3}$ (1500)**	**AutoMOEA$_{D5}$ (1002)**	**AutoMOEA$_{W2}$ (1692)**	**AutoMOEA$_{W3}$ (1375)**	**AutoMOEA$_{W5}$ (1170)**
SPEA2$_{D2}$ (1562)	IBEA$_{D3}$ (1719)	SMS$_{D5}$ (1550)	SPEA2$_{W2}$ (2097)	SMS$_{W3}$ (1796)	SMS$_{W5}$ (1567)
IBEA$_{D2}$ (1940)	SMS$_{D3}$ (1918)	IBEA$_{D5}$ (1867)	NSGA-II$_{W2}$ (2542)	IBEA$_{W3}$ (1843)	IBEA$_{W5}$ (1746)
NSGA-II$_{D2}$ (2143)	HypE$_{D3}$ (2019)	SPEA2$_{D5}$ (2345)	SMS$_{W2}$ (2621)	SPEA2$_{W3}$ (2600)	SPEA2$_{W5}$ (2747)
HypE$_{D2}$ (2338)	SPEA2$_{D3}$ (2164)	NSGA-II$_{D5}$ (2346)	IBEA$_{W2}$ (2777)	NSGA-II$_{W3}$ (3315)	NSGA-II$_{W5}$ (3029)
SMS$_{D2}$ (2406)	NSGA-II$_{D3}$ (2528)	HypE$_{D5}$ (2674)	HypE$_{W2}$ (2851)	HypE$_{W3}$ (3431)	MOGA$_{W5}$ (4268)
MOGA$_{D2}$ (2970)	MOGA$_{D3}$ (2851)	MOGA$_{D5}$ (2915)	MOGA$_{W2}$ (4320)	MOGA$_{W3}$ (4540)	HypE$_{W5}$ (4373)

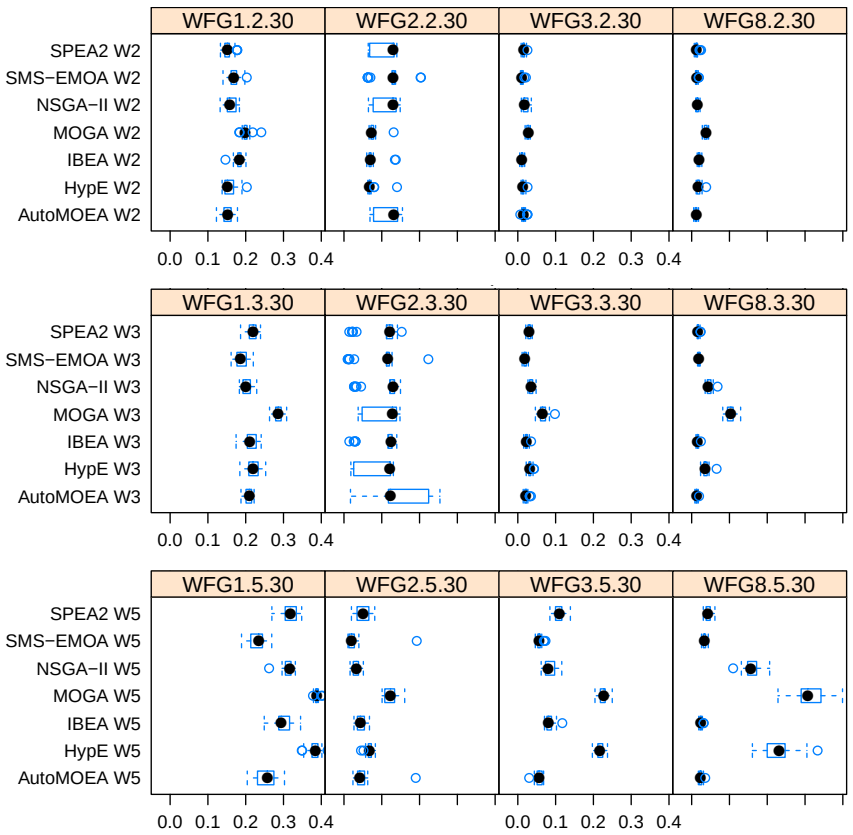


Figure 3.2: Hypervolume RPD on selected WFG benchmark problems with 30 variables. From top to bottom, 2, 3, and 5 objectives.

DTLZ2, DTLZ4, and DTLZ5, even though all MOEAs find approximation sets with I_H^{rpd} close to zero, the $I_{\epsilon+}$ tells us that only AutoMOEA$_{W5}$, SMS, and IBEA are able to converge to the actual fronts in functions DTLZ2 and DTLZ5, and only AutoMOEA$_{W5}$ in DTLZ4. Another function where the performance of the MOEAs is worse according to the $I_{\epsilon+}$ than according to the I_H^{rpd} is DTLZ6, which is explained by the difficulty of converging we have previously discussed. Finally, the general performance of all MOEAs according to the $I_{\epsilon+}$ for DTLZ7 are actually better than according to the I_H^{rpd} ones, which can be explained by the disconnectedness of this problem. Overall, the $I_{\epsilon+}$ rank sum analysis is consistent with the I_H^{rpd} analysis given in Table 3.8.

WFG benchmark

The performance of all MOEAs in the WFG problems is shown in Fig. 3.2. For brevity, we omit functions WFG4–WFG7 and WFG9 as we have noticed that the performance of all MOEAs is very similar on the WFG concave functions (WFG4–WFG9). We also remark that the boxplots of the I_H^{rpd} and the $I_{\epsilon+}$ indicators are very similar, and for this reason we omit the latter. Concerning the functions depicted in Fig. 3.2, one can clearly see a separation between WFG1 and WFG2 from the remaining functions. These two convex problems pose difficulties for MOEAs to converge regardless of the number of objectives. As for the other group of problems, MOEAs are able to perform well both on 2 and 3 objectives, with the exception of MOGA. Looking into the 5-objective problems in more detail, we notice that MOGA, NSGA-II, and HypE are unable to converge to the actual fronts, and so is SPEA2 for WFG3, WFG5, WFG7, and WFG9. IBEA, SMS, and $\mathsf{AutoMOEA}_{\mathsf{W5}}$ show the best performance on all problems except WFG3, where no MOEA is able to match the performance of $\mathsf{AutoMOEA}_{\mathsf{W5}}$.

The rank sum analysis of the I_H^{rpd} results given in Table 3.8 confirms that the AutoMOEAs designed for the WFG benchmark display the best performance among all MOEAs considered, and so does the rank sum analysis of the $I_{\epsilon+}$ indicator (see [28]). As for the remaining MOEAs, the rank sums of the algorithms are not consistent across different metrics, which indicates that some MOEAs favor convergence while others favor keeping a good trade-off between solutions (or are simply unable to find/preserve extreme solutions).

3.2.4 Experiments with a different stopping criterion

As shown by the results discussed above, standard MOEAs tend to perform better on the scenarios for which they have been properly tuned. Besides the benchmark set and the number of objectives considered, another major factor that affects the performance of algorithms is the stopping criterion used to terminate their runs. In continuous optimization, a maximum number of function evaluations (FE) is typically used because some applications present computationally costly FEs. As a result, algorithm designers tend to devise algorithms that are able to reach high-quality solutions with as few FEs as possible. Moreover, the time spent by the algorithms computing metrics or discarding solutions is not considered an issue in these scenarios and, hence, very fast and very slow algorithms are often considered equal. For instance, SMS-EMOA requires almost 10 minutes for executing 10 000 FEs in our computer environment, while IBEA terminates in seconds. However, in many practical situations the computational cost of the FEs may not be high enough to justify large computation times. In such scenarios, fast algorithms such as IBEA or NSGA-II could likely outperform slow ones such as SMS-EMOA by seeing many more solutions within a maximum runtime. By contrast, our design approach should be able to deal with such changes naturally. In this section, we investigate the structure of the AutoMOEAs generated and their performance relative to other MOEAs when all algorithms are given a maximum time limit of *one minute* CPU time. For brevity, we focus only on the AutoMOEAs designed and tested on the WFG benchmark for two, three and five objectives:

WFG, 2-objective (W2) $\mathsf{AutoMOEA}_{\mathsf{W2}}^{\mathsf{1min}}$ uses a intermediate population size, a high number of offspring, and a large external archive based on crowding. Mating selection relies solely on crowding distance, whereas replacement is done based on dominance depth and the $I_{\epsilon+}$ indicator. Both the external archive and the population removal are sequential. The structure of this algorithm is quite interesting because it combines computationally expensive components (large external archive with sequential removal) with computationally cheap ones (crowding distance, dominance depth, and the $I_{\epsilon+}$), adapting to the maximum runtime it is given. The I_H^{rpd} performance of $\mathsf{AutoMOEA}_{\mathsf{W2}}^{\mathsf{1min}}$ is shown in Fig. 3.3 (top left). We notice that the problem with most significant changes is WFG1, where the I_H^{rpd} of all algorithms is greatly improved. Although all MOEAs perform similarly, the rank sum analysis given in Table 3.9 indicates that $\mathsf{AutoMOEA}_{\mathsf{W2}}^{\mathsf{1min}}$ performs better on a larger num-

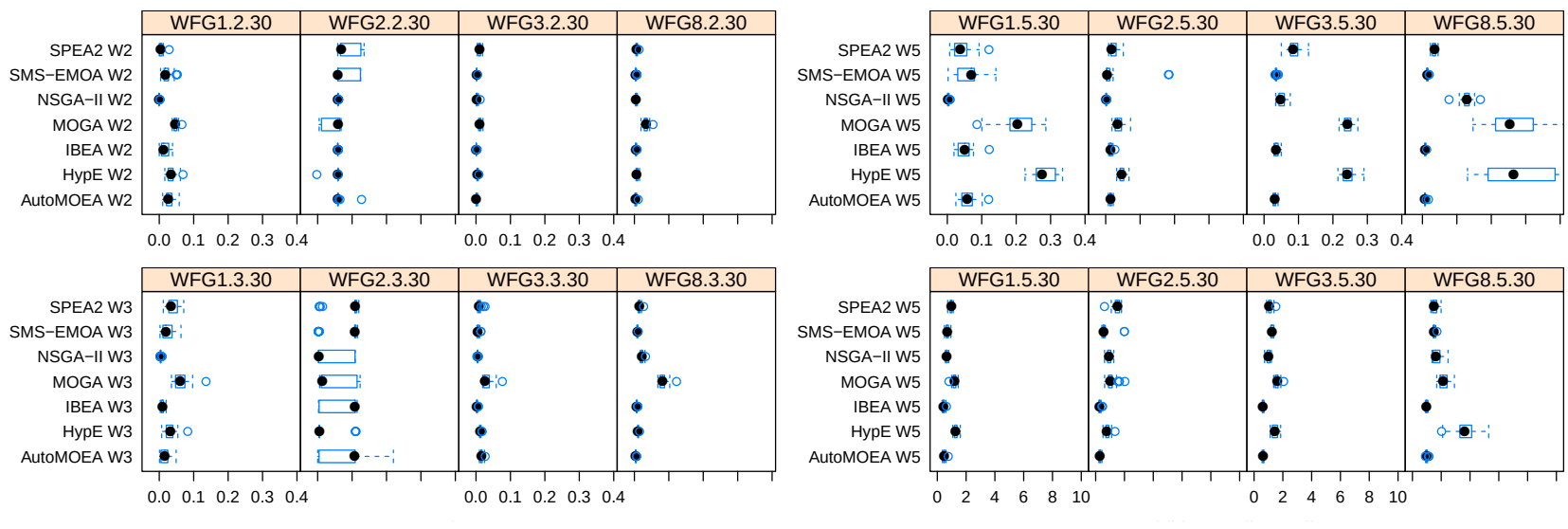


Figure 3.3: Performance of all MOEAs tuned for a maximum runtime on selected 30-variable WFG problems. On the top, I_H^{rpd} results for two (top) and three (bottom) objectives. On the right, I_H^{rpd} (top) and $I_{\epsilon+}$ (bottom) results for five objectives.

ber of problems, both according to the I_H^{rpd} and the $I_{\epsilon+}$. Concerning the remaining MOEAs, IBEA and NSGA-II rank equivalently according to the I_H^{rpd}, but IBEA performs better than NSGA-II more often according to the $I_{\epsilon+}$.

WFG, 3-objective (W3) $\mathsf{AutoMOEA}_{\mathsf{W3}}^{\mathsf{1min}}$ uses a large bounded internal archive, default number of offspring ($\lambda_r = 1.0$), and no external archive. The $\mathsf{Preference}_{Mat}$ component is based on nearest neighbor density, while $\mathsf{Preference}_{Rep}$ uses the exclusive hypervolume contribution (I_H^1) and sequential removal. This combination of components is coherent with this scenario, as the bounded internal archive and the I_H^1 provide convergence pressure at a low computational cost, and the nearest neighbor diversity has shown good results for SPEA2. Performance-wise, we see from Fig. 3.3 (bottom left) that again many algorithms present good performance according to the I_H^{rpd}. Overall, the rank sum analysis given in Table 3.9 indicates IBEA displays better I_H^{rpd} performance more often than $\mathsf{AutoMOEA}_{\mathsf{W3}}^{\mathsf{1min}}$, but the opposite happens for $I_{\epsilon+}$. This is actually surprising, since $\mathsf{AutoMOEA}_{\mathsf{W3}}^{\mathsf{1min}}$ has been tuned for the I_H^{rpd} and uses the I_H^1 indicator as its replacement mechanism, when IBEA uses the $I_{\epsilon+}$ instead.

WFG, 5-objective (W5) As the previous scenarios have indicated, IBEA is quite effective when facing a runtime-constrained scenario. The structure of $\mathsf{AutoMOEA}_{\mathsf{W5}}^{\mathsf{1min}}$ confirms this, as this algorithm presents the same exact components from IBEA, but can be considered a refinement of that algorithm as $\mathsf{AutoMOEA}_{\mathsf{W5}}^{\mathsf{1min}}$ uses crowding diversity both for mating and environmental selection. The similarity between these algorithms reflects on the boxplots shown in Fig. 3.3 (right-most plots), and is confirmed by the rank sums given in Table 3.9. We see that the crowding distance metric is unable to improve the $I_{\epsilon+}$ performance of a MOEA, but the I_H^{rpd} performance of $\mathsf{AutoMOEA}_{\mathsf{W5}}^{\mathsf{1min}}$ is greatly improved. Concerning the performance of the remaining algorithms on the non-concave WFG problem (WFG1–WFG3), we see a discrepancy between the I_H^{rpd} and the $I_{\epsilon+}$ results, as expected (see Section 2.1.4). This effect is greatly reduce on the concave functions, represented by WFG8. Concretely, the hypervolume-based SMS-EMOA and HypE present better performance on the I_H^{rpd} than on the $I_{\epsilon+}$, and so does SPEA2.

Overall, the results shown in this section have confirmed that the overhead incurred by MOEA components can greatly impair their efficiency when facing a problem that is not computationally expensive, but requires a constrained runtime. Next, we investigate the effect of the benchmarks used for tuning.

Table 3.9: Sum of ranks depicting the performance of MOEAs tuned for a maximum runtime. Algorithms in boldface present rank sums not significantly higher than the lowest ranked.

W2	I_H^{rpd}	**AutoMOEA$_{W2}$ (1700)**	NSGA-II (1909)	IBEA (1912)	SMS (2678)	SPEA2 (3082)	HypE (3360)	MOGA (4259)
	$I_{\epsilon+}$	**AutoMOEA$_{W2}$ (1402)**	IBEA (1868)	NSGA-II (2158)	HypE (2995)	SMS (3050)	SPEA2 (3190)	MOGA (4237)
W3	I_H^{rpd}	**IBEA (1363)**	AutoMOEA$_{W3}$ (1651)	SMS (2328)	HypE (2566)	NSGA-II (2986)	SPEA2 (3445)	MOGA (4561)
	$I_{\epsilon+}$	**AutoMOEA$_{W3}$ (1184)**	IBEA (1380)	SMS (2240)	SPEA2 (2959)	NSGA-II (3109)	HypE (3658)	MOGA (4369)
W5	I_H^{rpd}	**AutoMOEA$_{W5}$ (1192)**	IBEA (1446)	SMS (2072)	NSGA-II (2676)	SPEA2 (2857)	MOGA (4274)	HypE (4383)
	$I_{\epsilon+}$	**AutoMOEA$_{W5}$ (1052)**	**IBEA (1084)**	SMS (2717)	NSGA-II (2721)	SPEA2 (2932)	MOGA (4075)	HypE (4319)

(ΔR values from top to bottom: 157, 155, 136, 122, 102, 97)

Table 3.10: Sum of ranks depicting the performance of MOEAs for the cross-benchmark setup. ΔR is the critical rank sum difference for Friedman's test with 99% confidence.

DTLZ			WFG		
2-obj $\Delta R = 118$	3-obj $\Delta R = 126$	5-obj $\Delta R = 118$	2-obj $\Delta R = 165$	3-obj $\Delta R = 119$	5-obj $\Delta R = 118$
AutoMOEA$_{W2}$ (1142)	**AutoMOEA$_{W3}$ (1292)**	**AutoMOEA$_{W5}$ (1420)**	**SPEA2$_{D2}$ (1376)**	**AutoMOEA$_{D3}$ (1491)**	**AutoMOEA$_{D5}$ (1420)**
SPEA2$_{W2}$ (1692)	IBEA$_{W3}$ (1692)	**SMS$_{W5}$ (1485)**	NSGA-II$_{D2}$ (2334)	IBEA$_{D3}$ (1663)	**SMS$_{D5}$** (1485)
IBEA$_{W2}$ (1858)	SMS$_{W3}$ (1937)	IBEA$_{W5}$ (1774)	IBEA$_{D2}$ (2409)	SMS$_{D3}$ (1739)	IBEA$_{D5}$ (1774)
NSGA-II$_{W2}$ (1929)	SPEA2$_{W3}$ (2067)	NSGA-II$_{W5}$ (2279)	HypE$_{D2}$ (2666)	SPEA2$_{D3}$ (2395)	NSGA-II$_{D5}$ (2279)
SMS$_{W2}$ (2443)	NSGA-II$_{W3}$ (2451)	SPEA2$_{W5}$ (2291)	SMS$_{D2}$ (2904)	NSGA-II$_{D3}$ (3360)	SPEA2$_{D5}$ (2291)
MOGA$_{W2}$ (2791)	MOGA$_{W3}$ (2547)	HypE$_{W5}$ (2625)	AutoMOEA$_{D2}$ (2966)	HypE$_{D3}$ (3702)	HypE$_{D5}$ (2625)
HypE$_{W2}$ (2844)	HypE$_{W3}$ (2712)	MOGA$_{W5}$ (2824)	MOGA$_{D2}$ (4245)	MOGA$_{D3}$ (4550)	MOGA$_{D5}$ (2824)

3.2.5 Cross-benchmark setup

One may suspect that the better performance of the AutoMOEAs for the specific benchmark sets towards which they are tuned comes at the price of poorer performance on other benchmark sets. To examine whether this happens in our case, we applied the various MOEA algorithms tuned for one benchmark set to the respective other one, that is, the algorithms tuned on the WFG training set of functions to the DTLZ benchmark set and vice versa. We did this analysis only for the setup where MOEAs are given a maximum number of FEs to use, and we focus on the rank sum analysis of the I_H^{rpd}. The results of this analysis are given in Table 3.10. In most cases, the relative order among the algorithms remains very similar to the one encountered in Table 3.8. In five out of six cases the AutoMOEA algorithms remain the best performing ones, AutoMOEA$_{D2}$ being the only exception. The results for the $I_{\epsilon+}$ are consistent with these ones for all scenarios, despite minor differences. The full analysis of this cross-benchmark setup is provided as supplementary material [28].

3.2.6 Concluding remarks

The experiments conducted in this section have confirmed the importance of the automatic design methodology for developing MOEAs for continuous optimization, highlighting both its effectiveness and flexibility. Under all application scenarios and setups considered here, the AutoMOEAs were able to present a robust behavior and often outperform all standard MOEAs. At the same time, the performance of these standard MOEAs varied considerably. Although IBEA performed well on both setups we adopted, the AutoMOEAs designed here were able to consistently outperform it in the majority of cases.

3.3 Automatically designing combinatorial MOEAs

In this seciton, we apply the automatic MOEA design for tackling four multi-objective permutation flow shop problems (MO-PFSP), a well-known class of multi-objective combinatorial problems. Although many MOEAs have not been designed with combinatorial optimization problems in mind, many of the MOEAs we considered in Section 3.2 have been adapted to such problems using problem-specific variation operators [178]. Our results with the component-wise approach we propose here led to much better performance than standard MOEAs, and therefore we discuss the insights they produced.

As reviewed in Section 2.1.5, the multi-objective PFSP is a widely studied, practically relevant problem [69, 178] that is structurally different from continuous problems. Thus, we expected that the AutoMOEA designs for the MO-PFSP use different components when compared to those we have previously discussed in Section 3.2. We considered four variants of the MO-PFSP that combine the three most used objective functions from the PFSP literature, namely *makespan* (C_{max}), *total flow time* (TFT), and *total tardiness* (TT). In particular, we considered three bi-objective variants, C_{max}-TFT, C_{max}-TT, and TFT-TT, and the three-objective variant C_{max}-TFT-TT. All problems are NP-hard as already the underlying single-objective PFSPs are. We used irace to devise five AutoMOEAs: a variant specific for each of the four variants, and a general one that tackles all four MO-PFSP variants (PFSP).

3.3.1 Experimental setup

The experimental setup for the MO-PFSP experiments follows runtime-constrained scenarios as presented in Section 3.2 but with some differences to be mentioned. First, following [69], we allow algorithms to run for a maximum of $t = 0.1 \cdot n \cdot m$ seconds, where n and m are the number of jobs and machines, respectively. Second, the MO-PFSP literature has already shown that the number of nondominated solutions for these problems is low, and hence we run all algorithms with an unbounded external archive. Third, we use the regular I^H indicator instead of the previous metrics we used for continuous optimization. For tuning, irace was given a budget of 20 000 algorithm runs for designing the general $\text{AutoMOEA}_{\text{PFSP}}$, and 5 000 for designing each of the variant-specific AutoMOEAs and for tuning the standard MOEAs. The parameter space used for tuning the standard MOEAs and all AutoMOEAs is the same as presented in the previous section, except for the problem-dependent ones. The parameter space used for the numerical parameters is the same from the previous section, except for the PFSP-specific evolutionary operators. In particular, all MOEAs use random initial solutions, two-point crossover[5] and two problem-specific mutation operators, namely *insert* and *exchange*. These variators have associated numerical parameters tuned using the $[0, 1]$ domain, namely: (i) p_c, the probability of applying crossover to a given pair of individuals; (ii) p_m, the probability of applying mutation to a given individual, and; (iii) p_X the probability of using the exchange operator if mutation is performed[6]. For testing, each algorithm was run 10 times on each test instance, and the results presented here are the average hypervolume over these 10 runs. Following [69], all test instances are different from the instances used in the tuning. In addition,

[5] Used alongside a repair procedure to ensure the feasibility of the offspring solutions.
[6] Effectively, the insertion operator is applied with probability $1 - p_X$.

the testing set considers instances with 5, 10, and 20 machines, while the tuning set uses only instances with 20 machines. Next, we discuss the main experimental results and insights from this analysis.

3.3.2 Experimental results and discussion

The goals of this section are to (i) analyze the design automatically selected for the different variants of the PFSP and for the general version; (ii) to compare these designs with the previously discussed AutoMOEA designs for continuous optimization, and; (iii) to assess the performance of the PFSP AutoMOEAs. We next address each of these goals.

AutoMOEA designs for the PFSP. The tuned designs of the AutoMOEAs are shown in Table 3.7, and present two commonalities. First, all AutoMOEAs use a *bounded internal archive*, which reflects the need for convergence pressure for solving combinatorial problems such as the MO-PFSP. Second, all algorithms use a high value for the offspring factor (λ_r), which is always larger than 1.4. This is explained by the time-constrained setup used for combinatorial optimization since the number of offspring per generation influences the trade-off between the number of solutions seen versus the time spent computing metrics [26]. More precisely, if an algorithm produces too few solutions per generation, it will spend most of its time computing selection metrics rather than evaluating new solutions. Conversely, the number of offspring cannot be set to a very high value or this would reduce the number of generations, hindering the evolutionary process.

Besides these two components that are used by all AutoMOEAs, we also highlight other design choices that were often selected by irace. For mating, the crowding diversity operator was used by three of the five AutoMOEAs. Interestingly, the more general $\text{AutoMOEA}_{\text{PFSP}}$ uses the same BuildMatingPool components from $\text{AutoMOEA}_{\text{Cmax-TT}}$. Also, the only design that uses a quality indicator in component Preference_{Mat} is $\text{AutoMOEA}_{\text{Cmax-TFT}}$. For replacement, three components are again used by most of the designs: the binary ϵ-indicator or the shared hypervolume contribution as quality indicators, crowding distance as diversity metric, and one-shot removal. In fact, the only design that contradicts this pattern is $\text{AutoMOEA}_{\text{TFT-TT}}$, which uses fitness sharing for diversity and sequential removal. Concerning numerical parameters, an interesting (and apparently contradictory) observation is the fact that all variant-specific AutoMOEA designs favor the exchange mutation operator ($p_X > 0.5$), but $\text{AutoMOEA}_{\text{PFSP}}$ favors the insertion mutation operator. This is likely explained by the different tuning budgets used, since larger budgets are particularly beneficial for fine-tuning numerical parameters.

Design comparison with previous AutoMOEAs. When compared to the AutoMOEAs devised for continuous optimization, we noticed that most design choices differ considerably with the switch in application domain. For instance, while all continuous AutoMOEAs use deterministic tournament for mating, three out of five AutoMOEAs for the MO-PFSP use some form of randomized selection. In addition, nearly none of the AutoMOEAs designed for the MO-PFSP use quality metrics in Preference_{Mat}, while almost all continuous AutoMOEAs did. We do remark, though, the number of AutoMOEAs that use crowding diversity, both for continuous and combinatorial domains. Concerning component Replacement, we first remark that all PFSP use a bounded internal archive and none are steady-state, the opposite of what often happened with the continuous AutoMOEAs. As for Preference_{Rep}, both continuous and combinatorial AutoMOEAs consider dominance, quality metrics, and (very often) the crowding distance diversity metric. Finally, if we compare the external archive removal policy used by continuous AutoMOEAs with the internal archive ones used by the PFSP AutoMOEAs, we notice a much clearer pattern here, pointing to the effectiveness of the one shot policy for the PFSP.

Table 3.11: Sum of ranks depicting the overall performance of MOEAs on combinatorial problems. All algorithms have been tuned for the scenarios considered. ΔR is the critical rank sum difference for Friedman's test with 99% confidence. Algorithms in boldface present rank sums not significantly higher than the lowest ranked. CTT stands for variant Cmax-TFT-TT.

Cmax-TFT $\Delta R = 68$	**AutoMOEA PFSP (249)**	**AutoMOEA Cmax-TFT (301)**	IBEA (398)	NSGA-II (472)	SPEA2 (479)	HypE (585)	MOGA (687)	SMS (788)
Cmax-TT $\Delta R = 55$	**AutoMOEA Cmax-TT (209)**	**AutoMOEA PFSP (253)**	NSGA-II (357)	SPEA2 (464)	IBEA (547)	HypE (574)	SMS (770)	MOGA (786)
TFT-TT $\Delta R = 85$	**MOGA (304)**	**IBEA (371)**	AutoMOEA TFT-TT (475)	HypE (499)	NSGA-II (499)	AutoMOEA PFSP (553)	SPEA2 (615)	SMS (644)
CTT $\Delta R = 55$	**AutoMOEA Cmax-TFT-TT (161)**	AutoMOEA PFSP (251)	IBEA (417)	SPEA2 (525)	HypE (528)	NSGA-II (541)	SMS (735)	MOGA (802)

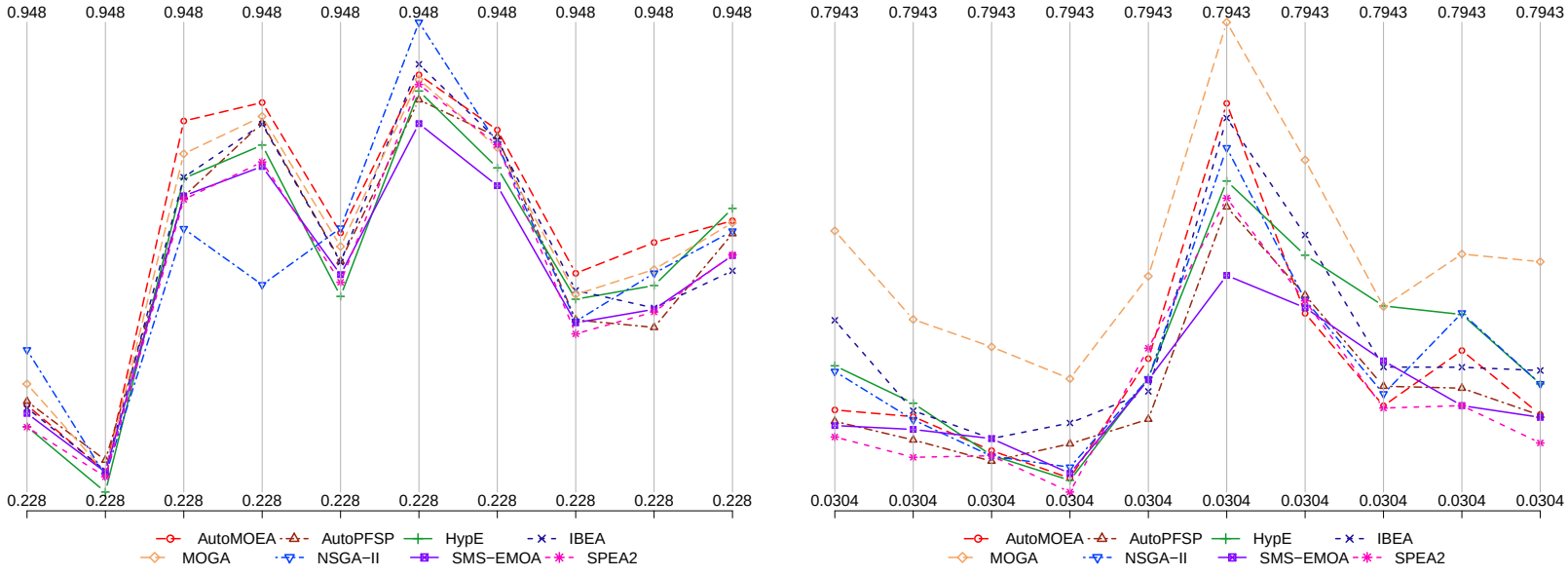


Figure 3.4: Mean hypervolume on 10 instances of TFT-TT. Left: 50 jobs, 20 machines. Right: 200 jobs, 20 machines. Each vertical line depicts one instance.

Overall performance comparison We analyzed the performance of all algorithms and variants using the rank sum analysis given in Table 3.11. In general, the AutoMOEAs perform much better than the standard MOEAs. Among these, IBEA is the algorithm that most often outperforms the others, while SMS-EMOA presents a particularly poor performance. The only scenario that contradicts this pattern is TFT-TT. Regarding the comparison between the variant-specific and the general AutoMOEAs, for most of the variants considered these two algorithms can be qualified as equally good. The difference in their rank sums is due to the fact that $\text{AutoMOEA}_{\text{PFSP}}$ performs particularly well for the 20-machine instances (the size used for the tuning), but for the remaining instances the variant-specific AutoMOEAs consistently outperform it. This indicates that the representativeness of the tuning set with respect to the test set is limited due to the different instance sizes present on each. The only variants where results differ slightly from this pattern are C_{max}-TFT, where $\text{AutoMOEA}_{\text{PFSP}}$ obtains a lower rank sum than $\text{AutoMOEA}_{\text{Cmax-TFT}}$, and C_{max}-TFT-TT, where the performance of $\text{AutoMOEA}_{\text{Cmax-TFT-TT}}$ is statistically significantly better than that of $\text{AutoMOEA}_{\text{PFSP}}$.

Concerning the TFT-TT, it is a problem with an increasing objective correlation as instance sizes grow (see Section 2.1.5). As a result, for large instances the number of nondominated solutions becomes particularly small and MOEAs sometimes return non-dominated sets with very few solutions. This heterogeneity of the testing set limits the representativeness of the tuning set affecting irace, as it cannot select a configuration that performs well across the whole set. To illustrate, we show the performance of all algorithms on two sets of 20-machine instances in Fig. 3.4. For smaller instance sizes like the ones given in Fig. 3.4 (left), all algorithms perform similarly, with $\text{AutoMOEA}_{\text{TFT-TT}}$ performing best in several instances. For the larger instance sizes given in Fig. 3.4 (right), how-

ever, all algorithms perform poorly, but MOGA is the one clearly less affected. When we limit the rank sum analysis to the testing instances with 20 machines, no significant difference is found for MOGA, AutoMOEA$_{\text{TFT-TT}}$, IBEA, and NSGA-II, the algorithms that present better performance.

3.3.3 Concluding remarks

As seen in this section, the designs of the AutoMOEAs devised for the PFSP differed in many aspects from those devised for continuous optimization problems. Nonetheless, the performance of the AutoMOEAs proves the efficacy of the automatic MOEA design also for combinatorial optimization. This further highlights the importance of having a flexible and representative MOEA framework. The good performance of the variant-specific AutoMOEAs reinforce this, showing that designing MOEAs for specific problem variants can lead to promising improvements. A comparison to a current state-of-the-art algorithm for the three bi-objective PFSPs is given in the supplementary material for interested readers [28]. However, reaching state-of-the-art results would require AutoMOEAs to incorporate fine-tuned local search algorithms [69, 178], which is beyond the scope of this chapter.

3.4 Conclusions

In this chapter, we have proposed a novel conceptual view of MOEAs to improve the way such algorithms are designed or applied to new scenarios. By considering MOEAs as combinations of lower-level components, such as preference relations and archives, our approach allows tailoring algorithms according to the characteristics of the target application. We have empirically demonstrated the efficacy of this component-wise view by automatically designing novel, efficient MOEAs for several application scenarios comprising continuous and combinatorial optimization problems. Concretely, we have proposed a flexible framework that extends both the number of algorithms that can be instantiated from a single template and the number of novel MOEA designs that can be produced from it. To navigate this large design space, we have used an offline parameter configuration tool, irace, following similar research work on other multi-objective metaheuristics [157]. An important focus of our work is the generalization aspect of the automatic design process. More precisely, a critical attribute of automatic configuration is the separation between training and testing scenarios, as highlighted in Section 2.3.2. In the context of continuous optimization, we have implemented this separation by using different dimensions of the functions within a benchmark set for training and testing, and additionally by conducting cross-benchmark experiments. For the MO-PFSPs studied, we have training instances that are different from the testing instances.

The applications of the conceptual view we propose here are numerous. First, by designing novel MOEAs to specific problem classes, one could identify particularly effective components for a given class. For instance, continuous benchmark sets often include disconnected problems. Using the methodology proposed here, one could create a benchmark set of disconnected problems and analyze AutoMOEAs specifically devised for this benchmark. Moreover, by comparing designs devised for several characteristic-driven benchmark sets like this one, patterns could likely be found, helping in future MOEA design when the characteristics of a given application are known in advance. A second and promising application is to couple the component-wise and/or the automatic design approaches with iterative design-space analysis tools such as ablation analysis [76]. This method generates intermediate configurations between pairs of algorithm designs, and hence can provide important insights about the contribution of individual components. For instance, Appendix A describes a preliminary work in this direction, where we consider several component-wise abstractions to compare different MOEAs on the PFSP. Another example of integrating ablation and our proposed approach would be to ablate between the continuous optimization AutoMOEAs devised for three and five objectives, and see how the intermediate configurations perform in these scenarios. One could then possibly reduce the number of components used by some of these

AutoMOEAs and understand better why they work so well. In fact, we conduct such an investigation in Chapter 5.

In the next chapters, we extend this work in order to automatically devise state-of-the-art MOEAs for continuous optimization. For starters, the next chapter describes our effort to identify the current state-of-the-art for this field, even though we reckon that the work presented in this thesis is still a first step towards this goal. In particular, although we are strong advocates of the component-wise methodology, the most urgent approach to such an experimental assessment requires a different component-wise approach to MOEAs, where the multi-objective components are considered a monolithic unit coupled with different underlying EAs, as we will discuss. Nonetheless, we attempt to analyze individual multi-objective components of the MOEAs to a certain extent, and an in-depth component-wise analysis for continuous optimization is one of the main contributions of the last chapter.

CHAPTER 4

A comprehensive performance assessment

The literature on multi-objective evolutionary algorithms (MOEAs) is extensive and MOEAs are one of the most wide-spread approaches to tackle multi-objective optimization problems (MOPs). Over the decades of MOEA research, a great number of algorithms were proposed in the literature, as reviewed in various surveys [46, 50, 149, 176] and also in Section 2.2.4. More recently, the interest in problems with (many) more than three objectives has raised, making the study of algorithms for *many-objective* problems (MaOPs) one of the currently most active fields within MOEA research in general [3]. Besides algorithmic advances, the MOEA community has been a driving force in the advancement of the performance assessment of multi-objective algorithms [61, 140, 233, 235]. In particular, those works were instrumental to help shape the research on MOEAs, defining problem benchmarks, performance metrics, and experimental setups to be reused in most of the subsequent MOEA research. However, the rate with which new MOEAs were proposed and the advancements related to the performance assessment of such algorithms is not well matched by the extent of the experimental analyses performed. In particular, no work has clearly identified the state-of-the-art for continuous MOPs, the default application domain used for evaluating novel MOEAs. As a result, many of the algorithms proposed in the literature were only compared to a handful of other MOEAs believed to be high-performing.

Besides the algorithm-wise developments, a large-scale experimental analysis should consider other experimental factors. Although some of these factors were addressed in the previous chapter, the analysis presented in this chapter is even more rigorous given our approaches to different stopping criteria, performance metrics, and underlying EAs. In particular, we consider a large set of the most prominent and diverse MOEAs proposed in the literature (see more details below). Considering other factors, we use a benchmark set comprising both the DTLZ and WFG benchmarks, with different numbers of variables (20 to 60), objectives (2, 3, 5, and 10), and stopping criteria (2 500, 10 000, and 40 000 FEs). In particular, the three different levels of FE budgets allow us to simulate different levels of computational overhead posed by function evaluations, as we explain later. In addition, we use an automatic configuration tool to properly tune the numerical parameters of each MOEA on each of the setups we consider. Finally, to evaluate the approximation sets produced by MOEAs, we consider a set of diverse performance metrics (I_H^{rpd}, $I_{\epsilon+}^1$ and I_{IGD}).

In addition to all the experimental factors discussed above, we consider MOEAs from a component-wise perspective by separating between the high-level algorithmic components specifically related to

multi-objective optimization (MO), and the traditional algorithmic components related to search in optimization problems, i.e., the underlying evolutionary algorithm (EA). We do so for two main reasons. First, clearly identifying the underlying EA used by a MOEA makes assessments fair, preventing a common mistake of comparing MOEAs that use different underlying EAs. Second, many of the papers where MOEAs were proposed concentrated solely on the MO components, regarding that the underlying EA would play a less important role. Later, when differential evolution (DE)-based versions of existing MOEAs were proposed [143, 201, 216, 219], some researchers assumed that it would be the best underlying EA choice for any MOEA. However, in recent papers [30, 31] and also in Appendix C, we show that this is not necessarily the case, with interactions between MO components and underlying EAs playing a critical performance role.

In this work, we select the group of MO components we consider the most relevant from the literature, namely the ones proposed for MOGA [80], NSGA-II [60], SPEA2 [234], IBEA [232], SMS [19], HypE [13], MOEA/D [150], and NSGA-III [59], and the two EAs most commonly used in the MOEA literature, i.e., genetic algorithms (GA) and differential evolution (DE). In particular, these MO components stem from different MOEA design paradigms and, thus, provide a comprehensive set of possibilities. To ensure that the most fit underlying EA is selected for the scenario considered, we use the automatic configuration tool to make the best choice. In addition, we include in our comparison a MOEA that uses neither of the underlying EA choices previously described, but that is very relevant in the literature, MO-CMA-ES [119]. In total, our analysis encompasses 15 algorithms independently proposed in the literature [13, 19, 59, 60, 80, 119, 143, 150, 201, 228, 232, 234].

Several relevant and interesting insights are drawn from the experimental results. First and foremost, we notice a pattern between results on two-, three- and five- objective problems, where the indicator-based SMS and IBEA perform best. By contrast, on ten-objective problems no MOEA is able to rank first according to all metrics, reinforcing the importance of using a diverse set of metrics in an assessment of many-objective optimization algorithms. In particular, NSGA-III and MOEA/D rank first according to given metrics, and IBEA is the only MOEA to be consistently good according to all metrics. Second, we show that some algorithms proposed more recently are unable to improve over their immediate predecessors. In fact, except for ten-objective problems, NSGA-III is one such example, with NSGA-II often presenting equivalent or better performance than it on the other experimental scenarios. We present additional investigation to explain these results, such as the effects of properly tuning algorithms before evaluating them. Finally, we notice that the difficulty levels of the problems considered depend on interactions between their structural characteristics and the increase in the number of objectives. We then conduct a brief follow-up investigation on the most important features these problems present.

The main contribution of this chapter can then be summarized as being a concrete first step towards identifying the state-of-the-art MOEAs for continuous optimization on two-, three-, five- and ten-objective problems, while producing a number of important insights, such as the impact of problem characteristics, underlying EA choices, and the remaining experimental factors considered. However, we also highlight a second important contribution concerning the experimental data produced. In particular, the number of insights that could be drawn from the campaign conducted in this work far extends the analysis we are able to fit in a chapter. In fact, this data is likely too large to be mined by a single research group, and for this reason we make it available to the MOEA community. Our goal is to stir a collective effort that can help draw further insights, either due to manual statistical analysis or to automated techniques. In particular, we strongly believe that this assessment can be a first step towards the application of machine learning approaches to better understand the nuances of many-objective optimization that have not yet been fully grasped.

The remainder of this chapter is organized as follows. In Section 4.1, we highlight the particular MOEAs we use in this study grouped by the design paradigm to which they belong. We also discuss the conceptual separation we adopt in this thesis. Next, we detail our performance assessment setup in Section 4.2, detailing how our work extends over others. In Sections 4.3 to 4.5, we present our

experimental results, starting by an overall analysis and moving on to further detailed insights. Finally, we conclude and discuss future work in Section 4.6.

4.1 MOEAs and underlying EAs

As discussed in Chapter 2, MOEAs have been traditionally categorized in the literature into one of the three following families (or design paradigms). To make our assessment comprehensive, we select at least two algorithms from each paradigm, which we highlight in boldface.

1. **Pareto-based approaches.** The best known examples from Pareto-based approaches are **NSGA-II** [60] and **SPEA2** [234]. In addition, we also consider **MOGA** [80], due to its historical relevance and as a baseline to put other results in perspective.

2. **Indicator-based approaches.** We consider the three most relevant MOEAs from this paradigm, namely **IBEA** [232], **SMS** [19], and **HypE** [13]. In particular, we remark that HypE was the first indicator-based algorithm designed specifically for MaOPs.

3. **Decomposition-based approaches.** To represent the decomposition-based paradigm we consider **MOEA/D** and **NSGA-III**. In particular, we consider the MOEA/D version that won the IEEE CEC 2009 competition on unconstrained multi-objective continuous optimization [150], which uses a *dynamic resource allocation* (DRA) approach. Concerning NSGA-III, we remark it was also proposed specifically for MaOPs.

More importantly, one feature that is typically overlooked when designing and comparing MOEAs concerns the underlying evolutionary algorithm used. As previously discussed, MOEAs can be seen as a combination of high-level MO components, responsible for dealing with the particular aspects of multi-objective optimization such as dominance, convergence, and diversity, coupled with an underlying EA responsible for selection and variation operators. In general, MOEA authors have focused their efforts on devising effective MO components, and reused the same underlying EA choices traditionally adopted in the literature, i.e., GAs or evolution strategies (ESs) typically using SBX crossover and polynomial mutation. As a result, the literature exploring different underlying EAs is limited, and the role variation operators of the underlying EAs have on performance is still rather poorly understood for both multi- and many-objective optimization. Below, we discuss some works that have tried to extend MOEA research in this direction:

Differential evolution (DE). During the maturing years of the MOEA design evolution, a few proposals using differential evolution as underlying EA arised, however, often re-using existing approaches as to MO components. For instance, many of the first DE-based MOEAs were fairly similar to NSGA-II concerning mechanisms to deal with convergence and diversity [1, 143, 166, 201]. The positive results by these DE-based MOEAs, however, motivated researchers to test other existing MO components with DE as underlying EA [219]. Table 4.1 shows a set of algorithms that are essentially equal as to their MO components, but differ mainly as to the underlying EA. We remark that not all of these algorithms pairs had been previously recognized as equivalent (e.g., SMS and IBDE [216]), most likely because of the different ways they were described by their authors.

In this work, we consider that MOEAs that differ only as to the underlying EA should be treated as one. More precisely, a given MOEA such as SPEA2 might use GA or DE as underlying EA depending on the performance they display on the experimental scenario considered. The choice of which underlying EA to use is made by the automatic algorithm configurator at the same time it tunes numerical parameters. Furthermore, if the configurator selects GA as underlying EA for a given algorithm, the SBX crossover and polynomial mutation operators are used. By

Table 4.1: MOEAs that differ only as to the underlying EA used.

	Underlying EA	
MO components	*GA/ES*	*DE*
dominance depth *crowding distance*	NSGA-II [60]	DEMO [201], GDE3 [143]
dominance strength *k-nn density estimator*	SPEA2 [234]	DEMO^{SP2} [219]
binary indicator	IBEA [232]	DEMO^{IB} [219]
dominance depth *hypervolume contribution*	SMS [19]	IBDE [216]

contrast, when the configurator selects DE as underlying EA, the differential mutation operator and binomial crossover are used by the MOEA. By doing so, we also prevent comparing algorithms that use different underlying EAs without ensuring that these are the best configurations for each MOEA. In addition, we remark that some algorithms such as MOGA, HypE, and NSGA-III had not yet been investigated coupled with DE, but we extend this EA choice also to these MOEAs.

Covariance matrix adaptation (CMA-ES). Although it would be possible to use CMA-ES as an underlying EA option for all MOEAs, as done with DE, we choose not to do so. Our main motivation is that there is currently no other work in the literature that considers CMA-ES as an underlying EA for MOEAs, and the goal of this chapter is not to propose novel algorithms. Furthermore, adapting the learning mechanism from CMA-ES to be coupled with other MO components is not a trivial task but rather a major research contribution in itself. Instead, in this work we represent this underlying EA using the **MO-CMA-ES** algorithm, and we model its different variants as parameters to be tuned, as explained in Section 4.2.3.

4.2 Assessment setup

In this section, we first explain the experimental factors we consider. Next we present the setup and the parameter space used for the automatic configuration process, followed by the testing setup we adopt. Finally, we discuss how our assessment differs from other existing analysis. We remark that all our experiments are run on a single core of Intel Xeon E5410 CPUs, running at 2.33GHz with 6MB cache size under Cluster Rocks Linux version 6.2/CentOS 6.2. In addition, MOEAs were implemented in C++ and compiled using gcc version 4.7.2 and the O3 optimization flag. When official implementations were not available, we verified our implementation against results from the papers where algorithms were originally proposed.

4.2.1 Experimental factors

To make our performance assessment as complete as possible, we consider several factors, detailed below.

Benchmark problems. In this work, we consider the 16 box-constrained functions that comprise the DTLZ [61] and WFG [112] benchmarks. We do not include the CEC 2009 competition benchmark [229] as one cannot scale the number of objectives presented by these functions.

Number of objectives (M). We consider four different values for the number of objectives, namely $M \in \{2, 3, 5, 10\}$. This allows us to conduct in-depth analysis for each of the values considered as well as to observe trends that arise as the number of objectives increases significantly. In fact, we choose to represent MaOPs using five and ten objectives to analyze the effect of this large increase

in the number of conflicting objectives. More precisely, we strongly believe that MOEAs will have considerable difficulty dealing with the large dominance resistance effect on MaOPs with ten objectives. Differently, based on the experiments from the previous chapter we believe that some MOEAs not specifically designed for MaOPs will be able to cope with the dominance resistance of five-objective problems.

FE computational cost. Traditionally, MOEAs have been designed and tested having as stopping criterion a maximum number of function evaluations. The rationale behind using this approach rather than actual computation times is that real-world problems (and specially MaOPs) present computationally and/or financially expensive to compute function evaluations, which is much higher than the time required by the algorithm itself. We see three potential drawbacks with this approach. First, many MOEA designers have created algorithmic components that present high computational complexity, considering that the overhead posed by such components would be compensated by the overhead presented by the function evaluations. In the literature, however, many of the assessments conducted have used a very large number of function evaluations, contradicting this basic assumption [61, 112, 229]. Second, some MOEAs present a very aggressive intensification behavior at early evolution stages under the assumption that not many function evaluations would be available. However, the optimization literature is rich on examples of drawbacks caused by such behavior [109], in particular the propensity of getting stuck in locally optimal solutions (in this case, local Pareto fronts). Finally, changes to this stopping criterion favor MOEAs in different levels, as we have demonstrated in the previous chapter.

In this work, we simulate different computational cost levels by using several budgets, i.e., different maximum numbers of function evaluations given to each run ($FE_{\max}$). Concretely, we consider $FE_{\max} \in \{2500, 10000, 40000\}$, which respectively represent function evaluations that present high, medium, and low computational cost. More precisely, our experimental scenarios are defined as combinations of M and $FE_{\max}$. This way, although the same benchmark functions are used in all experiments, scenarios where MOEAs are given a low number of function evaluations ($FE_{\max} = 2\,500$) represent real-world problems that are expensive to compute, and hence not many evaluations can be used. By contrast, scenarios where $FE_{\max}$ is high ($FE_{\max} = 40\,000$) represent real-world problems for which function evaluations are relatively cheap to compute. By doing so, we are able to analyze the performance of the MOEAs in all these different situations, assessing their flexibility and scalabity w.r.t. the stopping criterion.

Problem sizes. An experimental factor often not considered in the MOEA literature is the problem size, i.e. the number of variables present in the benchmark problems. For instance, most of the studies that use the DTLZ benchmark attain to the original number of variables suggested by the benchmark proposers, which can be considerably low ($n_{\text{var}} = 7$ for the three-objective DTLZ1). In this work, we also assess MOEAs as to their scalability w.r.t. problem sizes (n_{var}), considering the range $n_{\text{var}} \in \{20, 21, \ldots, 60\}$.

4.2.2 Tuning setup

Since we consider a number of different experimental factors, re-using numerical parameters proposed in other works for the MOEAs or even from the experiments conducted in the previous chapter is not advisable. Instead, we use irace to determine optimized parameter settings for each scenario. Our reasoning is that the number of objectives and the number of feasible function evaluations are usually known in advance before executing the algorithm, and that they may demand very different numerical parameter settings.[1] For instance, MOEA designers have traditionally tried to devise effective algorithms

[1] Even if the precise setting for a scenario is only available short time before actually executing the algorithm, one can easily imagine that MOEAs may have been previously tuned for many different scenarios and tuned settings are available

Table 4.2: Parameter space concerning variation operators for all MOEAs but MO-CMA-ES.

Underlying	GA			DE	
Parameter	t_{size}	p_c, p_m, p_v	η_c, η_m	CR	F
Domain	{2,4,8}	$[0, 1]$	$\{1, \ldots, 50\}$	$[0.01, 1]$	$[0.1, 2]$

regardless of the number of objectives presented by the problem. Based on the current insights available in the literature, however, we believe that such a goal is too high to be attained. The same reasoning goes for $FE_{\max}$, as previously discussed.

For fairness, we use a common tuning setup for all algorithms and scenarios. The configurator is given a maximum budget of 20 000 experiments per scenario, i.e. executions of the MOEA to be tuned. In particular, we remark that in this chapter the tuning targets both DTLZ and WFG benchmarks altogether, i.e., a single set of parameters is produced for each scenario. Primarily, each candidate configuration is allowed to run until it has used all given FEs. However, we set cutoff times ($t_{\max}$) for each run to keep the feasibility of the study. For scenarios where $FE_{\max} \in \{10000, 40000\}$, $t_{\max}$ is set to 10 minutes. When $FE_{\max} = 2500$, $t_{\max}$ is set to 1 hour. Once a candidate configuration reaches either stopping criteria, it is evaluated based on the I_H^{rpd} if $M \in \{2, 3, 5\}$, and the $I_{\epsilon+}^1$ when $M = 10$. In particular, our decision to use the $I_{\epsilon+}$ indicator when $M = 10$ was motivated by the practical and theoretical limitations of the I_H^{rpd} metric (see Section 2.1.4). However, in other tuning tasks it has been shown that hypervolume-based and ϵ-based metrics both lead to high-performing algorithm configurations [157].

In order to compute the quality metrics mentioned above, we first pre-process both reference fronts and approximation fronts produced by all candidates. Concerning reference fronts, we initially randomly generated 1 000 Pareto-optimal points using the methodology described in the original DTLZ [61] and WFG [112] papers. To improve these fronts, we followed a three-step approach. First, we used the randomly generated reference sets to tune selected MOEAs for scenarios with $FE_{\max} = 10\,000$, obtaining high-quality parameter settings for these algorithms. We then ran these algorithms for 100 000 FEs using the parameter settings obtained in the previous step. Finally, we merged the approximation fronts from all MOEAs with the randomly generated reference sets, and filtered out the dominated solutions. Regarding the pre-processing of approximation fronts produced by candidates, we took precautions to avoid skewed results due to normalization in the presence of strong outliers. Concretely, we have filtered out solutions that were dominated by an upper bound point $\mathbf{u}$, determined based on the extreme solutions found in the improved reference sets:

$$\mathbf{u} = \begin{cases} [10]^M, & \text{if } M \in \{2, 3\} \\ [15]^5, & \text{if } M = 5 \\ [25]^{10}, & \text{if } M = 10 \end{cases} \tag{4.1}$$

This upper bound is also used to determine the reference point $\mathbf{r}$ used to compute the I_H^{rpd} metric. In particular, we set $\mathbf{r} = \alpha \cdot \mathbf{u}$, where α is a factor used to ensure that extreme solutions are valued accordingly; we use $\alpha = 1.1$.

Finally, the last precaution we take regarding tuning concerns the separation between a training set and a test set, following the same approach from the previous chapter. We follow this approach as an alternative to the leave-one-problem-out as in cross-validation since that would incur a computational cost 16 times higher than the current tuning cost, which would make this assessment close to infeasible.

---

from a database.

Table 4.3: Parameter space for tuning specific MOEA/D parameters.

Parameter	ρ	δ	ϕ	t_{size}	ν
Domain	$[0.1, 1]$	$[0, 1]$	$[0.01, 1]$	$\{1, 2, \ldots, 20\}$	$\{2, 3, \ldots, 10\}$

Table 4.4: Parameter space for MO-CMA-ES learning parameters.

σ_0	d	p_{target}	p_{thresh}	c_p	c_c	c'_{cov}	u'_{cov}
$[0, 1]$	$\{1, 2, \ldots, 50\}$	$[0, 0.5[$	$[0, 0.5[$	$[0, 1]$	$]0, 1]$	$[0, 1[$	$[0, 1[$

4.2.3 Parameter space for tuning

Due to the very diverse set of MOEAs we consider, we use a flexible parameter space to meet their characteristics. For the parameters that apply to the majority of the algorithms, we define domains as follows:

Population size (μ): For the Pareto- and indicator-based algorithms, we consider $\mu \in \{10, 20, \ldots, 100\}$. For MOEA/D, the population size cannot be too small otherwise its weight set might lose representativeness. More precisely, the traditional methods used for generating weights make it so that a small number of weights on a large number of objectives might miss large regions of the objective space. We therefore tune MOEA/D considering $\mu \in \{100, 200, \ldots, 500\}$. Concerning NSGA-III, the population size and the number of weights used are tightly related parameters, so we detail them later.

Underlying EA: As previously explained, we give algorithms a choice between two different underlying EAs: (i) GA, using the traditional deterministic tournament, SBX crossover and polynomial mutation operators; or (ii) DE, using the differential mutation operator and the binomial crossover in the DE/rand/1/bin fashion. Additional parameters associated with each variation operator are given in Table 4.2, namely tournament size (t_{size}) and the crossover and mutation probabilities and distribution indices (p_c, p_m, η_c, and η_m, respectively) for the GA variation operators, and crossover rate and scaling factor (CR and F, respectively) for the DE operators. As previously explained, these parameters do not apply to MO-CMA-ES.

In addition, the MOEAs we include in this comparison that were not considered in the previous chapter present additional parameters which we briefly list below. We refer to the original papers for a better understanding of all parameters.

MOEA/D. The DRA strategy used by MOEA/D requires a number of additional parameters, which we tune using the domains given in Table 4.3. Furthermore, the decomposition part of the algorithm can be selected from three options: weighted sum aggregation (WS), Tchebycheff aggregation (TA), or penalty-based boundary intersection (PBI). When PBI decomposition is used, an additional penalty parameter $\xi \in \{1, 2, \ldots, 10\}$ has to be tuned.

MO-CMA-ES. Concerning learning-related parameters, MO-CMA-ES presents a set of default formulae that self-adjust in function of the number of variables the given MOP presents. In this work, we preserve this approach, but also give irace the option of tuning a parameter using the tuning domains given on Table 4.4. More precisely, for each of the parameters listed, irace first decides whether it will use the default formula or tune the parameter using its given domain[2]. Two parameters constitute an exception to this pattern: (i) the initial step size σ_0, for which no default

[2]We use surrogate parameters to tune the covariance matrix learning and unlearning rates and ensure they are always smaller than c_c. Concretely, $c_{cov} = c'_{cov} \cdot c_c$ and $u_{cov} = u'_{cov} \cdot c_c$.

formula is used, and; (ii) the covariance matrix unlearning rate u'_{cov}, which is not a parameter of the original MO-CMA-ES but we use it because it is part of the most widely adopted MO-CMA-ES implementation [120]. In fact, for this parameter irace can select between three different options: (i) the original approach, i.e., $u'_{cov} = c'_{cov}$; (ii) the formula used in Shark [120], or; (iii) tuning this parameter according to the domain given in Table 4.4. Concerning the multi-objective components of MO-CMA-ES, we consider three parameters to be tuned: (i) the quality indicator, which can be *epsilon* or *hypervolume*; (ii) using steady-state replacement or not; and (iii) the notion of success, which can be *population* or *individual*-based [221].

NSGA-III. The authors of NSGA-III proposed a particular weight-generation methodology, meant to prevent the objective space gaps we previously discussed for MOEA/D. The domains we set for these parameters are dependent on the number of objectives M, following the original NSGA-III paper. For $M \in \{2, 3, 5\}$, we set $H_1 \in \{2, \ldots, \Omega\}$, where Ω is set as follows:

$$\Omega = \begin{cases} 100, & \text{if } M = 2 \\ 30, & \text{if } M = 3 \\ 10, & \text{if } M = 5 \end{cases} \tag{4.2}$$

When $M = 10$, we use the $\{1, \ldots, 3\}$ domain for both H_1 and H_2. The population size is then calculated as a function of the number of weights generated. Concretely, $\mu = \mu' \cdot |W|$, where $|W|$ is the cardinality of the weight set W and $\mu' \in [0.5, 1.5]$.

4.2.4 Testing setup

Once all algorithms have been properly tuned for each scenario, we run them 25 times on the test set. In particular this set is unique w.r.t. tuning set since we reserve a set of problem sizes specifically for testing, namely $n_{\text{testing}} \in \{30, 40, 50\}$. The stopping criterion for all runs is the same used for tuning: primarily, MOEAs are allowed to use $FE_{\max}$ function evaluations, but the cutoff times $t_{\max}$ are used to ensure the feasibility of our investigation. Once algorithms finish running, the same pre-processing performed for approximation fronts generated by candidates during tuning is repeated during testing. In particular, the same definitions for the bounds and reference points (i.e., same $\mathbf{u}$, α and $\mathbf{r}$) are used. The processed fronts are then evaluated according to three quality indicators previously discussed, namely the I_H^{rpd}, $I_{\epsilon+}^1$, and I_{IGD} metrics (see Section 2.1.4).

4.2.5 Improvements over other assessments

Compared to other recent works that attempted to assess the performance of MOEAs on many-objective problems [98, 151], our work can be considered an improvement for two main sets of reasons. Algorithm-wise, we select at least two relevant MOEAs from each paradigm, and so we are able to draw conclusions within each paradigm, as we will later see. In addition, the conceptual separation between MO components and underlying EA allows us to analyze how the latter interacts with the former. Moreover, our work is the first to include a CMA-ES-based algorithm to a large performance assessment. In particular, by directly comparing the results of SMS and MO-CMA-ES we are able to draw conclusions about the efficacy of this underlying EA in the circumstances considered in this work.

The second set of improvements concern the carefully designed experimental setup described in this section. Concretely, this is one of the first MOEA assessments where an automatic algorithm configuration is used to properly tune numerical parameters. Furthermore, we take additional precautions concerning (i) the separation between tuning and test sets, (ii) assembling and improving the reference fronts, and (iii) using a set of diverse performance metrics. Finally, we also consider consider a rich set

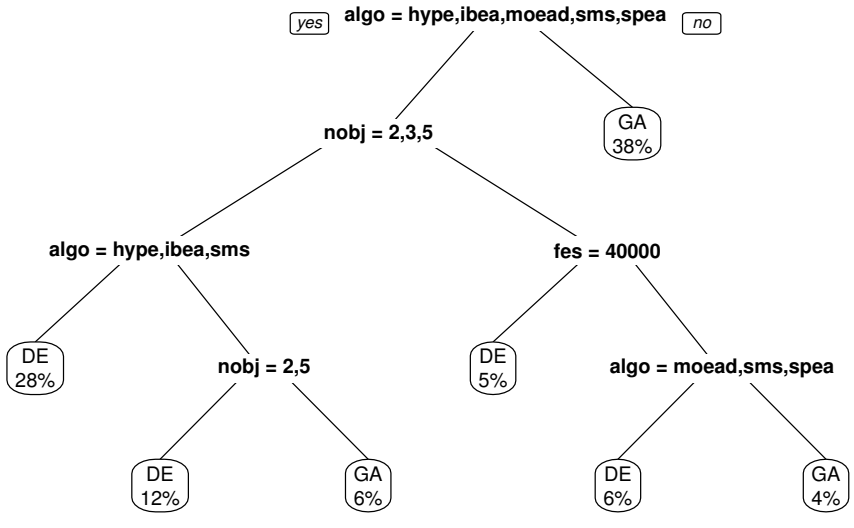


Figure 4.1: Classification tree depicting the choices of the underlying EA selected by irace for all MOEAs except MO-CMA-ES. Cluster cardinalities are given as percentages in leaf nodes.

of experimental scenarios and select problem sizes that are more representative of real-world situations than the ones traditionally used in the literature.

4.3 Preliminary insights

Given the large amount of experimental results produced, we split our discussion in three sections. In this first one, we conduct a high-level analysis, detailing observed insights that are critical for understanding results.

Underlying EA choices. We start our analysis discussing the choice of the underlying EA by the automatic configurator, using the classification tree given in Figure 4.1, generated using the rpart R library. In the tree, the left subtree of a node represents the subset of scenarios that satisfy the condition stated by the node. For instance, the root of the classification tree shows that the most important factor affecting the choice between the different underlying EAs is the MOEA considered. The left subtree refers to HypE, IBEA, MOEA/D, SMS, and SPEA2, while the right subtree refers to MOGA, NSGA-II, and NSGA-III. In fact, the first insight from this analysis is that for the three latter MOEAs, using GA as traditionally adopted in the literature is a proper choice regardless of the specific experimental factors. This is a remarkable observation, considering that previous literature insights would suggest DE as a more effective choice in general, both for single- and multi-objective optimization.

Moving along the left subtree, we see that for the remaining MOEAs DE is a more fit choice for most of the scenarios considered. Nonetheless, for specific MOEAs, the decision may depend on the particular scenarios. For example, for HypE or IBEA in most scenarios DE is the preferred choice but, when solving ten-objective problems given a low or moderate number of FEs, both algorithms perform best when coupled with GA. In fact, when we sum the percentages of different EA choices (given in leaf nodes), we see that DE was selected in 52% of the scenarios, with GA being selected in the remaining 48%. Although we remark that the differences in performance between using DE or GA varies significantly according to the MOEA considered, this result is strong evidence showing that the underlying EA choice should be taken into account at design time.

Parameter tuning effects. As previously discussed, we used irace as a configurator to fine-tune the parameters of the MOEAs for each experimental scenario. Our motivation was to avoid using sub-optimal parameter settings, either due to tuning via the trial-and-error approach or by transferring

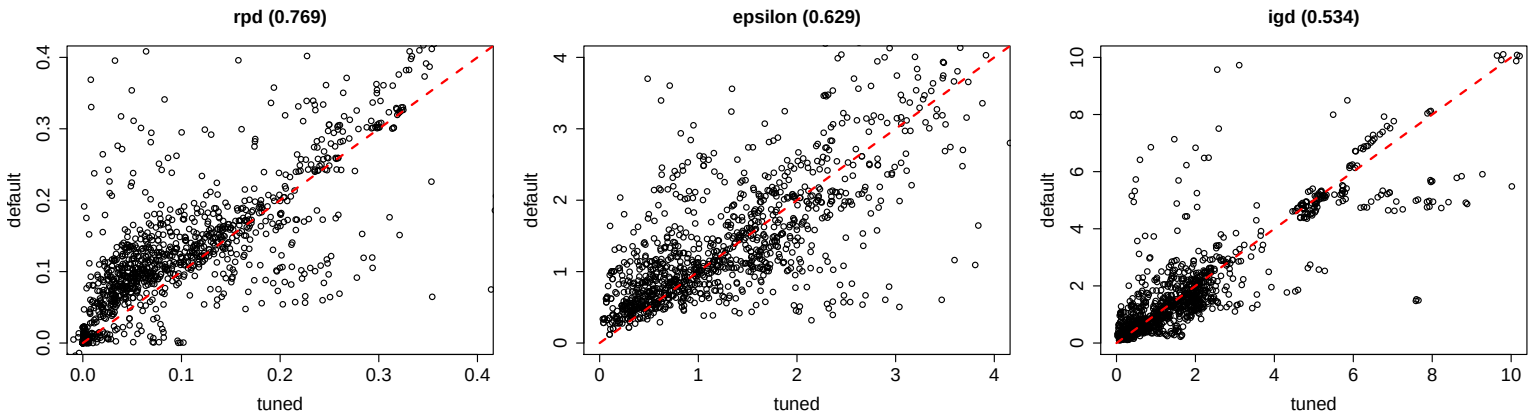


Figure 4.2: Scatter plots comparing tuned (x-axis) and default (y-axis) performances of MOEAs according to the (i) I_H^{rpd} (left), (ii) $I_{\epsilon+}^1$ (center), and (iii) I_{IGD} (right) metrics. Results have been averaged over each of the 25 run sets, paired by all the experimental factors considered.

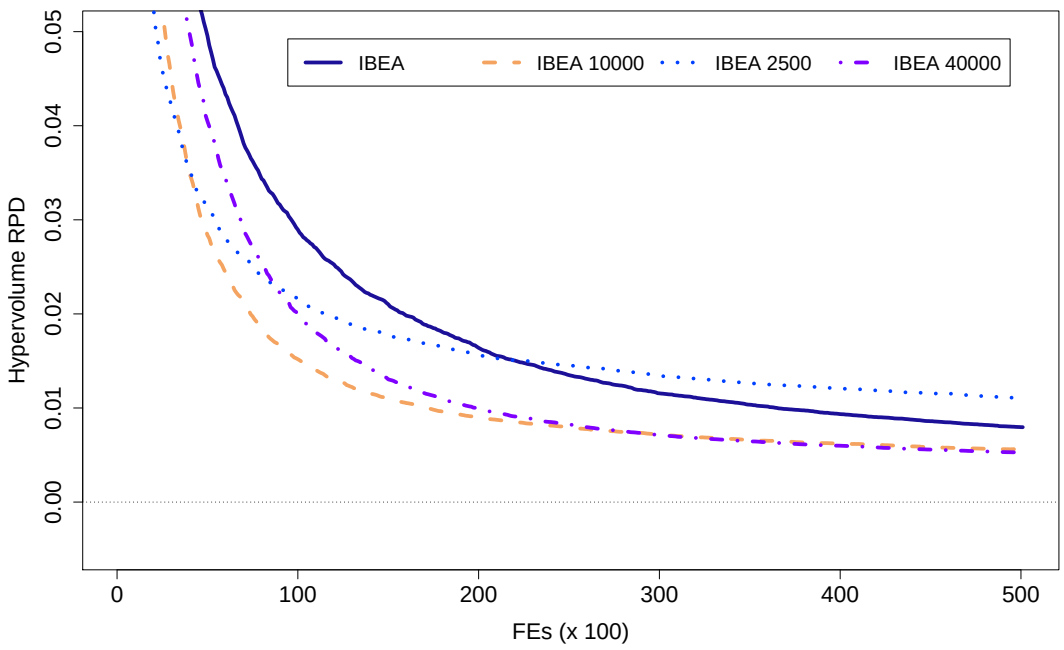


Figure 4.3: I_H^{rpd} performance of IBEA using DE as underlying EA with different numerical parameter settings on WFG8 ($M = 3, n_{\text{var}} = 30$).

settings across different experimental scenarios. To illustrate the effects of tuning, Figure 4.2 shows scatter plots comparing results obtained by MOEAs using default settings versus tuned ones. In particular, the points depicted in the plots are pairs (t, d), where t and d are the mean results over 25 runs according to the given metric of the tuned and the default version of a MOEA, respectively. The only algorithm left out of this analysis was MOGA, since the original work does not provide default parameter settings. Each point corresponds to the test result of one MOEA on a specific MOP (for each, three different numbers of variables) and a specific scenario. In total, our sample size for each plot totalizes 4032 pairs (3 n_{var}, 14 MOPs, 3 FE_{max}, 8 MOEAs, and 4 M). In addition, we provide the ratio of pairs where $t \leq d$, given in parenthesis. As it can be seen, the metric where improvements are more significant is the I_H^{rpd}, a consequence of having used it as the tuning metric on most scenarios. Benefits can also be observed for the remaining metrics, although clearly the contradicting natures of the I_H^{rpd} and the I_{IGD} reduce the tuning benefits for the latter.

To further illustrate the effects of tuning, Figure 4.3 shows the development of I_H^{rpd} over the number of function evaluations (averaged across 25 runs) of IBEA with DE as underlying EA on problem WFG8 ($M = 3$, $n_{\text{var}} = 30$), using four different numerical parameter settings. The first one, labeled *IBEA*, uses the parameter settings proposed in the original DEMO$^{\text{IB}}$ paper [219]. The remaining three curves refer to runs using the parameter settings selected by irace for the different FE_{max} values, and are labeled *IBEA* FE_{max}. For this plot, however, the algorithm was allowed to use $FE_{\text{max}} = 50\,000$.

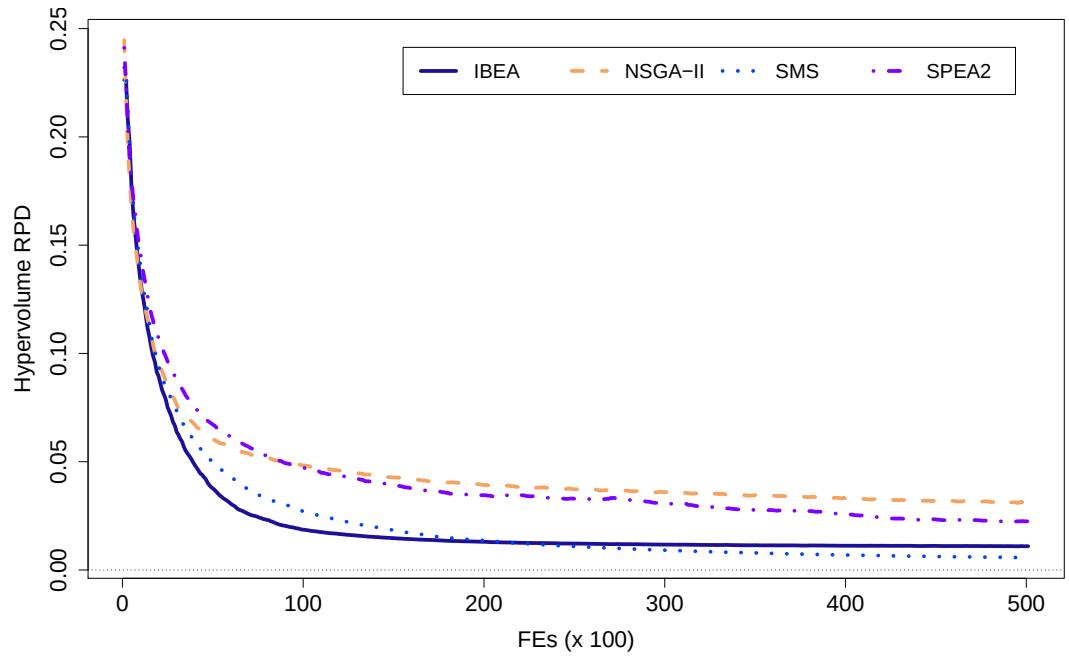


Figure 4.4: I_H^{rpd} performance of different MOEAs using parameter settings tuned for a common stopping criterion on WFG3 ($M = 3, n_{\text{var}} = 50$).

Table 4.5: Spearman's correlation coefficient between MOEA rankings for different $FE_{\max}$ values.

	$2\,500 \times 10\,000$	$2\,500 \times 40\,000$	$10\,000 \times 40\,000$
tuned	**0.77**	**0.7**	0.89
default	0.74	0.66	**0.96**

As one can see, the better performance presented by *IBEA 2500* on lower FE budgets contrasts with its poor performance on larger budgets. The same pattern is observed in all curves: their best performance w.r.t. the other curves is obtained around the number of FEs for which the algorithm was tuned. Concerning the default settings, this plot clearly shows that tuning via the trial-and-error approach can lead to sub-optimal performance.

The importance of $FE_{\max}$. To illustrate the effects of using different values for $FE_{\max}$, Figure 4.4 shows the performances of different MOEAs that were tuned for one of the selected $FE_{\max}$ values when allowed to run until $FE_{\max} = 50\,000$. As one can see, the performance curves intertwine at several points of the plot, meaning that different stopping criteria would favor different algorithms. For more general conclusions, we compute the correlations between the MOEA rankings grouped by $FE_{\max}$ values. More precisely, we take all the rank sums computed (all MOEAs according to each metric on each scenario) and group them by $FE_{\max}$ value, paired according to the remaining factors. Table 4.5 gives the Spearman correlation coefficients, considering both results from tuned and from default versions of the MOEAs. For brevity, scatter plots of this analysis are provided as supplementary material [33].

The correlation coefficients for the combinations of $FE_{\max}$ values provide some interesting insights. First, values neither indicate a perfect nor an absence of correlation. A more in-depth analysis of the scatter plots show that this correlation is stronger for the extreme results than for the central ones. In other words, the best- and worst- performing MOEAs are far more clearly defined than the relative order of an intermediary-performing group of algorithms. Furthermore, correlations between $FE_{\max} = 2\,500$ and the other values are always the lowest ones, reinforcing that the idea of designing an algorithm that can be simultaneously suited for low and high $FE_{\max}$ scenarios is difficult to accomplish.

Dominance resistance. As previously discussed, the most difficult issue related to the increase in the number of objectives is dominance resistance, and hence we make two observations about it. First,

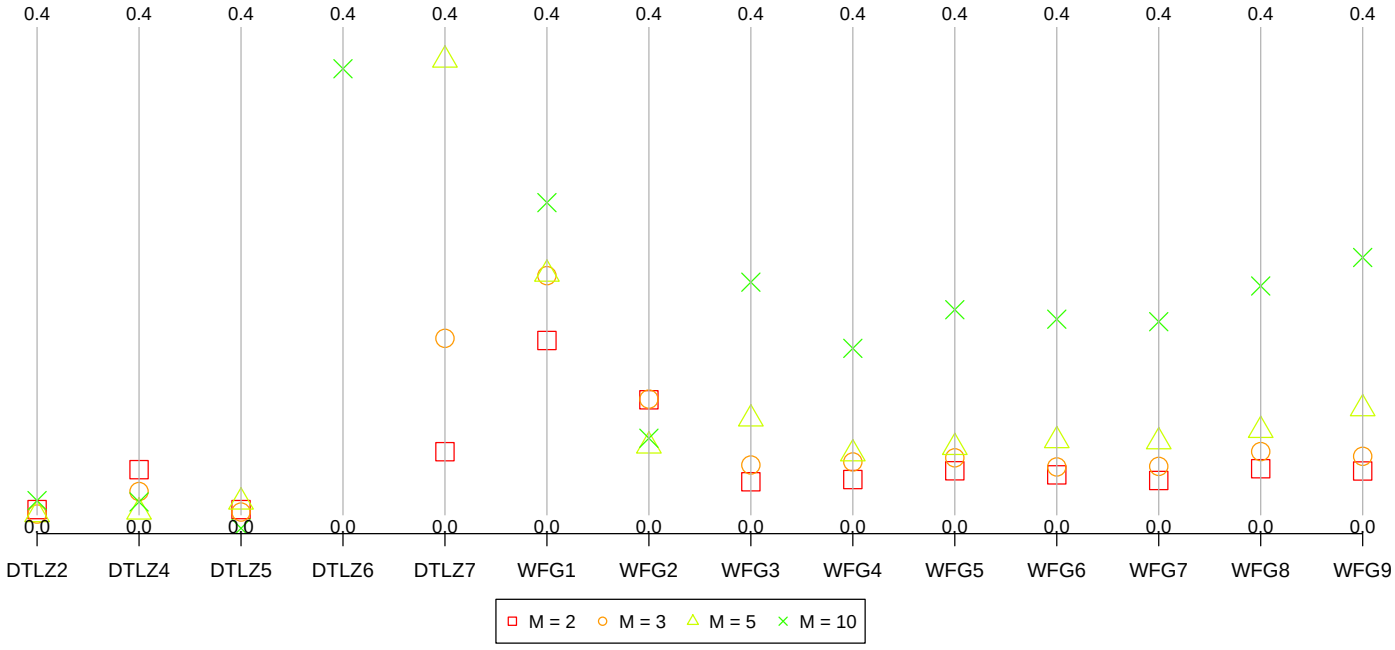


Figure 4.5: Parallel coordinate plot depicting the dominance resistance affects problems in different rates. Each point is the aggregation of all I_H^{rpd} results, grouped by indicator, M, and problem.

we see a clear trend in results confirming that the performance of MOEAs in general decreases as the number of objectives grow. In fact, the performance of algorithms on ten-objective problems is poor for many problems, showing that MOEAs still need major theoretical improvements to deal with dominance resistance. We remark, though, that not all algorithms are affected by dominance resistance in the same degree, as we will detail later. The second observation concerns the effect of particular problem characteristics on how much increases the difficulty of benchmark problems. In particular, the increase in M affects the performance of the algorithms in very different ways, as we illustrate in Figure 4.5. Each point is the aggregation (mean) of all I_H^{rpd} results, grouped by the number of objectives M, and the MOP. For brevity, the plots depicting the remaining indicators are provided as supplementary material [33]. Clearly, some problems such as DTLZ7 are far more affected by dominance resistance than others. In addition, for particular problems and indicators one can see the opposite effect. For instance, I_H^{rpd} results for DTLZ4 get better as M increases, and the same happens when we observe both I_H^{rpd} and $I_{\epsilon+}^1$ results for WFG2. Further analysis on this topic is given in Section 4.5.

Performance metric correlations. As previously discussed, each performance metric values specific characteristics that approximation fronts should present. For this reason, we make a few comments on the correlations between these metrics on selected scenarios. First, it is common to see different rankings for the same MOEAs depending on the metric given whatever the scenario one considers. To illustrate this, we use as case study the WFG8 function with $M = 2$ and $n_{\text{var}} = 30$. Boxplots for the I_H^{rpd} and I_{IGD} for SMS and IBEA are given in Figure 4.6 (left), the I_H^{rpd} values (top) indicating better performance for IBEA while I_{IGD} (bottom) indicates the opposite. As previously discussed, on concave problems the I_H^{rpd} values spread over distribution, while the I_{IGD} does the opposite. This is confirmed by the empirical attainment function (EAF) difference plot where IBEA finds more solutions on the extremes of the objective space, although it is clear that the differences between these fronts are subtle.

Despite the difference in rankings depending on the metric considered, the best and worst performing MOEAs are clearly defined on most scenarios when $M \in \{2, 3, 5\}$. It is only when $M = 10$ that one sees major variations in the rankings of the algorithms from metric to metric. In particular, we noticed that MOEAs tend to display better performance according to the metric used for their assessment on its original proposal time. To corroborate our conclusions, Table 4.6 shows Spearman's correlation

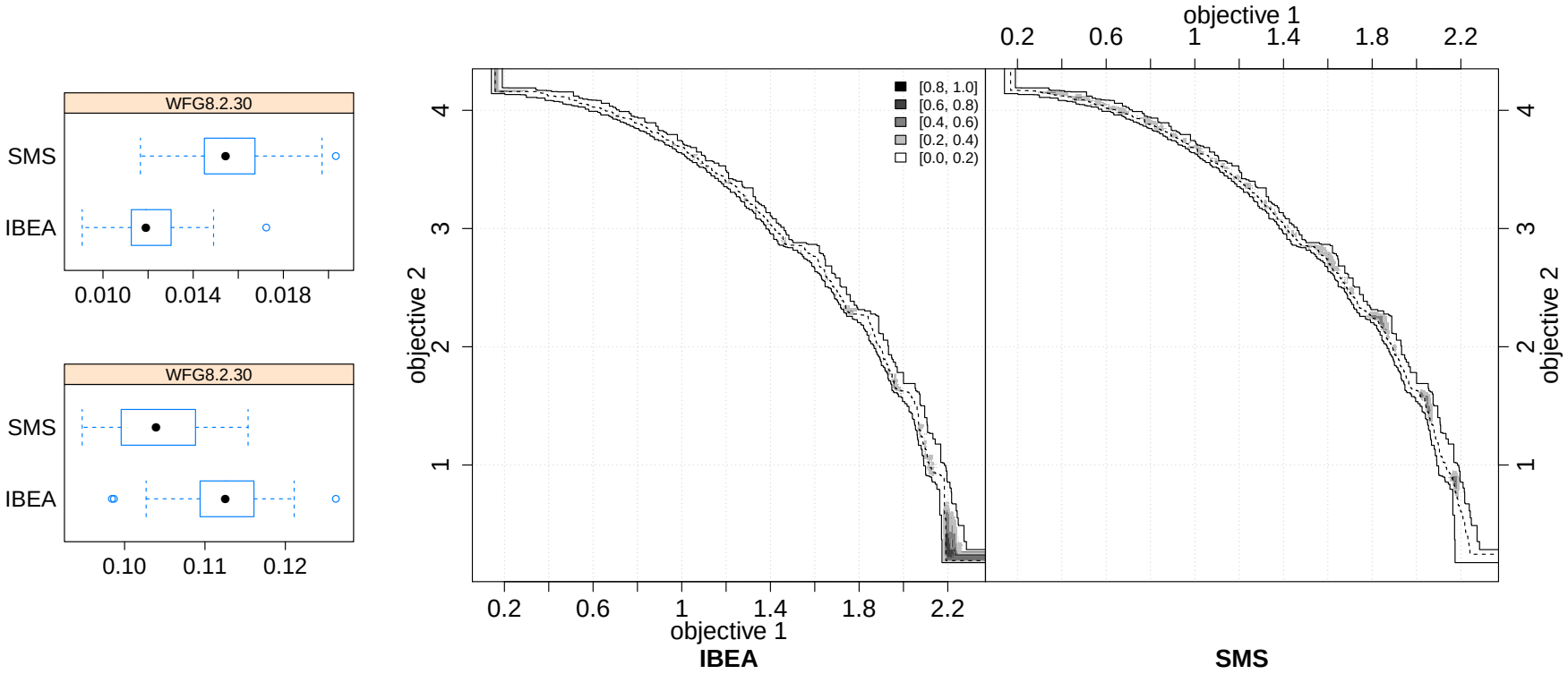


Figure 4.6: Contradictions between I_H^{rpd} and I_{IGD} considering SMS and IBEA performances on function WFG8 ($M = 2$, $n_{\text{var}} = 30$, $FE_{\max} = 10\,000$). On the left, boxplots depicting I_H^{rpd} (top) and I_{IGD} (bottom) performances. On the right, EAF difference plot of the fronts produced by IBEA (left box) and SMS (right box).

Table 4.6: Spearman's correlation coefficient depicting MOEA ranking similarities for pairs of performance metrics.

M	2	3	5	10
$I_H^{rpd} \times I_{\epsilon+}^1$	0.95220	0.94121	0.89684	**0.78321**
$I_H^{rpd} \times I_{IGD}$	0.86382	0.86173	0.76142	**0.56793**
$I_{\epsilon+}^1 \times I_{IGD}$	0.90959	0.93441	0.81812	**0.73905**

coefficients between the rankings assigned by the performance metrics to MOEAs on different M scenarios. The first column, for instance, gives correlation coefficients between different metrics on scenarios with $M = 2$. As one can see, correlation levels are considerably high for $M \in \{2, 3\}$, decreasing significantly for $M = 10$. In particular, we remark that the lowest correlation observed concerns the I_H^{rpd} and I_{IGD} indicators when $M = 10$. These indicators have recently been shown to consistently disagree on concave problems [129], and, as we will discuss in Section 4.5, this is the type of problem that becomes considerably harder with increasing M.

4.4 MOEA assessment

We now move on to discuss of the performance of the MOEAs across all scenarios considered. We group our discussion for scenarios where $M \in \{2, 3, 5\}$ since the previously discussed agreement between performance metrics allow us to clearly identify the best-performing MOEAs on these scenarios, and we discuss results for $M = 10$ separately. Moreover, we show only results for $FE_{\max} = 10\,000$ for brevity, although we remark that our conclusions are general enough to account for all scenarios. The complete set of results as well as a more detailed analysis concerning each specific scenario are provided in Appendix B.

4.4.1 Performance patterns when $M \in \{2, 3, 5\}$

A rank sum analysis is given in Table 4.7 from which we extract a few performance patterns in this section. From the rank sum analysis, we can cluster algorithms into four different groups according to their performance on scenarios with $M \in \{2, 3, 5\}$. In general, SMS and IBEA are the algorithms that present the best and most robust performance, regardless of the experimental factors considered. The second group is formed exclusively by MOGA, which consistently, as expected, presents the worst performance among all MOEAs. The remaining two groups comprise algorithms whose performance is sensitive to specific problem characteristics and to the FE budget they are given, but also to the number of objectives considered. The first of these two groups contains NSGA-II, SPEA2, and HypE, which present ranks that worsen with increasing number of objectives. By contrast, the rank sums of MOEA/D, NSGA-III, and MO-CMA-ES are particularly high for low number of objectives, but improve for more conflicting objectives.

The performance patterns observed raise some interesting insights. First and most important, IBEA and SMS prove that it is possible to design MOEAs that can be applied for a wide range of number of objectives, even if their effectiveness lowers in face of dominance resistance. Differently, the algorithms we consider that are designed specifically with many-objective problems in mind, HypE and NSGA-III are unable to meet this challenge. This is also the case for MOEA/D and MO-CMA-ES, which have been applied to different M values in the literature, but rank worse than many algorithms when $M = 2$. The second insight we observe is that algorithms are nearly perfectly clustered according to their underlying design paradigm, with only few exceptions. We now summarize the most important conclusions concerning each paradigm below.

Indicator-based. The performance displayed by SMS and IBEA is remarkable with both algorithms being particularly robust to all the experimental factors considered in this work. More importantly, these MOEAs rank best not only for the indicator that is used to drive their search but for essentially all indicators. By contrast, the performance of HypE is considerably dependent on the scenario under analysis. In fact, the only scenarios where HypE ranks immediately after SMS and IBEA are the ones with $M = 3$. In particular, we remark that the explanation for the low performance of HypE in $M = 5$ scenarios is not straightforward. At a first glance, one may consider that the I_H^h estimation for $M > 3$ is to blame, so we have tested this algorithm with (i) different numbers of samples to increase the estimation accuracy, and (ii) the exact computation instead of the estimation on a restricted setup. The changes in the number of samples produced no significant differences in the rank sums. In additon, although the exact computation improved the performance curves of HypE by a large margin on most problems considered, it was still not enough for it to match the remaining indicator-based algorithms. More importantly, during those experiments we noticed that, for many of the problems considered, the performance of HypE worsened over time after an initial improving stage. Although no theoretical proof concerning this issue has been published so far, these results make us believe that the I_H^h metric lacks Pareto-compliance.

Dominance-based. The performance trends of these algorithms are highly dependent on M. When $M \leq 3$, the performance differences between dominance- and indicator-based algorithms are relatively small. In fact, SPEA2 is sometimes able to outrank IBEA as a consequence of the previously reported particularly good performance of SPEA2 on the WFG benchmark [42]. When $M = 5$, the gaps to the indicator-based become larger, but overall the performance of these MOEAs is still reasonable. The poor performance observed occasionally is related to the specific problem characteristics we will later discuss. The only dominance-based MOEA that presents poor performance all along is MOGA. In part, this is explained by its lack of elitism, although exploratory experiments we conducted have shown that a simple elitist MOGA is unable to match the performance of NSGA-II, for instance.

Decomposition-based. The two decomposition-based algorithms we assess in this work are outranked by most MOEAs on $M = 2$, including many indicator-based and dominance-based algorithms. On $M \in \{3, 5\}$, the rankings of MOEA/D improve considerably, sometimes to the point of being considered the best MOEA under specific scenarios. By contrast, the overall performance of NSGA-III differs very little from the performance of NSGA-II. In fact, we conduct a set of additional experiments specifically targeted at comparing these two algorithms. Moreover, we include in that comparison the default versions of both algorithms to test whether tuning could be a factor. Rank sums of I_H^{rpd} results for selected scenarios are given in Table 4.8 to illustrate our conclusions. First, in low M scenarios like $M = 2$ both NSGA-II versions, i.e. default and tuned, outrank the NSGA-III versions. On scenarios with $M = 5$, we see that the tuned versions become equivalent, although the default version of NSGA-III outranks the default version of NSGA-II. This result reinforces the importance of proper parameter tuning when specific experimental scenarios are used. Finally, we remark that these conclusions are consistent with a recent discussion about the effectiveness of NSGA-III for a low number of objectives w.r.t. NSGA-II conducted by their authors [207].

MO-CMA-ES. Concerning its parameters, we remark that irace always selected the exclusive hypervolume contribution as quality metric. In addition, except for c_c, c'_{cov}, u'_{cov}, irace often selected learning parameter settings different from the default ones. Performance-wise, we observe two main factors that affect this algorithm. First, the performance displayed by MO-CMA-ES in $FE_{\max} = 2\,500$ scenarios is much worse than that of the remaining MOEAs. Requiring a large number of function evaluations is a known behavior of the original CMA-ES algorithm, and as we show it can also be observed by its multi-objective version. When given enough FEs to run, however, we see that the performance of MO-CMA-ES improves with the increase in M, as previously mentioned. In fact, for $M = 5$ scenarios it is often similar to that of SMS and IBEA, although its performance on WFG1 is always particularly poor.

4.4.2 Ten-objective results

The large increase in the number of objectives to ten leads to many important changes we individually address in this section. First, we notice that MOEA results show that much improvement regarding both the number of nondominated solutions and the spread of the Pareto fronts could still be achieved, especially for concave WFG problems. In fact, I_H^{rpd} values for most problems and MOEAs were in the $[-2, 0]$ range and, in many problems, MOEAs were able to find over 30 000 additional nondominated solutions. For this reason, the analysis we conduct in this section considers metrics computed using reference sets assembled from an aggregation of the approximation fronts produced by all MOEAs and the Pareto fronts we had initally assembled as described in Section 4.2.

A second important observation concerns the previously discussed disparity between the rankings given in Table 4.7, i.e., the different performance some MOEAs present depending on the metric considered. As previously discussed, these results are largely due to the contradictions the performance metrics present, leading to rather different results between I_H^{rpd} and I_{IGD}, particularly on the concave WFG problems. These results reinforce two known premises in multi-objective optimization that are typically overlooked in the literature. First, there is a strong indication that devising a single MOEA to optimize a diverse set of metrics is a challenging task on many-objective scenarios. Second, using a single performance metric to evaluate MOEAs during their design as traditionally done is an approach prone to major drawbacks, as an algorithm might excel in some characteristics while compromising others.

Concerning the overall performances observed, a few MOEAs are able to stand out. First, the only algorithm to be well-ranked according to all metrics is IBEA, being considered the best both according to I_H^{rpd} and I_{IGD}. By contrast, MOEA/D and NSGA-III also rank first for particular metrics, but not as well according to others. Finally, SMS and MO-CMA-ES display very good performance according to I_H^{rpd} and $I_{\epsilon+}^1$ in many problems, but rank worse according to I_{IGD}.

Table 4.7: Rank sum difference (in parenthesis) between the given MOEA and the lowest ranked ($FE_{\max} = 10\,000$). MOEAs highlighted in boldface present rank sums statistically significantly lower than the others according to Friedman's nonparametrical test.

					$M = 2$				
I_H^{rpd}	**IBEA**	**SMS (113)**	SPEA2 (1149)	NSGA-II (1846)	MOEA/D (2819)	HypE (3593)	CMA (3731)	NSGA-III (4017)	MOGA (6682)
$I_{\epsilon+}$	**SMS**	IBEA (425)	SPEA2 (894)	NSGA-II (1993)	MOEA/D (3091)	HypE (3401)	CMA (3749)	NSGA-III (4231)	MOGA (6740)
I_{IGD}	**SMS**	SPEA2 (717)	IBEA (1386)	HypE (2391)	NSGA-II (2787)	MOEA/D (4235)	NSGA-III (5076)	CMA (5247)	MOGA (7302)
					$M = 3$				
I_H^{rpd}	**SMS**	IBEA (556)	MOEA/D (1805)	HypE (2290)	SPEA2 (2302)	CMA (3616)	NSGA-II (4378)	NSGA-III (4627)	MOGA (7029)
$I_{\epsilon+}$	**SMS**	IBEA (494)	SPEA2 (2516)	CMA (2968)	HypE (3132)	MOEA/D (3552)	NSGA-III (4253)	NSGA-II (4885)	MOGA (7323)
I_{IGD}	**IBEA**	SMS (710)	SPEA2 (885)	MOEA/D (1375)	HypE (2932)	NSGA-II (3452)	NSGA-III (4148)	CMA (4198)	MOGA (6797)
					$M = 5$				
I_H^{rpd}	**SMS**	MOEA/D (1471)	IBEA (1535)	SPEA2 (3295)	CMA (3374)	NSGA-III (3897)	NSGA-II (4130)	HypE (5811)	MOGA (7562)
$I_{\epsilon+}$	**SMS**	IBEA (1713)	MOEA/D (2569)	CMA (2588)	NSGA-II (4086)	NSGA-III (4124)	SPEA2 (4701)	HypE (5907)	MOGA (7664)
I_{IGD}	**SMS**	IBEA (1936)	MOEA/D (1945)	NSGA-II (2555)	CMA (2594)	SPEA2 (3720)	HypE (5171)	NSGA-III (5204)	MOGA (7446)
					$M = 10$				
I_H^{rpd}	**IBEA**	SMS (222)	CMA (1116)	NSGA-III (2326)	SPEA2 (2532)	NSGA-II (3241)	HypE (4846)	MOEA/D (5114)	MOGA (6919)
$I_{\epsilon+}$	**MOEA/D**	IBEA (1258)	SMS (2794)	CMA (3250)	NSGA-III (3347)	NSGA-II (4045)	SPEA2 (4562)	HypE (5214)	MOGA (6922)
I_{IGD}	**NSGA-III**	**IBEA** (36)	SPEA2 (646)	NSGA-II (1776)	SMS (1828)	CMA (1987)	HypE (2557)	MOEA/D (4424)	MOGA (5151)

Table 4.8: Rank sum difference (in parenthesis) between tuned and default versions of NSGA-II and NSGA-III.

M	$FE_{\max} = 10\,000$			
2	**NSGA-II^t**	NSGA-II^d (1270)	NSGA-III^t (1424)	NSGA-III^d (2257)
5	**NSGA-II^t**	**NSGA-III^t (11)**	NSGA-III^d (901)	NSGA-II^d (1212)

We now proceed to a more detailed analysis of these algorithms.

IBEA. The overall performance of IBEA is remarkable given that it is robust w.r.t. all experimental factors and to the different performance metrics as well. In fact, the only exceptions to the good performance of IBEA concern (i) the concave WFG problems and (ii) the DTLZ6 and DTLZ7 problems. However, most MOEAs perform poorly on these problems, indicating that this is a general MOEA limitation rather than specific IBEA issues. Two other facts are worth mentioning. First, the performance of IBEA is consistently good across the different $FE_{\max}$ values considered, but its top-performance is observed when $FE_{\max} = 40\,000$, where it is considered the best MOEA according to all metrics. Second, the computational complexity of this algorithm is low even for $M = 10$ scenarios, making it suited for any kind of application.

MOEA/D. Results for the different metrics show that MOEA/D becomes the best ranked algorithm according to $I^1_{\epsilon+}$, but at the same time it becomes the second worst MOEA according to I_H^{rpd} and I_{IGD}. Given the characteristics of the metrics, these results mean that MOEA/D is unable to maintain a good search diversity. Although a more in-depth analysis targeting the complex algorithmic design of MOEA/D would be required to fully understand how this algorithm can modify its behavior to such extent, we conducted some exploratory investigation targeting a particular feature. Traditionally, this algorithm has been used coupling DE with polynomial mutation, a choice that is never used in other DE-based MOEAs. For fairness, in this work polynomial mutation is not used when the tuner selects DE as underlying EA. To evaluate the effects of this choice, we have tuned an alternative version of MOEA/D allowing polynomial mutation to be used alongside DE. However, our experiments on the $M = 10$ and $FE_{\max} = 40\,000$ scenario have shown that adding polynomial mutation to MOEA/D rather reinforces the improved convergence at the cost of more diversity loss.

A second hypothesis we tested is whether MOEA/D would still suffer from such diversity issues when tuned for a different metric. Since tuning for the I_H^{rpd} is infeasible, we have tuned MOEA/D for the I_{IGD} metric, again on the $M = 10$ and $FE_{\max} = 40\,000$ scenario. Results show that tuning for the I_{IGD} has improved search diversity, reflected by the much better I_{IGD} rank sums (fourth best ranked MOEA). Concerning the remaining metrics, however, the rankings from MOEA/D do not improve for the I_H^{rpd}, and worsen considerably for $I^1_{\epsilon+}$. In fact, MOEA/D becomes one of the worst algorithms according to the latter. Concerning I_H^{rpd}, we notice that no rank sum improvements can be noticed due to the contradictions between it and I_{IGD}. More precisely, on the problems where both metrics agree we are able to see I_H^{rpd} performance improvements, but these gains are counterbalanced by performance losses on the remaining problems, and hence rank sums remain roughly unaltered.

NSGA-III. The performance of NSGA-III varies greatly depending on the given metric. Its best performance is observed according to I_{IGD}, largely due to the results on the concave WFG functions. In particular, NSGA-III is the algorithm less affected by the large number of objectives in this type of problem. We then focus our analysis on comparing NSGA-III and the two algorithms that share similarities with it, NSGA-II and MOEA/D. Concerning the former, the performance of NSGA-III

Table 4.9: Summarized results from a problem-wise comparison between MO-CMA-ES and SMS ($M = 10$, $FE_{\max} = 10\,000$). Problem and MOEA names have been shortened for brevity.

	D5	D6	W1	W2	W4	W6	W7	W8
I_H^{rpd}	SMS	SMS	CMA	=	SMS	CMA	=	SMS
$I_{\epsilon+}^1$	=	SMS	=	SMS	SMS	CMA	=	CMA
I_{IGD}	=	CMA	CMA	=	CMA	CMA	CMA	=

represents a significant improvement, demonstrating that the diversity measure of this algorithm is much more suited for scenarios with many objectives than the crowding distance used by NSGA-II. Regarding MOEA/D, NSGA-III does not face the same issue of having a single solution replicated for multiple weights, and hence diversity is not an issue to this algorithm.

SMS. Differently from scenarios with less objectives, SMS is not able to maintain its top-ranking performance with ten objectives, although it still gets the best results according to I_H^{rpd} when $FE_{\max} = 2\,500$. Since this scenario is the one that presents the largest cutoff time and SMS is known to get ever more computationally expensive with the increase in M, the most logical explanation is that SMS cannot be explored to its full potential in these scenarios. To corroborate this conclusion, we compute the average number of function evaluations used by SMS across different M scenarios. While for all scenarios with 2, 3, and 5 objectives SMS is able to fully use the budget on function evaluations, on the 10 objective problems this is not anymore the case due to the maximum time limits we have imposed. In particular, it uses on average 2478.82, 8858.32, and 32723.9 function evaluations in the scenarios with $FE_{\max}$ equal to 2500, 10000, and 40000, respectively. Furthermore, we highlight that the population size selected by irace also decreases on the scenarios where the drops are observed, confirming that the algorithm tries to adapt towards using the given FE budget, but SMS is just not the best choice for the given setup.

MO-CMA-ES. The performance of MO-CMA-ES on $M = 10$ is particularly similar to that of SMS, indicating that the exclusive hypervolume contribution as a search-guiding metric is determinant for the performance of these two algorithms. For this reason, we focus the analysis of MO-CMA-ES on a direct comparison to SMS.

Overall, SMS gets better results than MO-CMA-ES when the testing benchmark is considered as a whole and $FE_{\max} = 10\,000$, but the opposite happens according to I_{IGD} when $FE_{\max} = 40\,000$. A problem-wise analysis reveals that for particular problems some metrics can favor MO-CMA-ES over SMS. For brevity, rank sums for all scenarios are provided as supplementary material [33]. Summarized results for $FE_{\max} = 10\,000$ are given in Table 4.9. In particular, an equality sign means no statistical difference could be found between rank sums for the given problem according to the given metric. The first observation we make concerns the previously reported agreements and contradictions between the performance metrics. On the convex problems (WFG1–2), the $I_{\epsilon+}^1$ disagrees with the remaining indicators, whereas on the concave problems it is the I_{IGD} that disagrees with the others. In fact, the only exceptions to these patterns are WFG6, where all metrics agree that MO-CMA-ES outperforms SMS, and WFG8, where metrics disagree.

The second observation we make refers to the I_{IGD}. For this indicator, MO-CMA-ES often ranks better or at least equivalently to SMS on most WFG problems and on DTLZ5–6. These results are largely due to the robustness w.r.t. the different numbers of variables presented by MO-CMA-ES when evaluated by I_{IGD}. By contrast, we notice that MO-CMA-ES faces considerable difficulties with the particular geometry of WFG3, and on the deceptive WFG problems (WFG5 and WFG9).

In addition to the algorithms we individually addressed above, an important observation that yet

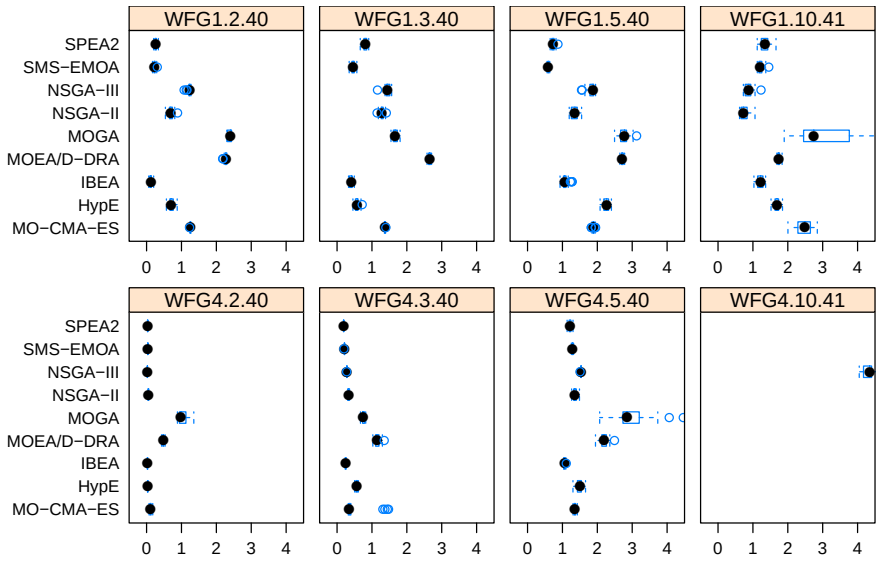


Figure 4.7: I_{IGD} performances of MOEAs given 10 000 FEs on selected problems with 40 variables and $M = 2$ to $M = 10$ objectives. Top: WFG1 having convex Pareto-optimal fronts; bottom: WFG4 having concave Pareto-optimal fronts.

had not been reported in the literature is the relatively good ranking of NSGA-II and SPEA2 in all ten-objective scenarios. Two precautions taken in this work are directly related to this good performance. First, both algorithms benefit directly from proper tuning, as the numerical parameters selected by irace differ considerably from the default adopted in the literature. Second, SPEA2 uses DE as underlying algorithm when given 10 000 and 40 000 FEs, reinforcing the need to consider different underlying EA algorithms when proposing a MOEA.

4.5 Problem feature analysis

The analysis conducted in the previous section raised some interesting observations we make in this section. In particular, the expected dominance resistance increase was seen in different rates according to the features presented by particular problems. We then address the most important features we have identified.

Geometry of Pareto-optimal fronts. On scenarios with $M \in \{2, 3\}$, having concave Pareto-optimal fronts does not affect the overall difficulty of the problems. However, we notice that all these problems become considerably harder as the number of objectives is increased, regardless of the other characteristics they present. By contrast, several MOEAs face difficulties on problems with convex Pareto-optimal fronts such as WFG1–3 already in scenarios with $M \in \{2, 3\}$, but the difficulty level of these problems is not significantly increased by the addition of more conflicting objectives. As a result, when $M \in \{5, 10\}$ MOEAs face less difficulty on convex problems than on the concave ones. We illustrate this with the I_{IGD} boxplots shown on Figure 4.7. The IGD values of MOEAs given $FE_{\max} = 40\,000$ on the (convex) WFG1 problem with $n_{\text{var}} = 40$ and increasing M are shown on the top, and the IGD values on the (concave) WFG4 problem with the same experimental factors are given at the bottom.

Local Pareto-optimal fronts. MOEAs face considerable difficulties on problems that present local Pareto-optimal fronts, namely DTLZ1, DTLZ3, and DTLZ6. In fact, DTLZ1 and DTLZ3 pose such a challenge when used with the number of variables we consider in this work that none of the MOEAs are able to find solutions within the bounds we set, no matter the number of objectives. To illustrate the challenge faced by some MOEAs on problems that present this characteristic, we run NSGA-II on several modified versions of the DTLZ6 problem, which we parameterize to regulate the number of local Pareto-optimal fronts. In the original DTLZ6 formulation, an auxiliary

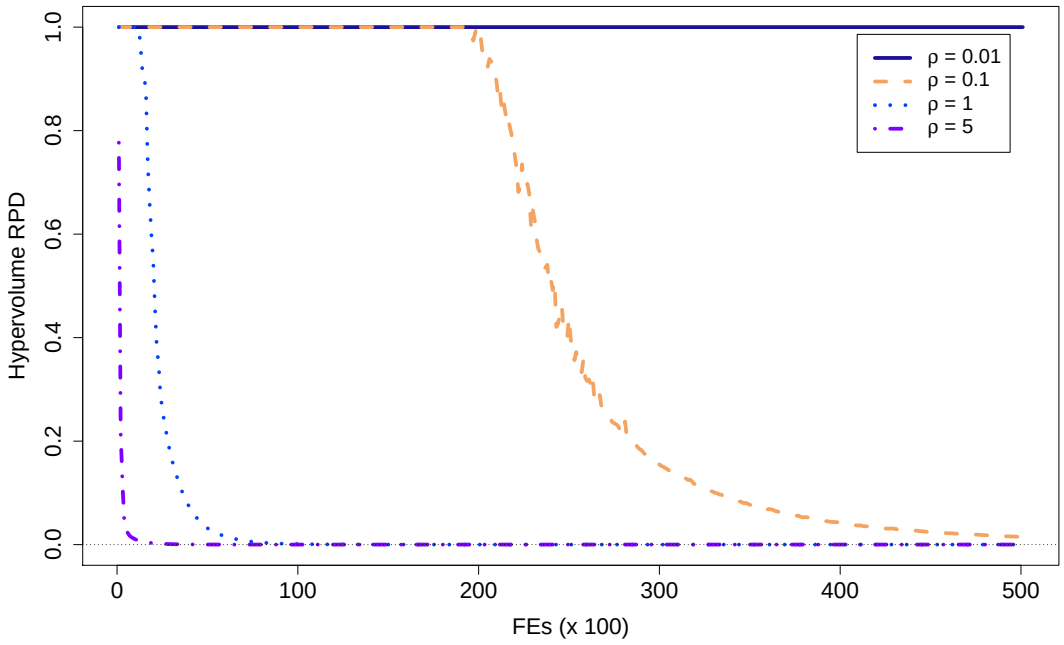


Figure 4.8: I_H^{rpd} performance of NSGA-II (tuned for $M = 3$, $FE_{\max} = 10\,000$) on different versions of the DTLZ6 problem ($M = 3, n_{\text{var}} = 50$).

function $g(x) = \sum_{x \in x_M} x^\rho$ with $\rho = 0.1$ is used to introduce local Pareto-optimal fronts. The plot given in Figure 4.8 shows the development of the average I_H^{rpd} value across the 50 000 FEs for NSGA-II (tuned for $M = 3$ and $FE_{\max} = 10\,000$) over 25 runs on the DTLZ6 problem ($n_{\text{var}} = 50$, $\rho \in \{0.01, 0.1, 1.0, 5.0\}$). In particular, the smaller ρ values lead to larger numbers of local fronts. As it can be seen, the shift in the curves follows the increase in ρ, confirming that for $M = 3$ the difficulty of this problem is strongly dependent on the number of local Pareto-optimal fronts.

Density bias. Two of the benchmark problems considered (DTLZ4 and WFG1) present bias, i.e., a shift in the density of the objective space meant to test whether MOEAs can maintain a well-spread search. When $M \in \{2, 3\}$, these problems pose difficulties for many MOEAs due to this feature. To illustrate the effect of bias, Figure 4.9 shows the development of the average I_H^{rpd} across the number of function evaluations for SPEA2 (tuned for $M = 3$ and $FE_{\max} = 10\,000$) across 25 runs on the DTLZ4 problem with $M = 3$, $n_{\text{var}} = 50$, and several different bias levels. More precisely, we test different values for the α parameter that regulates the level of bias presented by this problem, which was originally set to 100. We remark that larger α values lead to a stronger bias in the search space, and we test variants of this problem with $\alpha \in \{1, 10, 100, 1000\}$. As one may see, the performance of some MOEAs such as SPEA2 is greatly affected by the bias level chosen. However, the difficulty introduced by bias to both DTLZ4 and WFG1 does not increase with the addition of more objectives. In fact, the results for DTLZ4 are in general better for larger M values, as we had previously discussed.

As discussed in this section, understanding problem characteristics and their role on dominance resistance is critical for designing better-performing MOEAs. It is likely that tuning MOEAs in dependence of specific problem features would improve overall results, but the precise problem features are often not known in advance when dealing with real-world problems. We also remark that we refrain from further investigation on other problem features for two main reasons. First, fiddling with features from the DTLZ benchmark problems is not a trivial task, since this benchmark was not proposed for this goal. Second, we believe that the more rigorous approach to this investigation would be to create a set of feature-targeted benchmark problems, in order for one to have enough statistical evidence to draw definite conclusions, which we intend to do in future work.

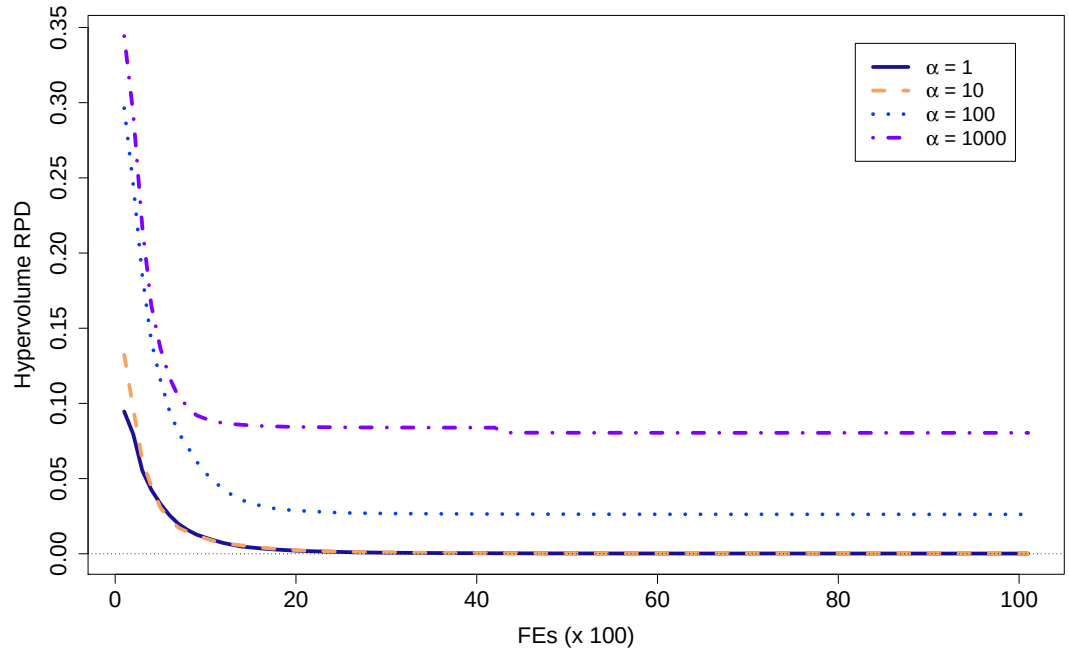


Figure 4.9: I_H^{rpd} performance of SPEA2 (tuned for $M = 3$, $FE_{\max} = 10\,000$) on different versions of the DTLZ4 problem ($M = 3, n_{\text{var}} = 50$).

4.6 Conclusions

In this chapter, we have conducted a comprehensive performance assessment of multi-objective evolutionary algorithms (MOEAs) both on multi- and many-objective optimization problems (MOPs and MaOPs, respectively). Our work goes beyond other comparisons given the number of algorithms and the number of different experimental factors we consider, namely: (i) different problem benchmarks, (ii) numbers of objectives, (iii) stopping criteria, (iv) numbers of variables, and (v) performance metrics. More importantly, this is the first work to clearly establish a separation between high-level multi-objective (MO) algorithmic components and the underlying evolutionary algorithm (EA) with which they are coupled, increasing both the fairness and extent of the comparison. In addition, we have used a rigorously designed experimental setup, taking precautions such as tuning numerical parameters and producing/refining high-quality reference fronts. Given the large amount of data produced in this work, we have focused the discussion on the most important insights we could observe. To cite a few, we have empirically demonstrated the importance of designing MOEAs with a set of diverse performance metrics in mind, as well as selecting at design time which underlying EA suits the given MO components best. In addition, we have shown that designing a single algorithm to simultaneously optimize problems with different numbers of objectives and different stopping criteria is a challenging task. Nonetheless, SMS shows the best and most robust performance across a number of objectives $M \in \{2, 3, 5\}$. By its turn, IBEA is the most robust algorithm concerning the full range of objectives we tested, even if SMS performs better for a significant number of problems.

In addition to all the insights discussed in this chapter, the extensive experimental campaign conducted in this work serves as a first step towards machine learning approaches that can further empower MOEA researchers. As previously discussed, all the data produced in this work is provided both as a detailed supplementary material analysis as well as raw data that can be used for mining by interested researchers. Our hope is that this initiative stirs a collective research effort from the MOEA community towards further analysis that can help us both better understand correlations between problem features and algorithmic component effectiveness, as well as devise prediction tools that can be used to improve MOEA performance on unseen real-world problems. Moreover, in the next chapter we further explore the knowledge obtained from this work in order to automatically generate state-of-the-art AutoMOEAs for multi-objective continuous optimization.

CHAPTER 5

Designing and understanding the state-of-the-art

In the context of many-objective optimization, the small performance improvements from more recent MOEAs over existing ones observed in the previous chapter can be explained by many different factors. A critical one concerns the difficulty of analyzing the vast literature on MOEAs, to understand which existing algorithmic components are effective and how to combine them. In particular, the number of different algorithm proposals is too large for researchers to properly assess their (dis)advantages in a practical way, a fact reinforced by the few experimental analyses available. More importantly, MOEAs are in general proposed as monolithic blocks, assuming that their components are equally effective and need to be jointly used. The assessment of MOEAs, however, seldom investigates how these individual components interact, or if other components proposed for existing MOEAs could provide more performance benefits to an algorithm than some of the ones being proposed. This is further evidenced by the number of MOEA frameworks currently available that consider algorithms from a monolithic view, rather than from a composable perspective [38, 39, 120].

In Chapter 3, we have proposed a component-wise template from which a researcher can easily instantiate several existing MOEAs, but also create novel MOEAs by freely combining the algorithmic components available in a repository. Clearly, the number of possible combinations of algorithmic components is even larger than the number of existing MOEAs, a fact that would reinforce the practicality issue already faced by researchers. However, we have also demonstrated that it is possible to use this component-wise template to apply an automatic algorithm design methodology that has been proven successful in other optimization fields [70, 133, 157]. In fact, results demonstrated similar effectiveness also for the application to MOEAs, with the AutoMOEAs created in Chapter 3 consistently outperforming the MOEAs from where the framework components had been gathered, both on continuous and combinatorial optimization problems.

Notwithstanding the very good results from our initial feasibility study, we strongly believe that this methodology can still be pushed further. More specifically, MOEAs proposed for the field of multi-objective continuous optimization are generally compared to a handful of other MOEAs believed to be high-performing. This way, the initial selection of MOEAs used to assemble our framework had to rely on the trends observed in the literature, rather than on sound, large-scale experimental assessments identifying a state-of-the-art in this field. In the analysis conducted in the previous chapter, for instance, we have seen that using a different underlying operator more suited to the context of continuous opti-

mization can significantly boost the performance of MOEAs. In addition, we have noticed that some recently proposed MOEAs do not present an empirical performance matching what their proposers had expected. In a large part, this is explained by the contradictions between performance metrics, which become further evidenced in the context of many-objective optimization.

In this chapter, we push the automatic MOEA design to further improve the state-of-the-art identified in the previous chapter. In particular, we believe that our proposed methodology has the potential to devise MOEAs with state-of-the-art performance for given application domains, as long as properly enriched with components that are reportedly effective for those domains. The primary goal of this chapter is, hence, to automatically devise state-of-the-art MOEAs for multi- and many-objective continuous optimization. Given the insights obtained from the assessment conducted in the previous chapter, we see that this is a challenging task for two main reasons. First, continuous optimization is the main application domain for MOEAs, having been studied for nearly three decades. Second, the inherent challenges from many-objective optimization are still open issues that the MOEA community has not been able to properly solve.

To accomplish this goal, we extend our AutoMOEA template in a number of directions. First, we integrate a further level of composability that allows us to separate between the multi-objective related aspects of the search, and the underlying evolutionary algorithm adopted, like we proposed in Chapter 4. Effectively, this composability refinement greatly expands the design space provided by our framework, as any existing MOEA can be coupled with the most relevant EAs from the literature. Second, we extend our template to comprehend decomposition-based algorithms [59, 100, 124, 228, 230], in addition to the originally included dominance- [60, 234] and indicator-based [13, 19, 232] ones. Concretely, we implement components from relevant decomposition-based MOEAs, such as MOPSO [124] and NSGA-III [59], allowing the free hybridization between all three design paradigms considered. In fact, this is the first work to consider such possibility, and it is one of the major contributions of our study. Third, we extend the repository of available algorithmic components to include one of the most relevant archive truncation techniques found in the literature, namely the *adaptive grid* originally proposed for PAES [137]. Moreover, our modeling of this truncation procedure allows it to be used as a component of preference relations, being reusable for other selection procedures such as mating and environmental selection.

We empirically demonstrate the effectiveness of the AutoMOEA+ algorithms produced in this chapter by comparing them to the results of the investigation on the state-the-of-art in MOEAs for continuous optimization, conducted in Chapter 4. Overall, the AutoMOEA+ algorithms are able to consistently outperform the 15 MOEAs used for that investigation. More importantly, the designs of the AutoMOEA+ algorithms vary considerably according to the experimental scenario given, i.e., the number of objectives of the problems and the function evaluation (FE) budget provided to the algorithms. In addition, all novel components implemented in this chapter are selected by irace, and in many cases in ways that are very different from what human designers would tend to do. These results further evidence the need for flexible approaches that can be explored in a systematic, automated, and effective way, as we propose in this chapter. On a couple scenarios, though, the challenge posed by performance metric contradictions lead irace to select designs that are high-performing according to some metrics, but not according to others. To overcome this issue, we test alternative metrics to guide the automatic design process, and demonstrate that a multi-objective formulation of the tuning problem can be instrumental for the effectiveness of the selected designs.

Besides designing state-of-the-art MOEAs, a secondary goal of this chapter is to help improve the pace with which novel better-performing MOEAs are proposed. Our template-based automatic design methodology is certainly a means for this purpose, and we make it available for researchers and practitioners in general. Another step in this direction is the complementary investigation we conduct, deconstructing the AutoMOEA+ algorithms generated in this chapter to help understand why their designs are effective. Concretely, we select a subset of experimental scenarios and *ablate* [77] between the AutoMOEA+ algorithms and existing MOEAs. Our primary goal with these analyses is to understand

which algorithmic components from the AutoMOEA+ design contribute the most to its effectiveness, and whether such components could help improve other existing algorithms. Results show that some existing state-of-the-art algorithms can be improved by changing few components. In addition, we identify a number of alternative designs for each experimental scenario considered that are statistically equivalent to the AutoMOEA+ designed in this chapter. Finally, we also ablate between AutoMOEA+ algorithms to understand the interactions between experimental factors and components, an investigation that produces several relevant insights. Among the most interesting, we observe the existence of design space false plateaus, meaning that an algorithmic component can be considered indifferent by a given metric but improve/worsen performance according to another. More importantly, we empirically observe the benefits of the multi-objective tuning formulation, which is considerably robust w.r.t. these plateaus.

The main contributions of this chapter can be summarized as follows:

1. An augmented template for instantiating MOEAs that comprehends the most relevant underlying EAs, and design paradigms, namely dominance-, indicator-, and decomposition-based.

2. An empirical demonstration that state-of-the-art MOEAs can be automatically designed for multi- and many-objective optimization on several experimental scenarios, and that these designs combine elements from different design paradigms.

3. An iterative design-space analysis that helps understand the contribution of the individual components used by the automatically designed MOEAs, as well as interactions between experimental factors and the performance of these components.

The remainder of this chapter is organized as follows. In Section 5.1, we review the original AutoMOEA template and detail how we augment it in this chapter. In Section 5.2, we automatically design a set of MOEAs that display state-of-the-art performance. Next, Section 5.3 details the ablation analysis we conduct, and the most relevant insights are discussed in detail. Finally, we conclude and discuss future work in Section 5.4.

5.1 An augmented MOEA template

In this chapter, we augment our previously proposed AutoMOEA template in several directions. The first concerns the possibility of considering decomposition-based algorithmic components such as the ones proposed for MOPSO [124] and NSGA-III [59]. More importantly, our modeling of these components allow designers to combine, in a single algorithm, components originally proposed for dominance-, indicator-, and decomposition-based MOEAs. The second improvement over the original template concerns adopting the separation between multi-objective related components and the underlying EAs used, as proposed in Chapter 4. Concretely, our template allows the same set of MO components to be coupled with different EAs by simply changing the value of categorical parameters. By doing so, we increase the representativeness of the AutoMOEA template, since one can now instantiate many MOEAs that are based on differential evolution [1, 2, 143, 166, 201, 219], for instance. Finally, we also include in our framework one of the most relevant archive truncation approaches, originally proposed for PAES [137]. Next, we first review our original proposal, and then describe how we implement these improvements in more detail. We remark that, rather than requiring major structural changes to the existing template, the improvements we propose in this chapter benefit from its originally high flexibility, as we later discuss.

5.1.1 Original template

The main components of the original AutoMOEA template proposed in Chapter 3 can be briefly summarized as follows:

Preference is a *composite* component that encapsulates a sequence of three *atomic* components used in the following order. The first, SetPart, partitions solutions into dominance equivalent. The second component, Refinement, ranks solutions within each partition, in general by means of quality indicators. Finally, component Diversity is used to keep the population well-spread across the objective space. A Preference component can also contain less than three atomic components since SetPart, Refinement, and Diversity can be set to *none*.

BuildMatingPool uses traditional Selection operators to assemble a mating pool. In the case of tournaments, solutions are compared based on a preference relation Preference_{Mat}.

Replacement and Replacement_{Ext} components respectively define environmental selection and external archive truncation (if an archive is used). Both Replacement components ensure elitism, and comprise two other components: a preference relation (Preference_{Rep} and Preference_{Ext}, respectively), used to compare solutions, and Removal, a policy that determines the frequency with which Preference is computed.

Initialization and Variation encapsulate problem-specific components, respectively how to generate an initial population and the variation operators used to produce novel solutions.

5.1.2 Deconstructing decomposition

Decomposition [59, 100, 124, 228, 230] is a search paradigm originally considered by the decision making community and adapted for MOEA research already in its earliest years. The basic principle behind this paradigm is to decompose the original MOP into subproblems and then optimize them in parallel. In particular, each subproblem is a single-objective projection of the original MOP, obtained by the different methods discussed in Section 2.1. A deconstructive analysis of the decomposition-based MOEA literature reveals that most proposals can be classified as Refinement components. Specifically, most decomposition-based algorithms are able to simultaneously evaluate convergence and diversity, with the latter being ensured by the existence of multiple subproblems and the former by optimizing each subproblem. More importantly, decomposition approaches are able to distinguish between dominance-equivalent solutions, the baseline definition for our Refinement components. One exception to this pattern is NSGA-III [59], an algorithm that uses decomposition solely for diversity purposes. In particular, the convergence of NSGA-III is ensured by the same SetPart component as used by NSGA-II [60].

In this version of our template, we select two components to represent decomposition-based MOEAs. The first one is aliased *weighted ranking*, originally proposed for MOPSO [124], which is provided as an option for component Refinement. The second is the diversity approach proposed for NSGA-III, referred to as *referece lines* and provided as an option for component Diversity. We detail both components below:

Weighted ranking. Given a set of weights Λ, solutions are ranked according to their performance on the subproblems defined by each $\lambda \in \Lambda$. Overall, the quality of a solution equals its aggregated performance considering the ranks from each subproblem. In particular, several different functions to aggregate the performance on the subproblems can be used, and in this chapter we adopt the algebraic sum. Effectively, this refinement method corresponds to the rank sum analysis typically done in statistics, with individuals as treatments and subproblems as blocks.

Reference lines. This diversity approach uses reference lines to keep the population spread. Concretely, each weight vector $\lambda \in \Lambda$ represents a reference point, and the line intersecting this point and the origin of the axes is its associated reference line. In a three-step approach, the procedure first computes the distances between solutions and reference lines, and associates each solution with its nearest reference line. Next, the algorithm computes niche counts for each reference line considering only solutions

Table 5.1: Novel atomic components implemented in this chapter for the AutoMOEA template.

Component	Domain
Quality	{ *weighted ranking**
Diversity	{ *adaptive grid** { *reference lines**

Component	Domain
Variation*	{ *GA*, *DE* }
OnlineReplacement*	{ *none*, *Pareto*, *WeakPareto* }
Λ_{dist}*	{ *uniform*, *dichotomic*, *two-layer* }

Table 5.2: Novel composite components implemented in this chapter for the AutoMOEA template.

Component	Parameters
UnderlyingEA*	⟨ BuildMatingPool, Variation ⟩

(*) Novel component implemented in this chapter.

already selected for the next iteration by previous preference components[1]. Effectively, this step identifies which reference lines are already represented in the next iteration. Finally, the procedure iteratively selects the reference line with lowest niche count and adds one of its associated solutions to the next generation population.

A second feature concerning decomposition-based MOEAs that is as important as the components we model above are the cardinality and distribution of the weight set Λ used by these approaches. Concerning cardinality, a numerical parameter Λ_r is used to set an upper bound $\Lambda_r \cdot \mu$ to the cardinality of Λ, where μ is the population size. However, the actual number of weights generated by a distribution may exceed this upper bound, in which case exceeding weights are discarded at random. In this chapter, we implement as baseline option the uniform distribution traditionally used in the literature [53]. However, for specific scenarios some alternative distribution approaches have been proposed and may be more suited depending on the given application. These alternatives are summarized in Table 5.1, and comprise (i) the *dichotomic* approach proposed by Aneja and Nair [7] for bi-objective scenarios, and (ii) the two layer approach proposed for many-objective scenarios by the authors of NSGA-III. In particular, this latter approach uses two numerical parameters H_1 and H_2 to determine how many weights will be generated using a uniform distribution in the outer and inner layers, respectively. In this chapter, we propose an automatic way of setting these parameters depending on the search focus a designer wants to use, as follows:

Peripheral focus favors the outer layer. More precisely, the algorithm sets H_1 to the maximum value feasible so that there exists an H_2 value for which $|\Lambda| \leq \Lambda_r \cdot \mu$.

Central focus favors the inner layer, and is the opposite of peripheral focus. In more detail, the algorithm sets H_2 to the maximum value feasible so that there exists an H_1 value for which $|\Lambda| \leq \Lambda_r \cdot \mu$.

Balanced focus favors neither layer. Concretely, H_1 and H_2 are set to their maximum feasible values so that $|\Lambda| \leq \Lambda_r \cdot \mu$. If, however, it is still possible to increase either H_1 or H_2, that parameter is increased to prevent wasting weights.

5.1.3 Underlying EAs

The literature on the application of different underlying EAs to the context of multi-objective optimization is rather restricted, as most of the studies on MOEAs have concentrated on devising MO

[1] For instance, in the case of dominance depth used as SetPart by NSGA-III, the solutions from the lowest depth fronts that fit in the next population.

components. In this chapter, we extend our template to encompass the possibility of freely combining MO components and different underlying EAs. In particular, we consider the two most relevant EAs used for continuous optimization, namely *genetic algorithms* (GAs, [96]) and *differential evolution* (DE, [193]). We model the underlying EA as a composite component UnderlyingEA, which comprises composite components BuildMatingPool and Variation (see Table 5.2). In particular, we do so as EAs differ not only as to the operators used for variation, but also on how to select individuals to undergo variation. Below we further describe how different EAs are modeled.

Genetic algorithms are the default option in the original AutoMOEA template, so they are instantiated in a straightforward way. Specifically, a mating pool is built as described by component BuildMatingPool. Variation comprises the sequential application of crossover and mutation operators, represented by option *GA variation* (see Table 5.1). Domain-specific operators are used as in the literature, in this case the SBX crossover and the polynomial mutation operators.

Differential evolution proposals for multi-objective optimization traditionally adopt the *DE/rand/1* scheme [1, 2, 143, 166, 193, 201, 219]. To instantiate this option, one only needs to configure component BuildMatingPool to use random selection and component Variation to *DE variation*. In more detail, differential mutation and binomial crossover are applied, and vectors are selected at random. Besides the traditional *DE/rand/1* scheme, we also propose a preference-based selection scheme, which can be understood as an adaptation of the *DE/target-to-best/1* scheme [193]. Concretely, we allow designers to use any of the Selection options available in the template for selecting vectors to be used by DE, coupled with a Preference_{Mat} component. Effectively, such a scheme differs from the default used by multi-objective DE algorithms in that vectors that rank well according to Preference_{Mat} will more likely be used in the variation process, either as trial or donor vectors.

Another DE-related component we implement in the AutoMOEA template has been proposed specifically for multi-objective optimization and can be understood as an *online* replacement strategy [143, 166, 201]. More precisely, when this strategy is used a newly created trial solution can immediately replace the target vector in case a given acceptance criterion is satisfied. For instance, the most relevant DE algorithms differ exactly by this acceptance criterion, as DEMO [201] considers Pareto dominance and the GDE3 [143] considers weak Pareto dominance. When online replacement is not used or when trial and target vectors are nondominated, the trial vector is added to the population and may only be discarded at the end of the generation by the environmental selection procedure Replacement. Available acceptance criteria are listed in Table 5.1. Finally, we remark that online replacement cannot be used (i) when preference-based mating selection is adopted, since preferences are only computed before mating starts, nor (ii) when steady state selection ($\lambda = 1$) is adopted, since it becomes equivalent to the environmental replacement.

5.1.4 Archiver-specific truncation techniques

As discussed in Section 2.2.4, many different archiver truncation techniques have been proposed. In our original framework, two of the techniques considered in the review conducted by López-Ibáñez et al. [160] can already be instantiated: (i) the dominating archiver, the baseline archive we consider, and (ii) the hypervolume archiver (AA_s, [135]), that can be obtained if one considers a Preference component comprising only the I_H^1 as refinement metric. In this chapter, we add to the AutoMOEA framework the archive truncation method proposed by PAES [137], namely the *adaptive grid* approach. Specifically, this archiver discretizes the objective space into grid cells that are dynamically computed as a function of the extreme solutions found during the run, and of a discretizing numerical parameter that regulates the number of divisions per objective. Solutions are compared based on the crowdedness of the grid cell to which they belong, with less crowded regions being favored. Given its characteristics, we implement the adaptive grid approach as a diversity component. We remark that, due to the flexibility of our template,

Table 5.3: Experimental factors used for the design of the AutoMOEA+ algorithms.

Factor	Domain
Benchmark set	DTLZ1–7 ∪ WFG1–9
M	{2, 3, 5}
$FE_{\max}$	{2 500, 10 000, 40 000}
$n_{testing}$	{30, 40, 50}
n_{var}	$\{20, 21, \ldots, 60\}$
n_{tuning}	$n_{\text{var}} \setminus n_{testing}$
t_{max}	1h ($FE_{\max} = 2\,500$), 10min (otherwise)
$\mathbf{u}$	$\begin{cases} [10]^M, & \text{if } M \in \{2,3\} \\ [15]^5, & \text{if } M = 5 \\ [25]^{10}, & \text{if } M = 10 \end{cases}$

Factor	Domain
$\mathbf{r}$	$\alpha \cdot \mathbf{u}$
α	1.1
Tuner	irace [159]
Tuning budget	20 000 experiments
Tuning metric	$I_{\epsilon+}$ ($M = 10$), I_H^{rpd} (otherwise)
Statistical test	Friedman non-parametric test and post-hoc
Test metrics	I_H^{rpd}, $I_{\epsilon+}$, I_{IGD}
Test repetitions	25

it is possible to use the adaptive grid approach in different preference components, such as the one used for mating or for the internal archive replacement. In addition, it is possible to combine an internal archive using the adaptive grid approach of PAES with an external archive using a decomposition-based diversity component such as the one proposed for NSGA-III. Altogether, the representativeness and expressivity of our template is greatly improved by the changes we propose in this chapter, translating into effective MOEAs as we discuss next.

5.2 Automatically designing effective MOEAs

The experimental investigation we describe in this section has three main goals. The first is to analyze patterns in the structure of the automatically designed MOEAs (hereon called AutoMOEA+ algorithms) to understand to what extent their designs match what human designers would choose as effective. Second, we compare the structure and performance of the AutoMOEA+ algorithms with the structure and performance from the AutoMOEAs proposed in Chapter 3. Third, we assess how the AutoMOEA+ algorithms perform compared to the state-of-the-art in MOEAs identified in Chapter 4. Finally, we propose improved tuning approaches to design better-performing AutoMOEA+ algorithms for specific scenarios, analyzing their structure and assessing their effectiveness. Next, we detail the experimental setup we adopt in the rest of the section.

5.2.1 Design setup

Since we are comparing to the state-of-the-art in MOEAs previously identified in Chapter 4, we follow the same experimental settings, briefly summarized in Table 5.3. In particular, we remark that experimental scenarios are obtained by the combinations of M and $FE_{\max}$ values, and are referred to using the $\langle M,\ FE_{\max}\rangle$ notation[2]. Since MOEAs have been tuned for each scenario, we design an AutoMOEA+ for each scenario, totalizing 12 AutoMOEA+ algorithms. Experiments are run on a single core of Intel Xeon E5410 CPUs @ 2.33GHz with 6MB cache size under Cluster Rocks Linux version 6.2/CentOS 6.2.

The parameter space we use for designing each AutoMOEA+ is given in Tables 3.1, 3.2, 5.2, 5.1, and 5.4. In particular, parameters depicted in Table 5.4 under column GA are only used when GA variation is selected, whereas parameters under column DE are used otherwise. Concerning GA variation, parameters p_c and $p_m \in [0, 1]$ respectively stand for the probability of applying the SBX crossover to a given pair

[2]For short, the $FE_{\max}$ value is presented in thousands of FEs. More precisely, a scenario $\langle 2,\ 2.5\rangle$ means $M = 2$ and $FE_{\max} = 2.5 \times 10^3 = 2\,500$.

Table 5.4: Parameter space for tuning numerical parameters of the AutoMOEA template.

		GA		DE	
$\mu = \lvert\texttt{pop}\rvert$	$\lambda = \lvert\texttt{pop}_{\texttt{new}}\rvert$	p_c, p_m	η_c, η_m	CR	F
$\{10, 20, \ldots, 100\}$	1 or $\lambda_r \cdot \mu$ $\lambda_r \in [0.1, 2]$	$[0, 1]$	$\{1, \ldots, 50\}$	$[0.01, 1]$	$[0.1, 2]$

Condition	**Additional param.**	**Domain**
type $(\texttt{pop}) =$ *bounded*	μ_r	$[0.1, 2]$
type $(\texttt{pop}_{\texttt{ext}}) =$ *bounded*	N_{ext}	$\{100, 300, 500\}$
mutation scheme = *fixed*	p_v	$[0.01, 1]$
Selection $= ST$	*tourn. size*	$\{2, 4, 8\}$
Selection $= ST$	γ	$[0.6, 0.9]$
Refine = *binary indicator*	*indicator*	$I_{\epsilon+}$, I_H^-
Diversity = *sharing*	σ_{share}	$[0.1, 1]$
Diversity = *adaptive grid*	l	$\{1, \ldots, 4\}$
$\text{Diversity}_{Mat} = kNN$	k_{method}	$\{default, k\}$ $k \in \{1, \ldots, 9\}$
Refine = *weighted ranking* or Diversity = *axial dist.*	$\Lambda_{params} = \langle \Lambda_r, \Lambda_{dist} \rangle$	$\Lambda_r \in [0.5, 2]$
$\Lambda_{dist} =$ *two-layer*	Λ_{focus}	$\begin{cases} peripheral, \\ central, \\ balanced \end{cases}$

of individuals, and the probability of applying polynomial mutation to a given individual. In addition, these operators have associated distribution indices η_c and η_m that must also be configured. Finally, different mutation schemes have been used in the GA literature for real-parameter optimization [58], two of which are implemented here: (i) *bitwise*, which sets the mutation probability per variable p_v to $1/n_{\text{var}}$; and (ii) *fixed*, where p_v must be configured with domain $[0.01, 1]$. Conversely, when DE variation is selected, only two parameters must be set, namely CR and F, respectively representing the crossover probability and the scaling factor of the DE operators.

5.2.2 Structural analysis

The designs of the automatically designed AutoMOEA+ algorithms are given in Table 5.5[3]. Given the similarities between the designs of the AutoMOEA+ algorithms devised for $M \in \{2, 3, 5\}$ scenarios, we first discuss this pattern, grouped by the most relevant components:

UnderlyingEA: Preference-based DE variation is always adopted. This is a remarkable fact since no work on multi-objective differential evolution had considered an adaptation of the DE/target-to-best/1 scheme so far. Concerning numerical parameters, the ranges of CR and F vary slightly but it is clear that lower values provide better results. Finally, the online replacement strategy is never selected since preference-based selection is always used.

type (pop): The choice between using a regular population or a bounded internal archive greatly depends on the experimental scenario, as shown by the nearly even division between AutoMOEA+ algorithms

[3]For brevity, some design choices are represented implicitly, as follows. First, if DE is used as underlying EA, its numerical parameters are depicted under column UnderlyingEA. In addition, the *DE/target-to-best/1* scheme is adopted unless Selection = *random*. Second, if μ_r is set to a numerical value, *type* (pop) = *bounded*. Finally, $\lambda = 1$ indicates steady-state selection.

Table 5.5: Parameters selected by irace for the AutoMOEA+ algorithms.

	BuildMatingPool				Replacement				Replacement$_{Ext}$				Numerical				UnderlyingEA		
	Selection	SetPart	Refine	Diversity	SetPart	Refine	Diversity	Removal	N_{ext}	Refine	Diversity	Removal	μ	μ_r	λ	λ_r	CR	F	Online
$\langle 2,\ 2.5\rangle$	DT(8)	strength	$I_{\epsilon+}$	—	—	I_H^h	—	—	300	I_H^h	kNN	1-shot	60	—	1	—	0.57	0.56	—
$\langle 2,\ 10\rangle$	DT(4)	depth	I_H^1	kNN	DR	I_H^h	—	—	—	—	—	—	100	0.44	1	—	0.39	0.49	—
$\langle 2,\ 40\rangle$	ST(0.79)	—	—	sharing	—	$I_{\epsilon+}$	sharing	seq.	500	I_H^h	crowd.	1-shot	60	0.84	—	0.61	0.28	0.51	—
$\langle 3,\ 2.5\rangle$	DT(8)	DR	I_H^1	crowding	rank	I_H^1	kNN	—	500	I_H^h	ref. lines	1-shot	60	—	1	—	0.19	0.73	—
$\langle 3,\ 10\rangle$	ST(0.77)	—	I_H^1	grid(1)	—	I_H^1	grid(2)	seq.	500	—	grid(3)	seq.	40	—	—	1.64	0.28	0.44	—
$\langle 3,\ 40\rangle$	DT(2)	—	I_H^1	—	—	I_H^1	—	seq.	500	—	sharing	seq.	90	0.76	—	0.88	0.19	0.3	—
$\langle 5,\ 2.5\rangle$	DT(4)	depth	w-rank	crowd.	rank	I_H^1	crowd.	—	300	—	grid(2)	1-shot	30	—	1	—	0.37	0.65	—
$\langle 5,\ 10\rangle$	DT(4)	DR	w-rank	grid(1)	depth	I_H^1	sharing	—	—	—	—	—	70	—	1	—	0.26	0.46	—
$\langle 5,\ 40\rangle$	ST(0.82)	—	I_H^1	kNN(8)	—	I_H^1	crowd.	seq.	—	—	—	—	60	—	—	0.51	0.05	0.47	—
$\langle 10,\ 2.5\rangle$	random	—	—	—	—	$I_{\epsilon+}$	kNN	—	500	w-rank	ref. lines	1-shot	50	—	1	—	0.49	0.24	—
$\langle 10,\ 10\rangle$	random	—	—	—	rank	$I_{\epsilon+}$	sharing	—	500	I_H^h	ref. lines	1-shot	90	—	—	0.9	—	—	—
$\langle 10,\ 40\rangle$	ST(0.84)	count	—	sharing	DR	—	ref. lines	1-shot	—	—	—	—	20	—	1	—	—	—	—

(All AutoMOEA+ algorithms use DE as underlying EA. $\Lambda_{params}(\text{Diversity}_{Ext}) = \langle 1.15, \textit{uniform}\rangle$ for AutoMOEA+$_{\langle 3,\ 2.5\rangle}$, $\Lambda_{params}(\text{Refine}_{Mat}) = \langle 1.54, \textit{two-layer}, \textit{peripheral}\rangle$ for AutoMOEA+$_{\langle 5,\ 2.5\rangle}$, and $\Lambda_{params}(\text{Refine}_{Mat}) = \langle 0.98, \textit{uniform}\rangle$ for AutoMOEA+$_{\langle 5,\ 10\rangle}$. σ_{share} values for given AutoMOEA+ algorithms are provided as supplementary material.)

that use each type. In particular, we notice that μ values do not depend directly of this choice, as the range of the selected values for this parameter is considerable.

BuildMatingPool: Tournament Selection is always chosen, although much more often deterministic than stochastic. One possible explanation is that in our template deterministic tournaments can be n-ary, and can hence provide more convergence pressure. Concerning Preference$_{Mat}$, nearly all AutoMOEA+ algorithms use relations comprising SetPart, Refinement, and Diversity components, except when an internal archive is used instead of a regular population. In particular, we remark that for $M = 5$ two AutoMOEA+ algorithms use preference relations that contain elements from both dominance- and decomposition-based MOEAs. In addition, the adaptive grid component from PAES is used for mating selection twice. Overall, it is difficult to find a pattern in the components that comprise the selected preference relations, the only exception being the I_H^1 often used for refinement. This absence of a more clearly defined mating preference pattern reinforces our argument for flexible approaches to MOEAs.

Replacement: In contrast to BuildMatingPool, it is far easier to find patterns in component Replacement. We first remark the number of AutoMOEA+ algorithms that use steady-state selection combined with a hypervolume-based Refinement. This combination was originally proposed by SMS to reduce the overhead posed by this indicator and proves successful once again here. In addition, the overhead posed by this indicator is further reduced by the use of a set-partitioning relation, a design choice only disregarded when an internal archive is selected over a regular population. Concerning component Removal, sequential replacement is adopted whenever steady-state selection is not used. Finally, the only component for which no clear pattern can be observed is Diversity, although we remark that it is interesting that the decomposition-based option is never selected.

Replacement$_{Ext}$: Most AutoMOEA+ algorithms use a large external archive, the most notable exception being the scenarios with $M = 5$. The most straightforward explanation concerns the overhead posed by the archive, a fact reinforced by the Preference$_{Ext}$ component often comprising only Diversity. Concerning Diversity, all algorithmic options available are adopted, corroborating the need for a diverse repository that can adapt to different scenarios. This time, however, the decomposition-based reference lines approach is selected for scenario $\langle 3, 2.5 \rangle$, coupled with the I_H^h indicator. We remark that such a combination of an indicator-based refinement with a decomposition-based diversity metric had never been proposed in the literature. Finally, we remark that the I_H^h indicator is often selected as Refinement, but always coupled with one-shot Removal due to its computational complexity. Moreover, whenever Preference$_{Ext}$ comprises solely a Diversity component, sequential removal is selected since the overhead of this preference relation is minimal.

In contrast to the design patterns discussed above, the designs of the AutoMOEA+ algorithms for scenarios with $M = 10$ can be summarized as follows. First, most designs are GA-based, a fact that can be explained the by computational overhead posed by the DE-based designs previously observed. In more detail, the DE/target-to-best/1 scheme combined with hypervolume-based refinements proved an effective combination for other scenarios. When the number of objectives is increased to ten, however, the I_H^1 poses a significant computational overhead, and hence the whole design of the AutoMOEA+ algorithms changes accordingly. A second design choice that is often adopted by these algorithms concerns decomposition-based diversity, which is used for environmental selection by AutoMOEA+$_{\langle 10, 40 \rangle}$ and for archive truncation by the other scenarios. Regarding Preference$_{Ext}$, it is interesting to observe how two components from the decomposition paradigm are selected for AutoMOEA+$_{\langle 10, 2.5 \rangle}$, and how quality indicators and decomposition-based components are once again combined in AutoMOEA+$_{\langle 10, 10 \rangle}$. Finally, the design of AutoMOEA+$_{\langle 10, 40 \rangle}$ is the one that differs the most from all other AutoMOEA+ designs. Specifically, this algorithm resembles NSGA-III in its randomized mating selection, absence of refinement metrics, and use of GA. However, given that no external archive is used, the population size is

rather small for a many-objective scenario. As we will later discuss, this design suffers from the same issues observed in MOEA/D, conducting a search that is too restricted in the objecive space and hence can only be considered good according to the $I_{\epsilon+}$ indicator.

We next discuss the similarities and differences in the structure of AutoMOEA+ algorithms focusing on the experimental factors that constitute scenarios:

M: besides the previously discussed differences between $M = 10$ and the remaining scenarios, the only remarkable pattern in the structure of the AutoMOEA+ algorithms grouped by M concerns component Refinement. More precisely, the I_H^h component is clearly a suitable component for replacement on bi-objective scenarios. By contrast, on $M \in \{3, 5\}$ scenarios the I_H^1 indicator becomes a more suitable choice. The most likely explanation concerns the computational overhead of the I_H^h indicator since, when $M = 3$, no Refinement component is used for the external archives except for the scenario with a larger cutoff time. When $M = 10$, $I_{\epsilon+}$ becomes the standard refinement option for Preference_{Rep}, but further investigation would be required to determine whether this is a consequence of changing the tuning metric. Finally, we highlight that for $M = 5$ component weighted rank is the most selected Refinement option for mating selection, and is also used by $\text{AutoMOEA+}_{\langle 10,\,40\rangle}$.

$FE_{\max}$: concerning the effects of $FE_{\max}$, the most evident insight we observe is that external archives tend to become prohibitive when this budget is increased but the maximum runtime is kept constrained. This is initially observed for scenarios with $M = 3$, where only Diversity components are used when $FE_{\max} \in \{10\,000, 40\,000\}$, and made worse on scenarios with $M = 5$, where external archives are not used at all. The extreme situation is observed for $\text{AutoMOEA+}_{\langle 10,\,40\rangle}$, where no refinement metrics nor external archives are used. Overall, when one analyzes solely scenarios with $FE_{\max} = 2\,500$, it is clear that the external archives selected for these AutoMOEA+ algorithms are far more computationally demanding than the options selected for the remaining scenarios.

5.2.3 Comparison to the original template

To assess the improvements provided by the extensions proposed in this chapter, we compare the AutoMOEA+ algorithms designed in this chapter to the AutoMOEAs designed in Chapter 3. In particular, those AutoMOEAs were created for scenarios with $FE_{\max} = 10\,000$ and $M \in \{2, 3, 5\}$, and for this reason only these scenarios are considered in this comparison. In addition, those AutoMOEAs are benchmark-specific, and hence benefit from tuning much more than the AutoMOEA+ algorithms or the state-of-the-art MOEAs. For this reason, we disregard the conclusions that could be drawn from a comparison between the AutoMOEAs and the state-of-the-art. Moreover, we remark that results in favor of the AutoMOEA+ algorithms are yet more remarkable for that matter.

Concerning the structural comparison of the two AutoMOEA sets, we group this discussion by the most relevant components:

UnderlyingEA: As previously discussed, DE is always used in the AutoMOEA+ algorithms. This design choice highlights the importance of providing different underlying EAs for a component-wise design.

type (`pop`): The bounded internal archive is selected far more often for the AutoMOEA+ algorithms than for the original AutoMOEAs. In particular, this design choice had only been selected for a few DTLZ scenarios and denotes that different benchmark sets may require different designs.

BuildMatingPool: Selection approaches are similar between AutoMOEA sets, since tournaments are always used. However, the tournaments from the original AutoMOEAs are deterministic and enforce greater convergence pressure since eight-ary tournaments are adopted. This design difference is likely explained by the different underlying EAs used. Concerning Preference_{Mat}, the original AutoMOEAs often used crowding diversity, whereas the AutoMOEA+ algorithms use several different diversity metrics. In

addition, while the I_H^1 was always adopted in the AutoMOEAs, we notice that AutoMOEA+$_{\langle 5,\,10\rangle}$ uses a decomposition-based Refinement instead.

Replacement: Environmental selection from both AutoMOEA sets are very similar. In fact, the only remarkable difference is that in the original AutoMOEAs niche sharing was adopted when a Diversity component was selected, whereas in the AutoMOEA+ algorithms we see the adaptive grid component being selected for scenario $\langle 3,\,10\rangle$.

Replacement$_{Ext}$: The biggest difference between the two AutoMOEA sets. While AutoMOEAs always use external archives, only AutoMOEA+$_{\langle 3,\,10\rangle}$ uses this design choice.

The rank sum analysis given in Table 5.6 shows that the rank sums achieved by the AutoMOEA+ algorithms is statistically significantly better than the sums achieved by the original AutoMOEAs when $M \in \{2, 3\}$, whichever the metric considered. In particular, we see from Figure 5.1 that the original AutoMOEAs are only able to improve over the AutoMOEA+ algorithms on the concave WFG problems. When $M = 3$, we see a similar pattern on Figure 5.2 (left), but only for the I_H^{rpd} metric. For all other metrics, the performance of the AutoMOEA+ algorithms is far better than that of the original AutoMOEAs.

The scenario where results differ the most is the $\langle 5,\,10\rangle$ scenario. In part, the lower rank sums achieved by the AutoMOEAs are explained by the previously remarked disagreements between the performance metrics. On the other hand, the overall good performance of the original AutoMOEAs indicate that for larger M values the benchmark set used for tuning plays a critical role. On Figure 5.2 (right), we see that the original AutoMOEAs outperform AutoMOEA+ on DTLZ7 and also on the concave WFG problems when one considers the I_H^{rpd} and the I_{IGD} metrics. Regarding the $I_{\epsilon+}$, we notice that the performance of the AutoMOEAs is considered better than that of the AutoMOEA+, a difference explained by the disagreement between the $I_{\epsilon+}$ and the remaining metrics on convex problems with large M values.

5.2.4 State-of-the-art comparison

A rank sum analysis depicting the comparison of the AutoMOEA+ algorithms with state-of-the-art algorithms is given in Table 5.6. In particular, we first analyze results for $FE_{\max} = 10\,000$, and next we make a more general analysis on other $FE_{\max}$ settings. In addition, Table 5.6 includes the original AutoMOEAs but, as previously discussed, this comparison should be disregarded.

Concerning the comparison of the performances of the AutoMOEA+ algorithms with the state-of-the-art in MOEAs for continuous optimization, we notice different results depending on the given performance metric. Concerning the I_H^{rpd}, used for tuning, the performance of the AutoMOEA+ algorithms is always equivalent to that of the best-performing MOEA, which varies depending on the given scenario. Concerning the remaining metrics, the AutoMOEA+ algorithms are always statistically significant better than the best-ranked MOEA. The only exception to this pattern is seen for the $I_{\epsilon+}$ when $M = 5$, where no statistically significant difference can be observed between the AutoMOEA+ and SMS. This fact is explained by the divergences between metrics previously reported that becomes more significant with the increase in M. Although these divergences affect all MOEAs, the automatic design approach is far more sensitive to this issue.

Boxplots depicting the performance of MOEAs on selected problems and different M scenarios are given in Figures 5.1 and 5.2. In particular, we remark that the problems selected are representative of problem subclasses, summarized as follows. DLTZ2 represents the easier functions, comprising DTLZ2, DTLZ4, and DTLZ5, although DTLZ4 can be considered moderate for scenarios with $M = 3$. DTLZ6 and DTLZ7 comprise the hardest DTLZ functions, although the difficulty level of the latter is only substantially increased as M grows. Regarding WFG functions, the non-concave WFG functions are represented by WFG1, whereas the remaining functions are represented by WFG4. For brevity, we remark that only $n_{\text{var}} = 40$ problems are depicted, although these are representative of other n_{var} sizes.

Table 5.6: Rank sum difference (in parenthesis) between the given MOEA and the lowest ranked ($FE_{\max} = 10\,000$). MOEAs highlighted in boldface present rank sums statistically significantly lower than the others according to Friedman's nonparametrical test.

					$M = 2$					
I_H^{rpd}	**Auto+**	**IBEA** (37)	SMS (56)	Auto (92)	SPEA2 (99)	NSGA-II (113)	CMA (193)	HypE (194)	MOEA/D (196)	NSGA-III (237)
$I_{\epsilon+}$	**Auto+**	SMS (59)	IBEA (64)	SPEA2 (99)	NSGA-II (129)	Auto (141)	HypE (186)	CMA (210)	MOEA/D (220)	NSGA-III (248)
I_{IGD}	**Auto+**	SMS (48)	SPEA2 (75)	IBEA (104)	HypE (139)	NSGA-II (157)	Auto (160)	CMA (268)	NSGA-III (268)	MOEA/D (277)
					$M = 3$					
I_H^{rpd}	**Auto+**	**SMS** (34)	IBEA (75)	Auto (79)	SPEA2 (143)	HypE (158)	MOEA/D (159)	CMA (204)	NSGA-II (248)	NSGA-III (253)
$I_{\epsilon+}$	**Auto+**	SMS (70)	IBEA (96)	Auto (166)	SPEA2 (177)	CMA (208)	HypE (223)	MOEA/D (252)	NSGA-III (267)	NSGA-II (293)
I_{IGD}	**Auto+**	IBEA (88)	SMS (118)	Auto (129)	SPEA2 (139)	MOEA/D (187)	HypE (213)	NSGA-II (252)	CMA (278)	NSGA-III (280)
					$M = 5$					
I_H^{rpd}	**Auto+**	**SMS** (3)	**Auto** (26)	MOEA/D (99)	IBEA (109)	CMA (165)	SPEA2 (172)	NSGA-II (201)	NSGA-III (218)	HypE (267)
$I_{\epsilon+}$	**Auto**	**Auto+** (10)	**SMS** (31)	IBEA (104)	MOEA/D (138)	CMA (146)	NSGA-II (209)	NSGA-III (231)	SPEA2 (241)	HypE (283)
I_{IGD}	**Auto**	**Auto+** (17)	SMS (65)	IBEA (146)	NSGA-II (163)	MOEA/D (167)	CMA (177)	SPEA2 (225)	HypE (280)	NSGA-III (300)

				$M = 10$					
I_H^{rpd}	**IBEA** (0)	**SMS** (13)	**Auto+** (39)	CMA (63)	SPEA2 (123)	NSGA-III (124)	NSGA-II (164)	MOEA/D (230)	HypE (234)
$I_{\epsilon+}$	**MOEA/D** (0)	**Auto+** (39)	IBEA (67)	SMS (148)	CMA (150)	NSGA-III (158)	NSGA-II (176)	SPEA2 (206)	HypE (244)
I_{IGD}	**Auto+** (0)	NSGA-III (65)	IBEA (67)	SPEA2 (96)	CMA (140)	NSGA-II (140)	SMS (142)	HypE (173)	MOEA/D (230)

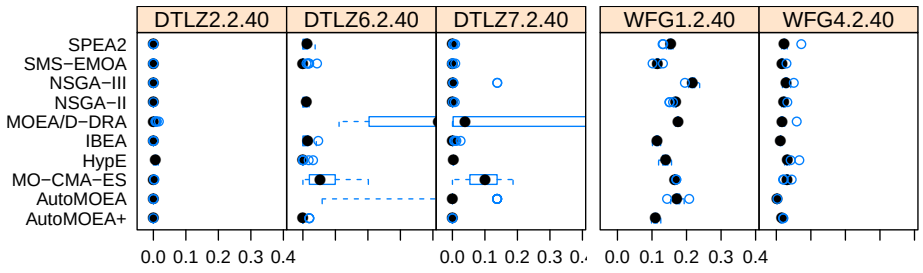


Figure 5.1: Performances of MOEAs given 10 000 FEs on selected two-objective problems with 40 variables according to the I_H^{rpd}.

Moreover, when $M = 2$ metrics are very consistent and so only I_H^{rpd} results are given. The full set of results is provided as supplementary material [34].

Concerning results for $M = 2$, we notice that most algorithms are able to display good performance on the DTLZ problems. The most significant difference concerns DTLZ6, where the AutoMOEA+, SMS, and HypE perform best. Regarding the WFG problems, the performance of the AutoMOEA+ is clearly better than the performance of the remaining state-of-the-art MOEAs. In fact, we identify that the best performance of the AutoMOEA+ algorithms is seen for the WFG set, an observation related to the predominance of WFG functions on the test benchmark set considered. When $M = 3$, we again see a very good performance of the AutoMOEA+ algorithm. Concerning DTLZ problems, the only state-of-the-art MOEA that displays performance equivalent to that of the AutoMOEA+ is HypE. However, when one analyzes the WFG functions we notice that none of the state-of-the-art MOEAs is able to display a performance as good as that of the AutoMOEA+. The only exception concerns SMS, which is able to seldom outperform the AutoMOEA+ on non-concave functions according to I_H^{rpd}. We also remark that the good performance of the AutoMOEA+ is even more evident for metrics for which it was not tuned. As previously discussed, this is explained by the strong agreement between metrics on scenarios with few objectives.

Finally, results on scenario $\langle 5, 10 \rangle$ are depicted in Figure 5.2 (right). On DTLZ problems, we notice two different situations. For the problems represented by DTLZ2 and for DTLZ6, we notice that some state-of-the-art MOEAs are able to present performance equivalent to AutoMOEA+, but according to the I_{IGD} the performance of the latter cannot be matched. On DTLZ7, the AutoMOEA+ performs much worse than the remaining MOEAs. This drawback is understandable when one observes that the automatic design methodology considers benchmarks as a whole, and specific functions may constitute exceptions to a broader picture, as in this case. Regarding WFG problems, the improvements achieved by AutoMOEA+ over other MOEAs is remarkable in two situations. The first concerns the I_H^{rpd} for non-concave problems. Although the same exceptional situation is not observed for the other metrics, AutoMOEA+ is only outperformed by SMS. The second remarkable result concerns the concave problems, for which the performance of AutoMOEA+ is far better than that of the remaining MOEAs according to the $I_{\epsilon+}$ and I_{IGD} metrics. For the I_H^{rpd}, differences are not so high, but only IBEA is able to present a nearly equivalent performance to that of the AutoMOEA+ algorithm.

We next discuss the effects of the other $FE_{\max}$ values:

2 500 FEs. When only a limited number of FEs is given to MOEAs, the differences between the best-performing algorithms are reduced. In fact, in many scenarios it is not possible to identify statistical significant differences between the performances of the AutoMOEA+ algorithms, SMS, and IBEA. Nonetheless, for $M = 3$ and $M = 5$ scenarios we observe that the I_{IGD} performance of the AutoMOEA+ algorithms is considered statistically significantly better than that of the remaining MOEAs. Altogether, these results indicate that it is possible to design more robust MOEAs w.r.t. different performance metrics, even for scenarios where few FEs are available. However, it is clear that MOEAs still require the development of algorithmic components that are better suited for these scenarios, as done

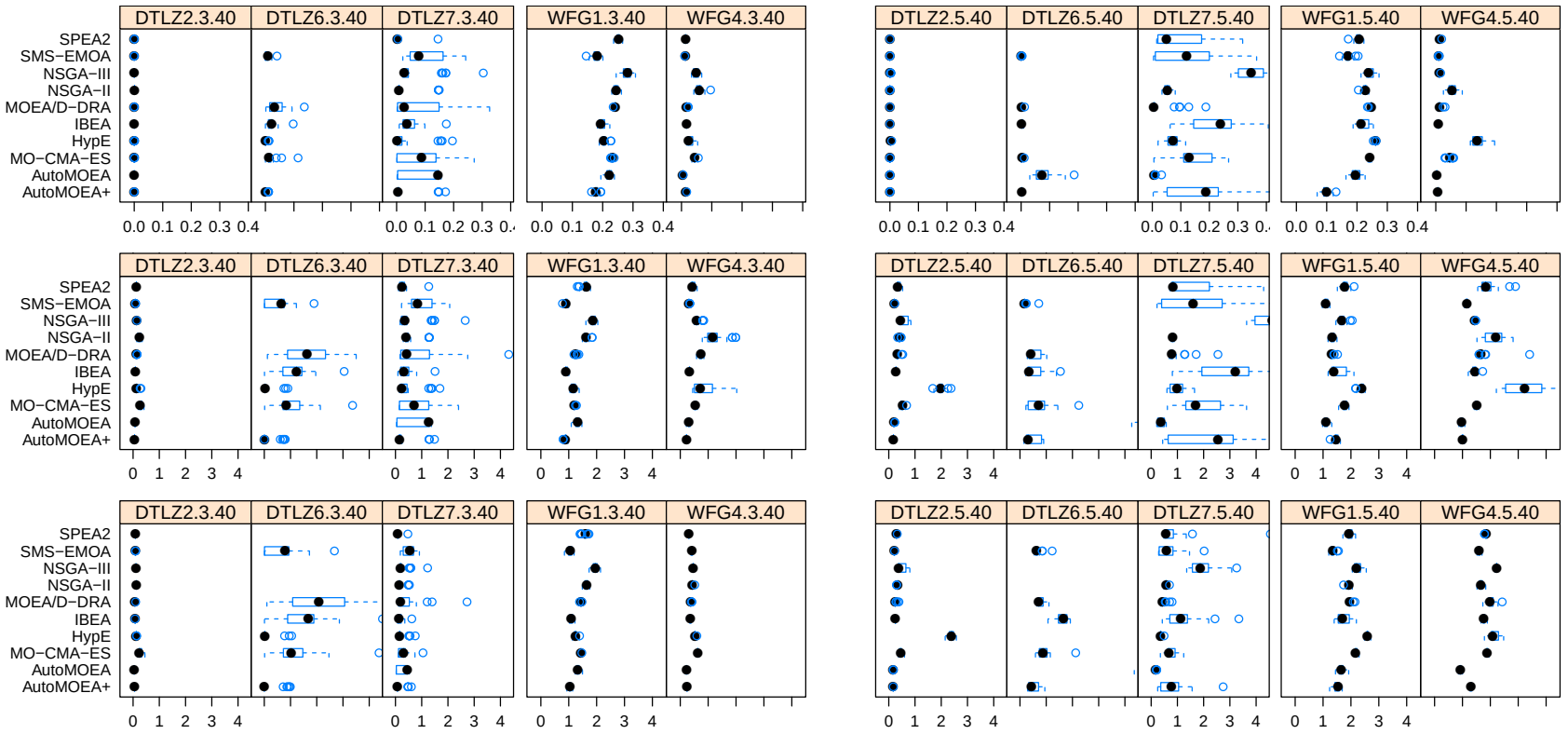


Figure 5.2: Performances of MOEAs on the $\langle 3, 10 \rangle$ (left) and the $\langle 5, 10 \rangle$ (right) scenarios for selected problems with 40 variables. From top to bottom, I_H^{rpd}, $I_{\epsilon+}$ and I_{IGD}.

on the rich literature on this topic for single-objective EAs [136].

40 000 FEs. When a larger FE budget is considered, the best-performing MOEAs are able to solve most problems when $M = 2$. Nonetheless, the rank sums analysis demonstrates that the performance of the AutoMOEA+ is statistically significantly better than that of the state-of-the-art MOEAs for all metrics. By contrast, when $M = 3$, the only algorithm to approximate the actual fronts reasonably is the AutoMOEA+. However, when the overall benchmark is considered, no statistically significant difference can be seen between the AutoMOEA+ and the best-performing state-of-the-art MOEAs.

Results on the $M = 5$ scenario differ from the remaining scenarios in that the disagreements between performance metrics become significant. This way, the best-performing MOEAs vary depending on the given metric. In particular, since the I_H^{rpd} and the I_{IGD} metrics are the most contradicting ones, the AutoMOEA+ ranks first according to the former as it was tuned for this metric, but ranks fourth according to the latter. Finally, when $M = 10$ the AutoMOEA+ is unable to outperform the best-performing MOEAs for any of the metrics. In fact, the performance of AutoMOEA+ resembles the performance observed for MOEA/D, being competitive on $I_{\epsilon+}$ but much worse on I_{IGD}. As previously discussed, this is explained by the disagreement between metrics, and is an expected result given the design of this AutoMOEA+.

5.2.5 Alternative metrics to guide design

Given the strong disagreement between performance metrics on scenarios with $M = 5$ and, specially, $M = 10$, the good results obtained by the AutoMOEA+ algorithms for all metrics in most scenarios is remarkable. Nonetheless, in this section we conduct further investigation to understand whether using different metrics could affect the automatic design process. Specifically, we use alternative metrics for designing AutoMOEA+ algorithms for the $\langle 5, 40 \rangle$ and $\langle 10, 40 \rangle$ scenarios, as we detail below. The structure of the AutoMOEA+ algorithms designed in this section are given in Table 5.7.

$\langle 5, 40 \rangle$: given the good performance of the AutoMOEA+ algorithms designed for scenarios with $M = 10$, we test the effectiveness of adopting the $I_{\epsilon+}$ as guiding metric to design an AutoMOEA+ for this scenario.

Table 5.7: Parameters selected by irace for the AutoMOEA+ algorithms.

	BuildMatingPool				Replacement			Replacement$_{Ext}$			Numerical				EA	
	Selection	SetPart	Refine	Diversity	SetPart	Refine	Removal	Refine	Diversity	Removal	μ	μ_r	λ	λ_r	CR	F
$\langle 5, 40\rangle$	DT(2)	depth	I_H^1	kNN	strength	I_H^1	seq.	$I_{\epsilon+}$	ref. lines	1-shot	80	—	—	0.73	0.37	0.26
$\langle 10, 40\rangle$	DT(4)	count	$I_{\epsilon+}$	crowd.	rank	$I_{\epsilon+}$	1-shot	$I_{\epsilon+}$	kNN	seq.	90	0.77	—	1.78	0.24	0.67

(All AutoMOEA+ algorithms use DE as underlying EA, and an external archive with $N_{\text{ext}} = 500$.
$\Lambda_{params}(\text{Diversity}_{Ext}) = \langle 0.74, \textit{two-layer}, \textit{balanced}\rangle$ for AutoMOEA+$_{\langle 5, 40\rangle}$.)

The most significant structural changes concern (i) BuildMatingPool, which now uses deterministic tournament and dominance depth for set-partitioning, (ii) Preference$_{Rep}$, which now includes dominance strength for set-partitioning but does not use a diversity metric, and (iii) the use of an external archive. In particular, adding set-partitioning relations to both Preference$_{Mat}$ and Preference$_{Rep}$ reduces the computational overhead posed by the I_H^1 indicator, and allows this AutoMOEA+ to use an external archive that is complementary to the remaining components. More precisely, Preference$_{Ext}$ uses $I_{\epsilon+}$ for refinement, whereas the other preference relations use the I_H^1 indicator.

The rank sum analysis given in Table 5.8 shows that the AutoMOEA+ algorithm designed to optimize the $I_{\epsilon+}$ (Auto-ϵ) is statistically equivalent to the best-performing algorithms according to the I_H^{rpd}, but is the only algorithm to display low rank sums also for the remaining metrics. In fact, the statistically equivalent performance of both AutoMOEA+ algorithms according to the I_H^{rpd} is remarkable. More importantly, results from Auto-ϵ evidence that the effect of the performance metric contradictions over the automatic design methodology can be alleviated when $M = 5$.

$\langle 10, 40\rangle$**:** as previously discussed, the $I_{\epsilon+}$ was used for the design of the AutoMOEA+ algorithms on scenarios with $M = 10$. This way, it seems hardly possible to design an effective AutoMOEA+ algorithm according to all metrics for the $\langle 10, 40\rangle$ scenario using a single metric as guide. Instead, in this section we propose a multi-objective formulation of the tuning problem, where performance metrics represent different, contradicting objectives to be simultaneously optimized. As tuning metric, we consider the I_H of the metric space comprising the three indicators, and we use a two-stage normalization approach to ensure all metrics contribute equally. Concretely, we first use bounds[4] for each of the metrics to avoid strong outliers, and then normalize the metrics to the $[1, 2]$ interval. The I_H metric is computed using point 2.2 as reference.

The structure of the AutoMOEA+ algorithm designed using the proposed multi-objective formulation (hereon called Auto-I_H) is given in Table 5.7. Compared to the AutoMOEA+ algorithm designed to optimize the $I_{\epsilon+}$, we notice significant structural differences, evidenced by the use of (i) refinement metrics for all preference relations, and (ii) a large-size external archive, and; (iii) a Replacement component with one shot removal instead of steady-state. Altogether, one can understand these structural changes as a trade-off between computationally demanding components, with irace favoring the combination of refinement metrics and an external archive over steady-state replacement. The rank sum analysis depicted in Table 5.8 confirms the effectiveness of this design choice, as Auto-$I_H$is able to rank first for all metrics considered, and is considered statistically significantly better than all other MOEAs according to the I_{IGD}.

5.2.6 Concluding remarks

In this section, we have demonstrated that the extended AutoMOEA template we propose in this chapter can be used to automatically design algorithms that demonstrate state-of-the-art performance for con-

[4]For $I_{\epsilon+}$ and I_{IGD}, bound is set to 100. For the I_H^{rpd}, bound is set to 1.0.

Table 5.8: Rank sum difference (in parenthesis) between the given MOEA and the lowest ranked ($FE_{\max} = 40\,000$). MOEAs highlighted in boldface present rank sums statistically significantly lower than the others according to Friedman's nonparametrical test. For brevity, only the seven best-performing MOEAs from each scenario are shown.

				$M = 5$			
I_H^{rpd}	**Auto+** (0)	**SMS** (1)	**Auto-ϵ** (31)	IBEA (58)	MOEA/D (103)	SPEA2 (138)	CMA (140)
$I_{\epsilon+}$	**Auto-ϵ** (0)	**SMS** (39)	IBEA (44)	Auto+ (61)	CMA (129)	MOEA/D (134)	SPEA2 (202)
I_{IGD}	**Auto-ϵ** (0)	IBEA (89)	MOEA/D (106)	SMS (113)	Auto+ (142)	SPEA2 (173)	CMA (194)
				$M = 10$			
I_H^{rpd}	**Auto-I_H** (0)	**IBEA** (48)	SMS (104)	SPEA2 (114)	CMA (143)	Auto+ (143)	NSGA-III (164)
$I_{\epsilon+}$	**Auto-I_H** (0)	**MOEA/D** (40)	IBEA (55)	Auto+ (98)	NSGA-III (149)	SMS (163)	NSGA-II (179)
I_{IGD}	**Auto-I_H** (0)	IBEA (67)	NSGA-III (103)	SPEA2 (115)	NSGA-II (185)	HypE (201)	CMA (237)

tinuous optimization. In particular, we remark that all novel components implemented in this chapter have been selected by the automatic configurator for an AutoMOEA+, except for the online replacement component. Appendix C details an investigation specifically targeting the effectiveness of this component, where we show that its use seldom provides performance gains to the MOEAs considered. Finally, we have also demonstrated the performance improvements of the AutoMOEA+ algorithms designed in this section over the original AutoMOEAs, further corroborating the benefits of the extensions we propose in this chapter.

Although the automatic design methodology adopted initially considered a single guiding performance metric, we have demonstrated that, for nearly all scenarios, the performance of the AutoMOEA+ algorithms is consistently equivalent and often superior to that of the other MOEAs. For the only scenarios where this was not observed, we have demonstrated that it is possible to overcome the challenge posed by the contradicting metrics. When $M = 5$, using the $I_{\epsilon+}$ instead of the I_H^{rpd} was enough to design a state-of-the-art AutoMOEA+ algorithm. By contrast, the strong contradiction between metrics when $M = 10$ demanded a different approach, and we have proposed a multi-objective formulation of the tuning problem that proved effective. In fact, the AutoMOEA+ algorithm devised using this methodology demonstrated state-of-the-art performance for all metrics considered, being the only MOEA to do so.

Finally, we have demonstrated that many of the design choices selected for the AutoMOEA+ algorithms differ considerably from what human designers have so far considered. More importantly, we have seen that components from design paradigms that have so far been regarded entirely different can be coupled to produce a high-performing MOEA design, namely the indicator- and decomposition-based paradigms. Furthermore, we have highlighted the differences in the designs from the AutoMOEAs produced from the original template and the AutoMOEA+ algorithms proposed in this extended version. In particular, we have seen how components differ in function of alternative underlying EAs, sometimes in a complex way. For instance, the need for a stronger convergence pressure from GAs demands a reduced-size population, larger tournaments, and the use of external archives, whereas DE allows a completely opposite design.

5.3 Understanding MOEA effectiveness with ablation analysis

One of the major challenges in MOEA research is to identify effective algorithmic components and design patterns that can help propose a novel generation of algorithms that improves over the previous one. For instance, MOEA research experienced major breakthroughs when dominance sorting was proven effective by several algorithms that adopted it, even if differing as to their concrete implementation. Later, elitism was recognized as a component that could radically improve MOEA performance, and shortly after it became default practice in MOEA design. Since those early years, many algorithmic components have

been proven effective, but their combined used has been rather restricted. As previously discussed, the integration between the different design paradigms is at best incipient.

In this section, we conduct an investigation to understand why the designs selected for the AutoMOEA+ algorithms introduced in the previous section are high-performing. In particular, we want to assess the actual performance contribution of specific design choices, and whether they present interactions that provide insights to be reused by the MOEA community in general. More importantly, since our experimental setup considers a number of scenarios and algorithms, a broad analysis can be performed if carefully designed. Next, we describe the analysis methodology we choose for this investigation, and the setup we employ in this section.

5.3.1 Ablation analysis

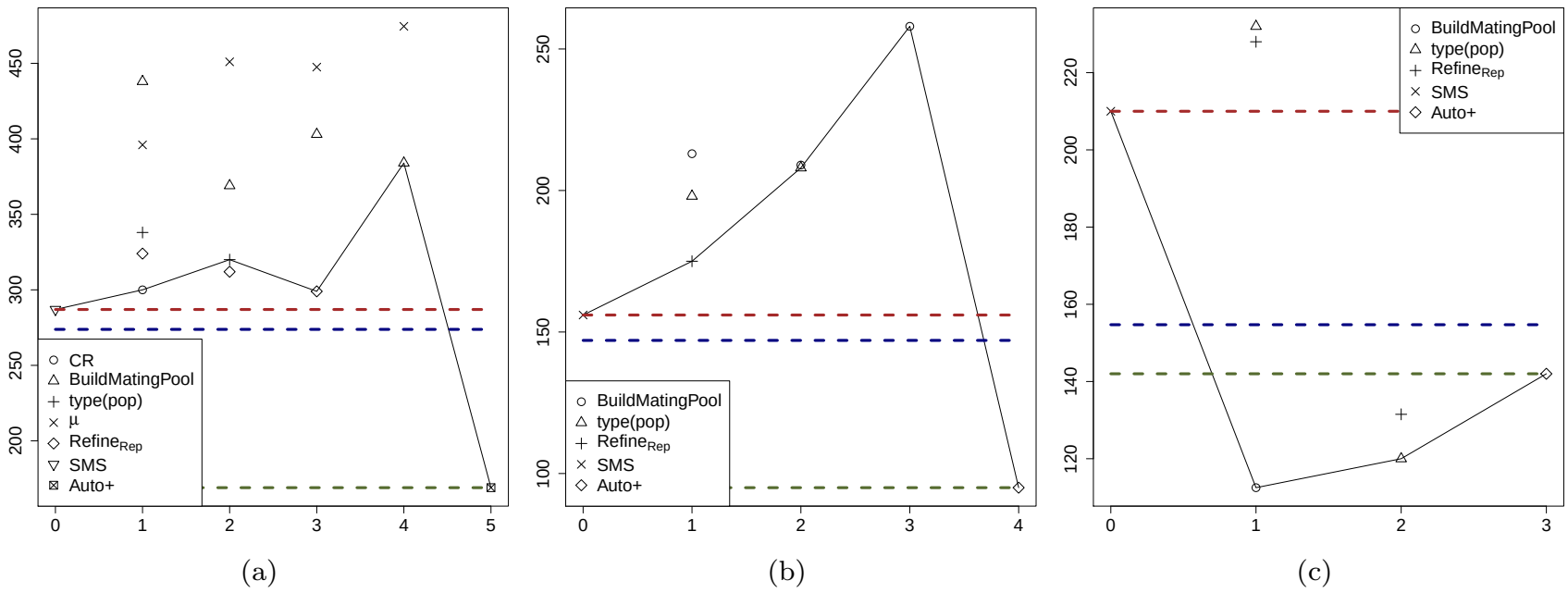
As mentioned in Chapter 3, ablation analysis [77] is an automated methodology that can be used to assess the contribution and interactions of individual algorithmic components, as done in the proof-of-concept investigation on the PFSP described in Appendix A. In a nutshell, ablation is an iterative technique that greatly resembles the path-relinking metaheuristic and that was originally proposed for the context of automatic algorithmic configuration. Given source and target configurations, ablation searches the intermediate configuration space that results from individual parameter changes, replacing parameter values from the source configuration with parameter values from the target configuration. Since the number of combinations can be too large depending on the parameter space considered, ablation restricts the set of intermediate configurations tested as follows. Iteratively, a given ablation step considers all parameters from the current incumbent configuration that differ from the target configuration, and tests the intermediate configurations that are obtained by switching each of these parameter values. In the first step, the incumbent solution is the source algorithm, and the intermediate configurations tested differ from the source by exactly one parameter value. Effectively, the performance of these intermediate configurations indicate the contribution of the individual parameter values. The intermediate configuration that leads to maximal performance change is selected as the new incumbent configuration, and the process continues until the target configuration is reached. In effect, the ablation path formed by the intermediate configurations rank the contributions of the parameter values and help identify possible interactions between parameters.

In the context of algorithm design, source and target algorithms are MOEA designs, and ablation investigates the contribution of design choices rather than parameter values. For this reason, an analysis of algorithms that are structurally different has the potential to provide a number of interesting insights. In addition, it is possible to ablate between algorithms that have been designed for different experimental scenarios and understand how factors interact with the performance of the individual components, or even more complex design choices comprising a sequence of component changes.

5.3.2 Ablation setup

To ablate between two algorithms, some experimental design issues must be properly defined, as we discuss below.

Source/target definition. In the experiments we conduct in this section we test both using AutoMOEA+ as source and target algorithm. More precisely, when an AutoMOEA+ is used as target, the best-performing state-of-the-art MOEA is used as source, and we investigate whether algorithmic components selected for the AutoMOEA+ can help improve the performance of the existing MOEA. Conversely, when an AutoMOEA+ is used as source algorithm, our goal is to identify which components contribute the most to its effectiveness. In this case, we test IBEA and SMS as target algorithms, since they are both high-performing and structurally different, and can hence provide a rich set of insights.



(a) (b) (c)

Figure 5.3: Intermediate configurations tested for $\langle 2, 10\rangle$, considering SMS as source (Step 0) and AutoMOEA+$_{\langle 2, 10\rangle}$ as target (last step) algorithms, with different intermediate design numerical parameter approaches. (a): numerical parameters as part of the ablation; (b) and (c): numerical parameters from source and target designs, respectively. The solid line connects the changes that caused the largest performance improvement. Dashed lines represent the rank sums of source and target algorithms, and rank sum threshold for statistically significant difference w.r.t. to the best ranked design.

Finally, we also conduct investigations where both source and target are AutoMOEA+ algorithms, in an attempt to understand how experimental factors interact with their designs.

Intermediate design numerical parameters. We test three setup alternatives: (i) using numerical parameters as part of the ablation components, or reusing numerical parameters from (ii) source or (iii) target algorithms.

Intermediate design assessment. We adopt the same experimental setup used by in Chapter 4 and also adopted in the previous section. Specifically, we use the same benchmark sets, statistical tests, number of repetitions, and performance metrics. In addition, we perform ablation for all experimental scenarios we adopt in this chapter. For brevity, we only discuss here results for $FE_{\max} = 10\,000$, and we omit results from different performance metrics when they agree. The full set of results is provided as supplementary material [34].

5.3.3 Ablation insights

Next, we describe the most relevant insights observed from the ablation analyses conducted.

Numerical parameter effects. We illustrate the I_H^{rpd} performance of the different intermediate design numerical parameter approaches in Figure 5.3, where SMS and AutoMOEA+$_{\langle 2, 10\rangle}$ are respectively used as source and target designs, and ablation is conducted on the $\langle 2, 10\rangle$ scenario. In more detail, the ablation depicted in Fig. 5.3a considers numerical parameters as part of the ablation space. By contrast, Fig. 5.3b and Fig. 5.3c depict ablations where intermediate designs use the numerical parameters from source and target algorithms, respectively. As one can see, when numerical parameters from the source design are used (Fig. 5.3b) the intermediate designs present higher rank sums than both source and target designs. This is true also in the case where numerical parameters are considered as part of the ablation space (Fig. 5.3a). More generally, we have observed that intermediate designs are better-performing when the numerical parameters from the better-performing algorithm is used (Fig. 5.3c), whether source or target.

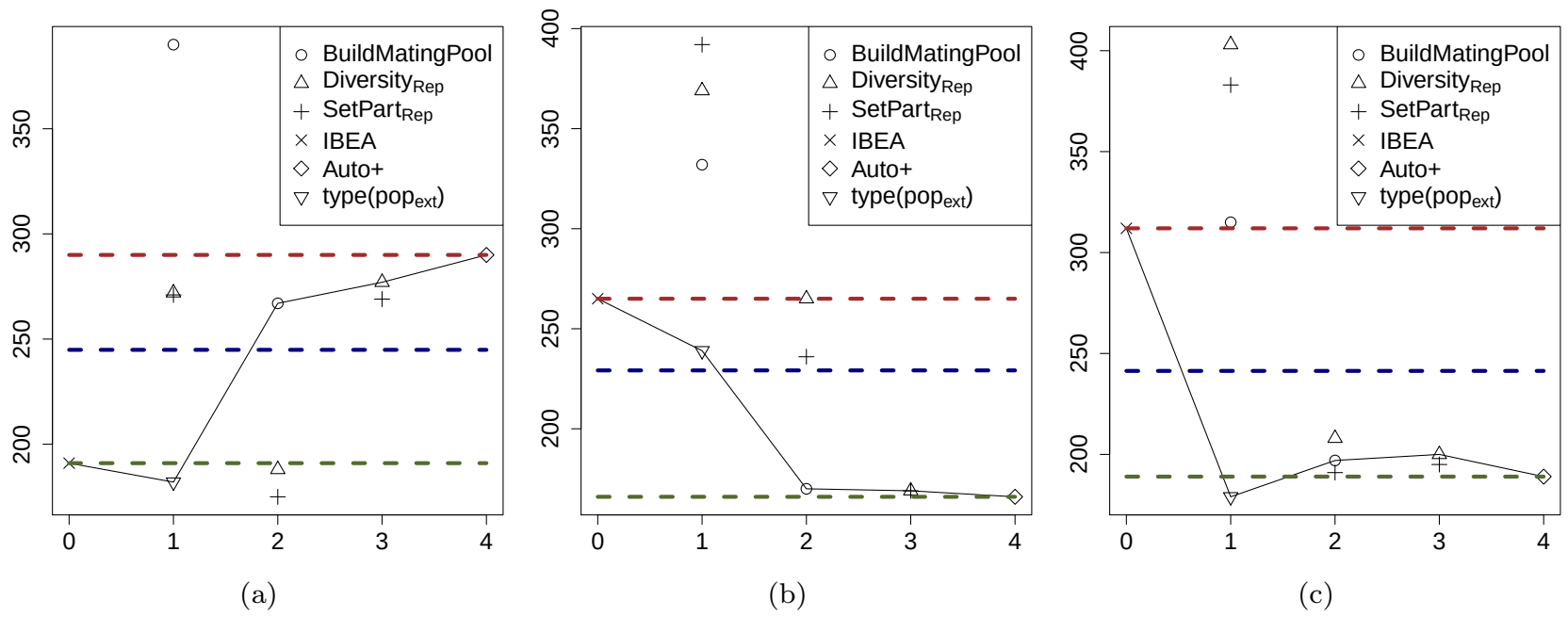


Figure 5.4: Intermediate configurations tested for $\langle 10, 10 \rangle$, considering IBEA as source (Step 0) and AutoMOEA+$_{\langle 10, 10 \rangle}$ as target (Step 4) algorithms, evaluated according to different performance metrics. From left to right, I_H^{rpd}, $I_{\epsilon+}$, and I_{IGD}.

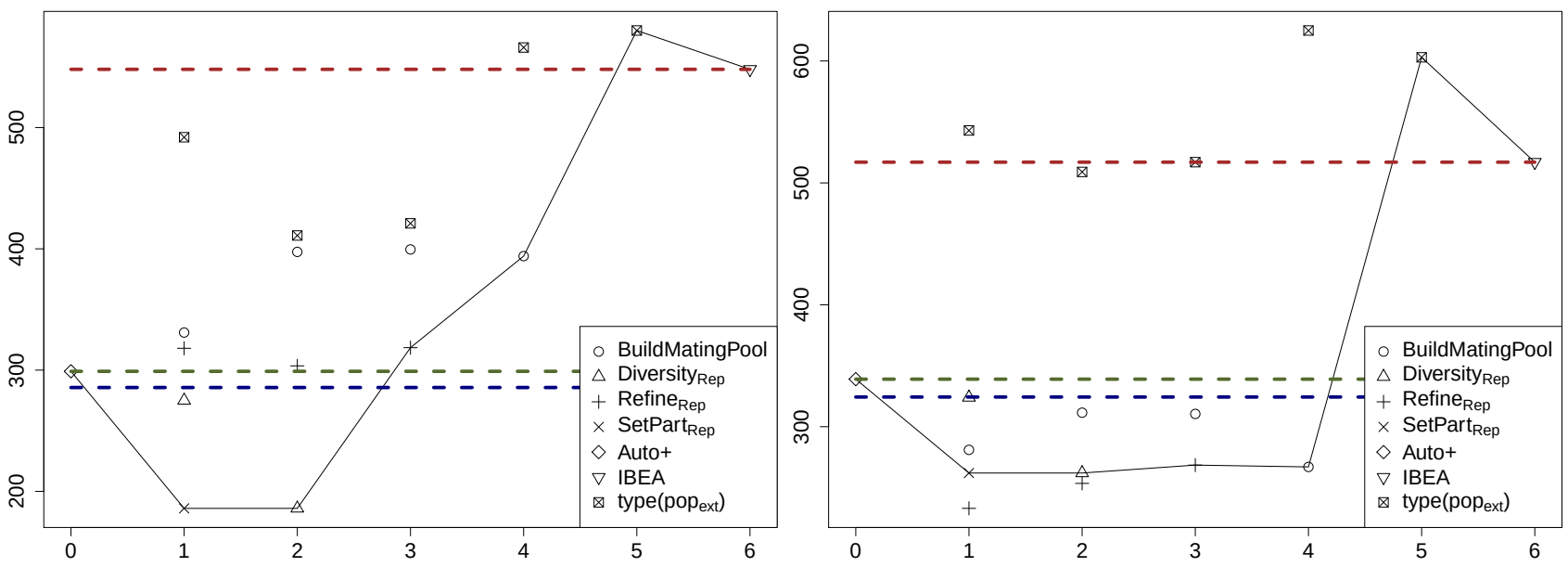


Figure 5.5: Intermediate configurations for $\langle 3, 10 \rangle$, evaluated according to the I_H^{rpd} (left) and I_{IGD} (right). Source: AutoMOEA+$_{\langle 3, 10 \rangle}$. Target: IBEA.

Interactions between components and performance metrics. As previously discussed, the increase in the number of objectives results in a stronger contradiction between performance metrics, with MOEAs being unable to excel for all metrics at once. An important insight we have observed reveals that these interactions can be identified in a design choice level. We illustrate this in Figure 5.4, where IBEA and AutoMOEA+$_{\langle 10, 10 \rangle}$ are respectively used as source and target designs, and intermediate configurations are evaluated by the three different metrics considered in this work. At Step 2, adopting different DE mating schemes can lead to improvements for one metric but worsen for another. In fact, this is a recurring pattern in all ablation analyses where DE schemes are considered as an ablation option. In general, ablations revealed that many components present interactions with performance metrics, explaining the difficulty of having a single MOEA perform well according to multiple metrics when the design process uses a single metric, as traditionally done in the literature.

Existence of design space (false) plateaus. Similarly to other optimization problems, we have observed that the landscape of the MOEA design space presents *plateaus*, i.e., components that could

be interchangeably used with minor changes to performance. However, on scenarios where metrics disagree consistently, we have noticed the existence of *false plateaus*, i.e., component changes that do not affect the performance of the intermediate designs according to a given metric but that improve or worsen performance according to another metric. The first type of plateau is illustrated in Steps 2 and 3 of Figure 5.4, where no significant rank sum changes can be observed for any metric. The second type is illustrated at Step 3 of the ablation analysis depicted in Figure 5.5, where AutoMOEA+$_{\langle 3,\,10\rangle}$ and IBEA are respectively used as source and target designs, and experiments are conducted on the $\langle 3,\,10\rangle$ scenario. Concretely, changing the Refinement component of Preference$_{Rep}$ does not affect the I_{IGD} performance of the intermediate design (right plot), but significantly worsens its I_H^{rpd} performance (left plot). The existence of these false plateaus help explain the effectiveness of the multi-objective formulation adopted for the automatic design of the AutoMOEA+ algorithm on the $\langle 10,\,40\rangle$ scenario.

Contribution of AutoMOEA+ components to effectiveness. Two sets of analyses with AutoMOEA+ algorithms as source designs were conducted to determine the effectiveness of individual components from these algorithms. The first used SMS as target design and considered scenarios with $M \in \{2, 3, 5\}$ and $FE_{\max} = 10\,000$, and is depicted in Figures 5.6a–5.6b ($M = 3$) and 5.6c–5.6d ($M \in \{2, 5\}$). Overall, results demonstrate the importance of the external archive, the DE/target-to-best/1 scheme, and the steady-state replacement. In addition, for $M = 5$ we see a clear symmetry between the plots for the I_H^{rpd} (5.6c) and I_{IGD} (5.6d), reinforcing the interactions between algorithmic components and performance metrics. Similar results are observed for the ablations with IBEA as target. In this case, however, Preference$_{Rep}$ components also proved instrumental when $M \in \{2, 3, 5\}$. By contrast, Figure 5.7 illustrates the ablation between AutoMOEA+$_{\langle 10,\,10\rangle}$ and IBEA on the $\langle 10,\,10\rangle$ scenario. The most important component is the external archive, which leads to improvements for all metrics. Conversely, metrics evaluate the DE/target-to-best/1 scheme in different ways, with the I_H^{rpd} and the $I_{\epsilon+}$ evaluating this component oppositely. In addition, the I_{IGD} performance of intermediate designs does not change significantly with the change in the DE mating scheme.

Improving existing MOEAs with AutoMOEA+ components. In a set of ablations using the AutoMOEA+ algorithms as target designs, we have noticed that it is possible to improve the best-performing existing MOEAs with small modifications to their structure. Specifically, we have identified that both SMS and IBEA can benefit from external archives and the DE/target-to-best/1 scheme proposed in this work. However, we have noticed that these performance improvements are much more significant for the metrics that were not used for tuning. For instance, Step 1 of Figure 5.4c shows the benefits of using an external archive for the I_{IGD} performance of IBEA on the $\langle 10,\,10\rangle$ scenario. Concerning the metric used for tuning ($I_{\epsilon+}$), we see that only a combination of external archive and mating selection can lead to an intermediate design with statistically significant improvements over the original IBEA design.

5.3.4 Effects of different experimental factors

Undoubtedly, the greatest degree of freedom one could consider for ablation is obtained using AutoMOEA+ algorithms as both source and target. In addition, such analyses can also help us understand interaction between experimental factors that comprise scenarios. Below we conduct three sets of investigations of this type, to understand the effects of M, $FE_{\max}$, and the performance metric used for tuning.

M. We conduct two sets of ablations between AutoMOEA+ algorithms designed for scenarios with different M. The first is given in Figure 5.8, where AutoMOEA+$_{\langle 5,\,10\rangle}$ is used as source and AutoMOEA+$_{\langle 2,\,10\rangle}$ as target designs, and only I_H^{rpd} results are shown since metrics agree. Not surprisingly, the component change that would affect the most performance would be to replace the I_H^1 refinement of Preference$_{Rep}$

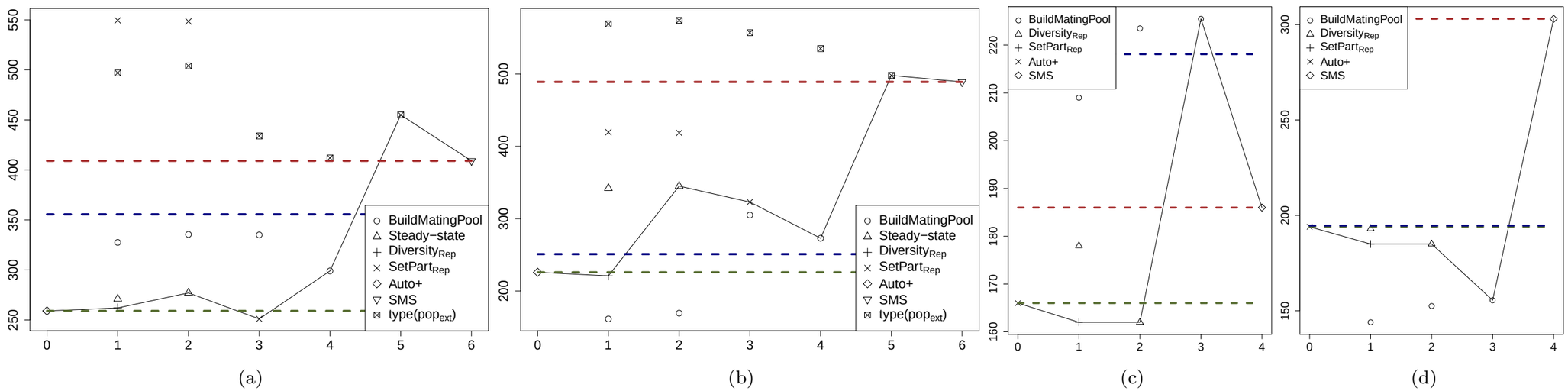


Figure 5.6: Intermediate configurations tested for $\langle 3, 10\rangle$ (a: I_H^{rpd}; b: I_{IGD}) and $\langle 5, 10\rangle$ (c: I_H^{rpd}; d: I_{IGD}). Target: SMS.

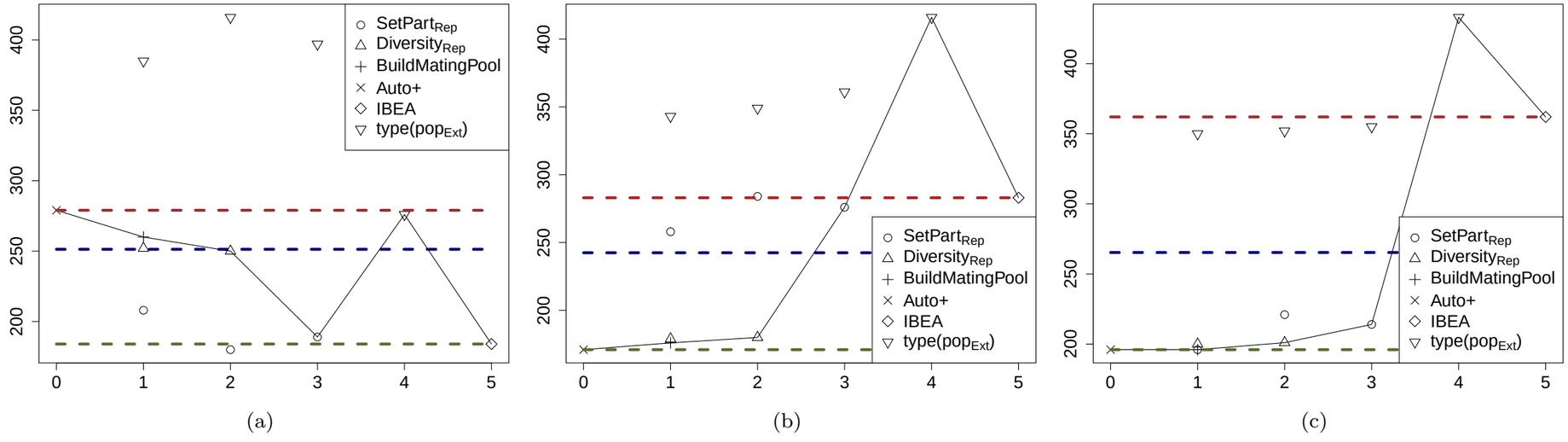


Figure 5.7: Intermediate configurations tested for $\langle 10, 10\rangle$, considering AutoMOEA+$_{\langle 10, 10\rangle}$ as source (Step 0) and IBEA as target (last step) algorithms, evaluated according to different performance metrics. From left to right, I_H^{rpd}, $I_{\epsilon+}$, and I_{IGD}.

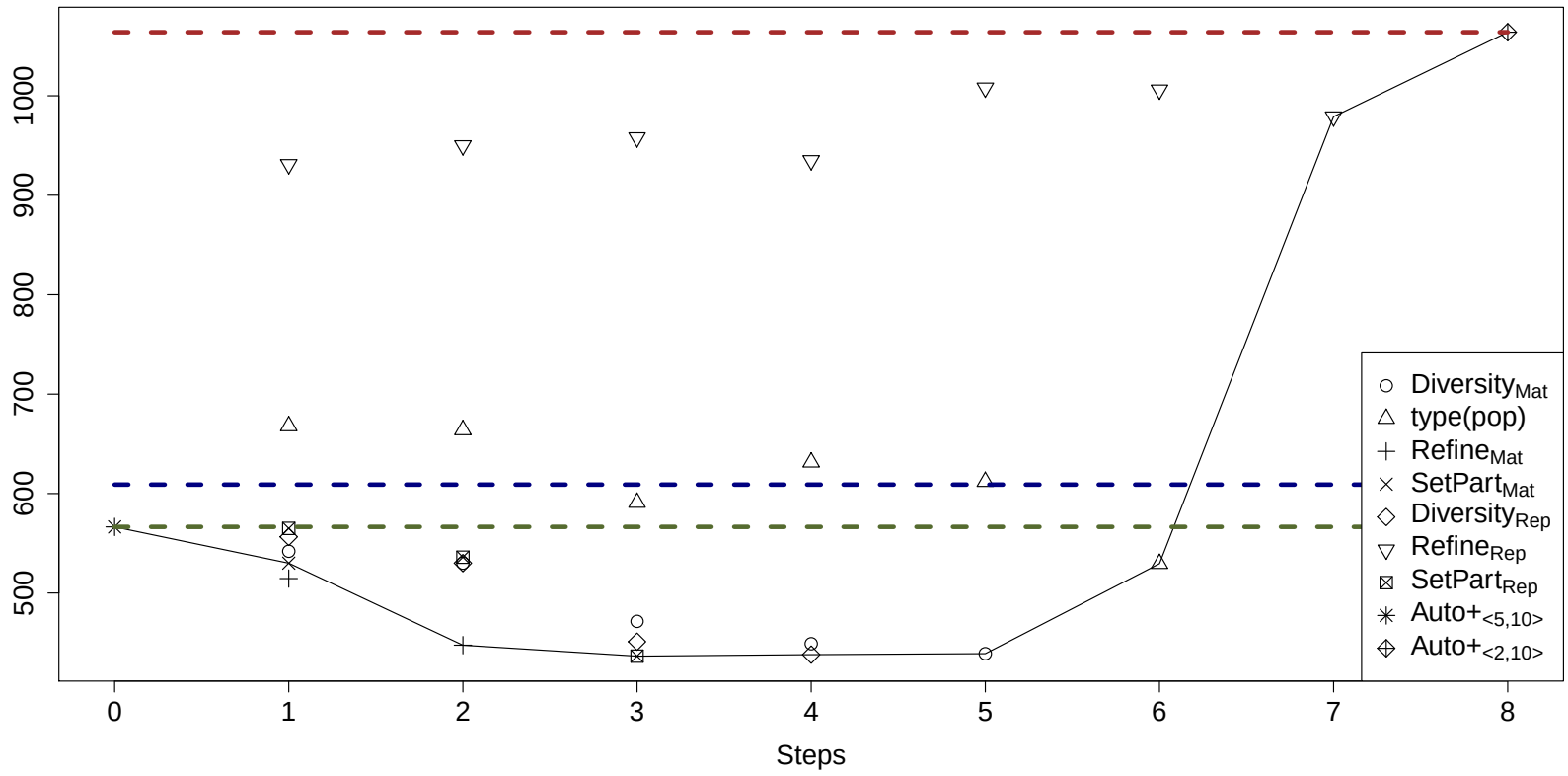


Figure 5.8: Intermediate configurations of the ablation between AutoMOEA+$_{\langle 5,\,10\rangle}$ (source) and AutoMOEA+$_{\langle 2,\,10\rangle}$ (target) according to the I_H^{rpd} on the $\langle 5,\,10\rangle$ scenario.

with the I_H^h, since for more than three objectives the sampled I_H^h values are not accurate enough. A second critical component is the regular population (instead of a bounded internal archive), a fact related to the convergence pressure level required by this scenario. It is, however, interesting to notice that at Step 6 the intermediate design that uses a bounded internal archive can still present state-of-the-art performance. Effectively, this design could likely be used for different M scenarios and still improve over the state-of-the-art in MOEAs for continuous optimization.

The second ablation, given in Figure 5.9, is performed on the $\langle 10,\,10\rangle$ scenario and considers AutoMOEA+$_{\langle 10,\,10\rangle}$ as source and AutoMOEA+$_{\langle 5,\,10\rangle}$ as target designs. The different metrics mostly agree as to the importance of the components, with the computationally expensive components such as I_H^1, external archives, or steady-state replacement being poor choices. More importantly, the intermediate configuration obtained at Step 3 is statistically significantly better than AutoMOEA+$_{\langle 10,\,10\rangle}$ for all but the metric used for tuning. In fact, this is an important example of a false plateau, and reveals that yet better performing MOEAs could likely be found for the $\langle 10,\,10\rangle$ scenario using our proposed multi-objective tuning formulation.

***FE*$_{\max}$.** Figure 5.10 depicts the ablation between AutoMOEA+ algorithms designed for different $FE_{\max}$ scenarios. In particular, AutoMOEA+$_{\langle 5,\,10\rangle}$ is used as source and Auto-ϵ as target designs, and plots are given for the I_H^{rpd} (left) and I_{IGD} (right). We remark that $I_{\epsilon+}$ results are omitted as they are consistent with I_{IGD} results. The first important observation regards the difference in performance from source and target designs depending on the metric considered. More precisely, each AutoMOEA+ algorithm is considered statistically significantly better than the other according to the performance metric used for its design. In addition, we see that AutoMOEA+$_{\langle 5,\,10\rangle}$ could be made better-performing if the external archive from Auto-ϵ were adopted, but this design was not found by irace because it does not lead to a statistically significant improvement according to the I_H^{rpd}. Effectively, this result suggests that the multi-objective formulation could also be helpful in scenarios with $M = 5$.

A second investigation concerning the effect of $FE_{\max}$ is given in Figure 5.11, where AutoMOEA+$_{\langle 10,\,10\rangle}$ is used as source and Auto-$I_H$as target designs, and plots are given for all metrics. Overall, we see that the most relevant components from AutoMOEA+$_{\langle 10,\,10\rangle}$ depend on the performance metric considered.

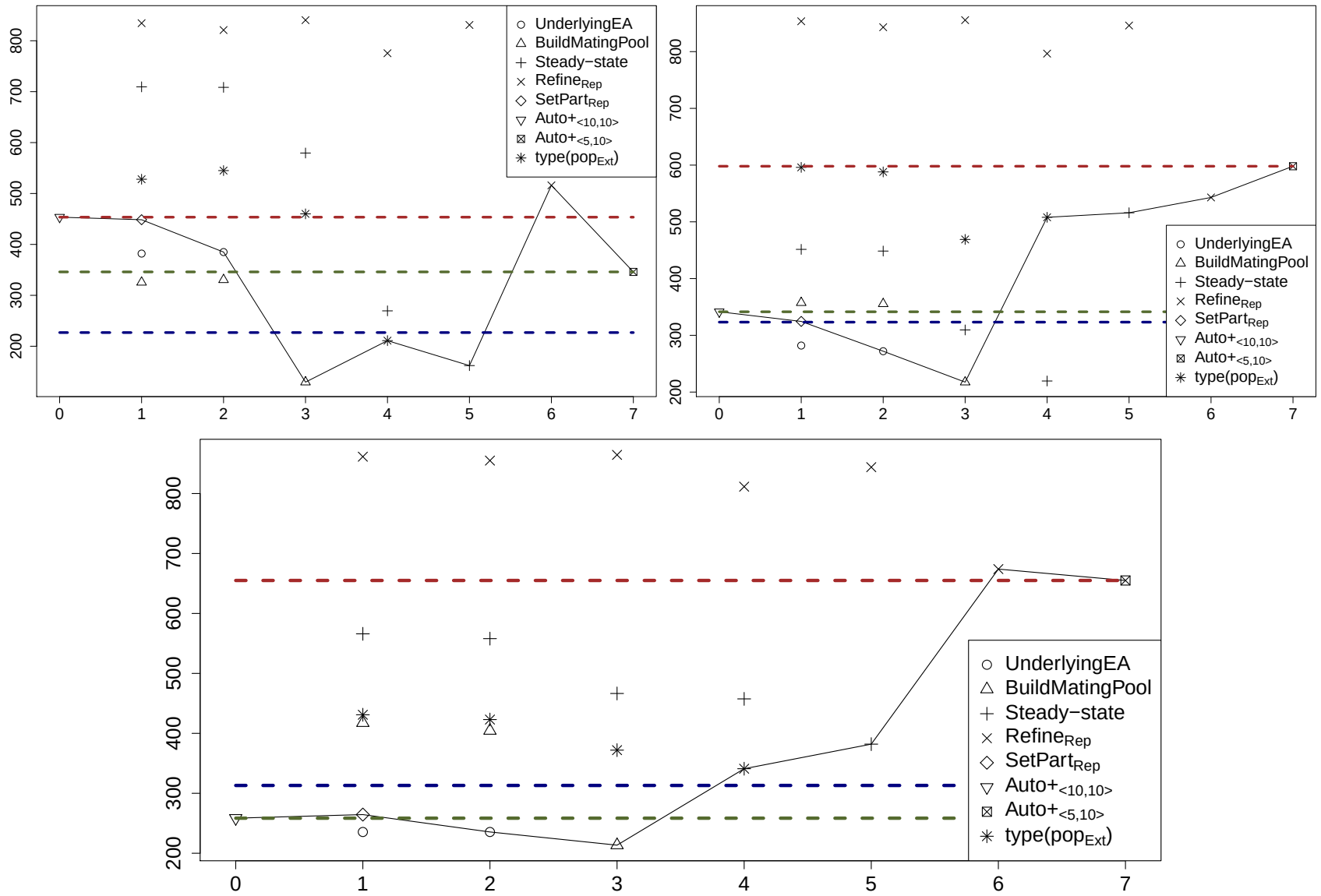


Figure 5.9: Intermediate configurations of the ablation between AutoMOEA+$_{\langle 10,\,10\rangle}$ (source) and AutoMOEA+$_{\langle 5,\,10\rangle}$ (target) according to the I_H^{rpd} (top left), I_{IGD} (top right), and $I_{\epsilon+}$ (bottom) on the $\langle 10,\,10\rangle$ scenario.

Nonetheless, an important observation concerns the performance of the intermediate design depicted at Step 7. This design only differs from Auto-I_H on the numerical parameter settings adopted, indicating the design of this algorithm is robust w.r.t. different stopping criterion. More importantly, this result suggest that the automatic design of anytime MOEAs is a feasible task, as long as adaptive numerical parameters are considered.

Different tuning metrics. Figure 5.12 shows the ablation results on scenario $\langle 5,\,40\rangle$ using Auto-ϵ as source and the AutoMOEA+$_{\langle 5,\,40\rangle}$ designed to optimize the I_H^{rpd} as target. In particular, I_H^{rpd} results (left) differ from the remaining metrics (right) to a remarkable extent. Specifically, all designs are considered statistically equivalent according to the I_H^{rpd}, but very different according to the $I_{\epsilon+}$ and I_{IGD}. We notice that the external archive selected for Auto-ϵ is instrumental for the good performance of this algorithm according to these metrics. Effectively, these observations explain why Auto-ϵ is able to match the performance of AutoMOEA+$_{\langle 5,\,40\rangle}$ on the I_H^{rpd}, and significantly surpass it for the remaining metrics.

Finally, Figure 5.13 shows the ablation results on scenario $\langle 10,\,40\rangle$ using Auto-I_H as source and the AutoMOEA+$_{\langle 10,\,40\rangle}$ designed to optimize the $I_{\epsilon+}$ as target. In particular, we show rank sums computed according to the proposed multi-objective formulation (top left) or to the individual performance metrics (remaining plots). Clearly, the multi-objective formulation is in general able to compensate the contradictions between metrics. In fact, the only situation where we see a big difference between the component selected by the multi-objective formulation and the component that a given metric is

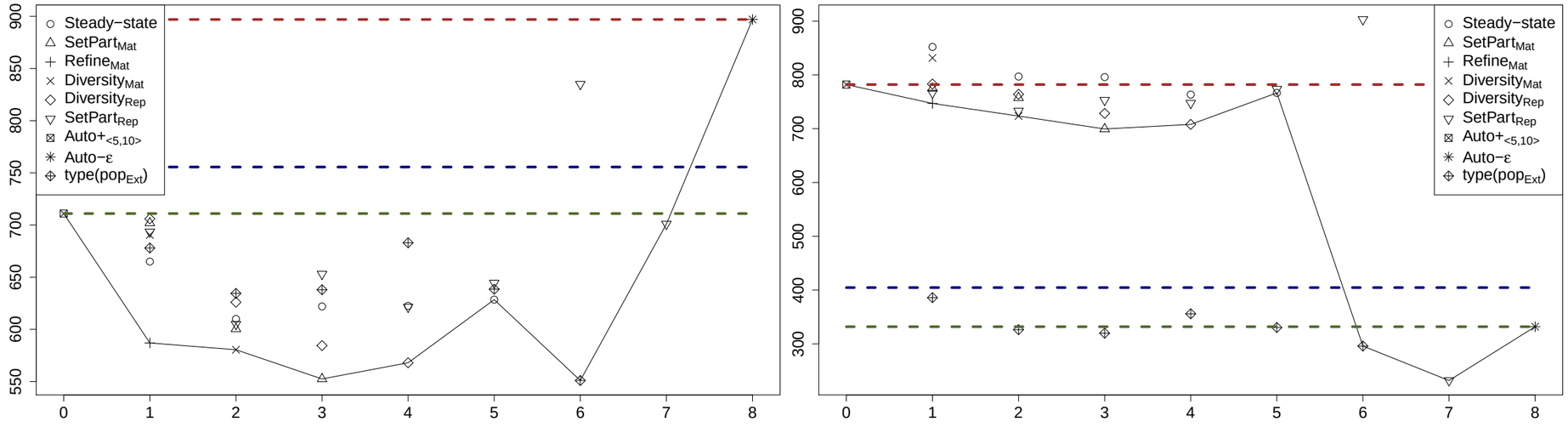


Figure 5.10: Intermediate configurations of the ablation between AutoMOEA+$_{\langle 5,\,10\rangle}$ (source) and Auto-ϵ (target) according to the I_H^{rpd} (left) and I_{IGD} (right) on the $\langle 5,\,10\rangle$ scenario.

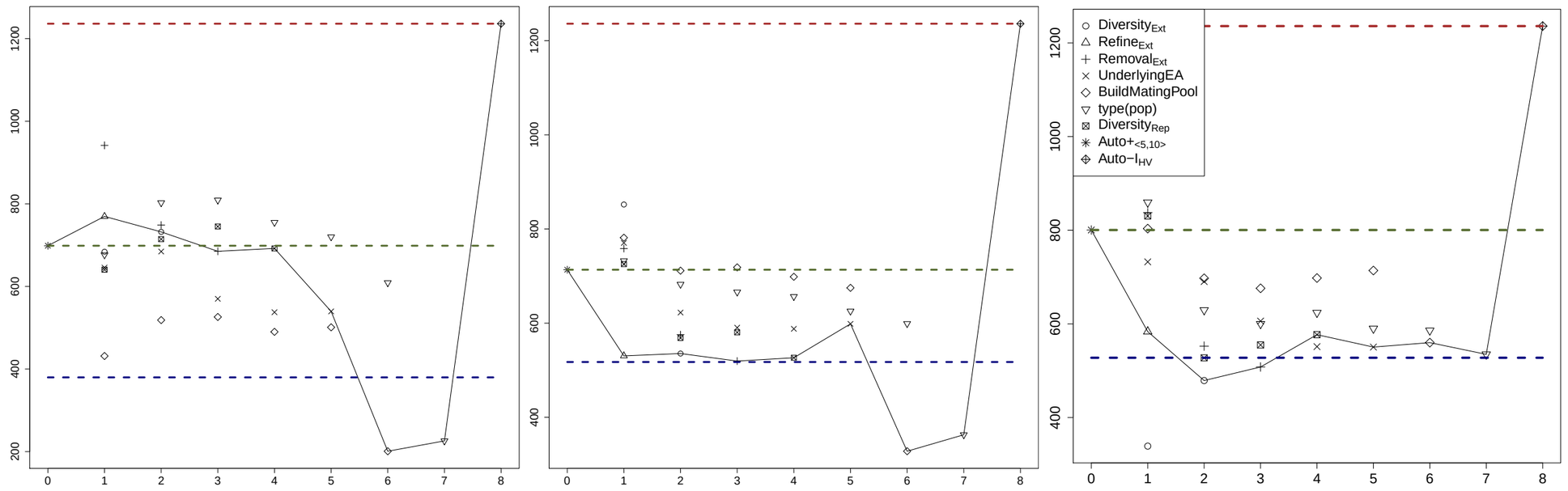


Figure 5.11: Intermediate configurations of the ablation between AutoMOEA+$_{\langle 10,\,10\rangle}$ (source) and Auto-I_H (target) according to I_H^{rpd} (left), $I_{\epsilon+}$ (center), and I_{IGD} (bottom) on the $\langle 10,\,10\rangle$ scenario.

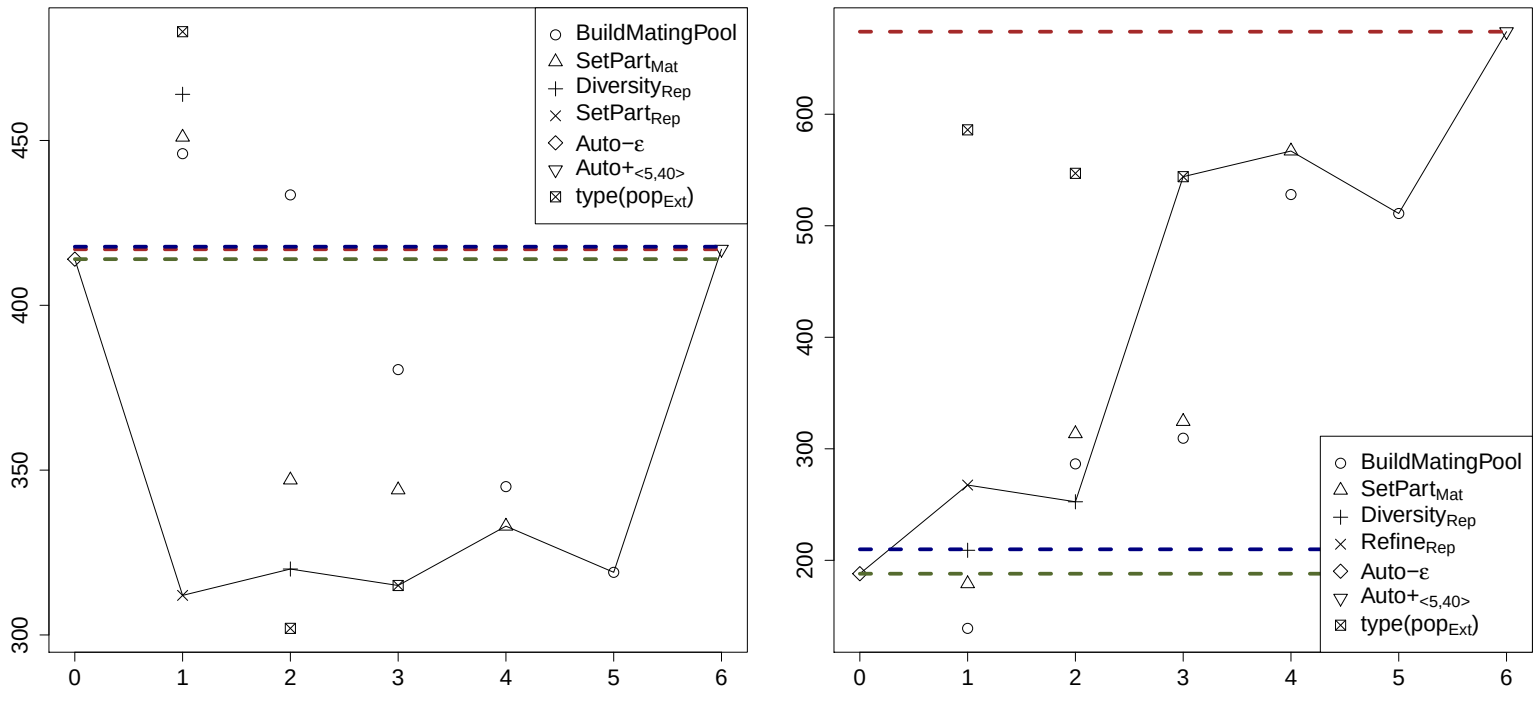


Figure 5.12: Intermediate configurations of the ablation between Auto-ϵ (source) and AutoMOEA+$_{\langle 5,\,40\rangle}$ (target), according to the I_H^{rpd} (left), and I_{IGD} (right) on the $\langle 5,\,40\rangle$ scenario.

observed at Step 6, specifically for the I_{IGD} (bottom right). However, this is a natural consequence of having the hypervolume being computed based on raw metric values, and indicates that the absolute differences in the I_{IGD} values are rather small for the intermediate designs considered at Step 6. Concerning the contribution of individual components, we notice that the Refinement component of Preference$_{Rep}$ is the most critical to performance. Conversely, using a regular population or a regular bounded archive does not really affect the performance of Auto-I_H, and neither does using steady-state selection.

5.3.5 Concluding remarks

In this section, we have conducted an extensive experimental campaign, demonstrating that the analysis of the component-wise performance of MOEAs can be done in an automated way. In particular, we have coupled our proposed framework with an automated analysis methodology originally proposed for the context of automatic algorithm configuration, which we have adapted to the context of component-wise algorithm analysis.

Several different ablation setups were considered. The first setups confirmed the importance of proper parameter configuration, and revealed the existence of (i) interactions between components and metrics and of (ii) plateaus. Specifically, the existence of false plateaus is consistent with the contradicting nature of the performance metrics generally adopted for MOEA assessment, but had never been reported on a component-wise level. The same setups also revealed the effectiveness of the adaptation of DE/target-to-best/1 scheme we propose in this work, and confirms that external archives can be instrumental to the performance of the algorithms. Nonetheless, even these components present interactions with the performance metrics, specially the DE mating selection scheme.

Another set of ablation setups focused on the effects of experimental factors over component-wise performance. Among the most important conclusions, we have observed that a single intermediate design is sometimes able to prove effective for different M or $FE_{\max}$ scenarios, even if adaptive parameter mechanisms are required for that end. In addition, we have clearly seen the strong interaction between metrics and components on many-objective scenarios, but have shown that this effect can be alleviated either by using different tuning metrics or by a multi-objective formulation of the tuning problem.

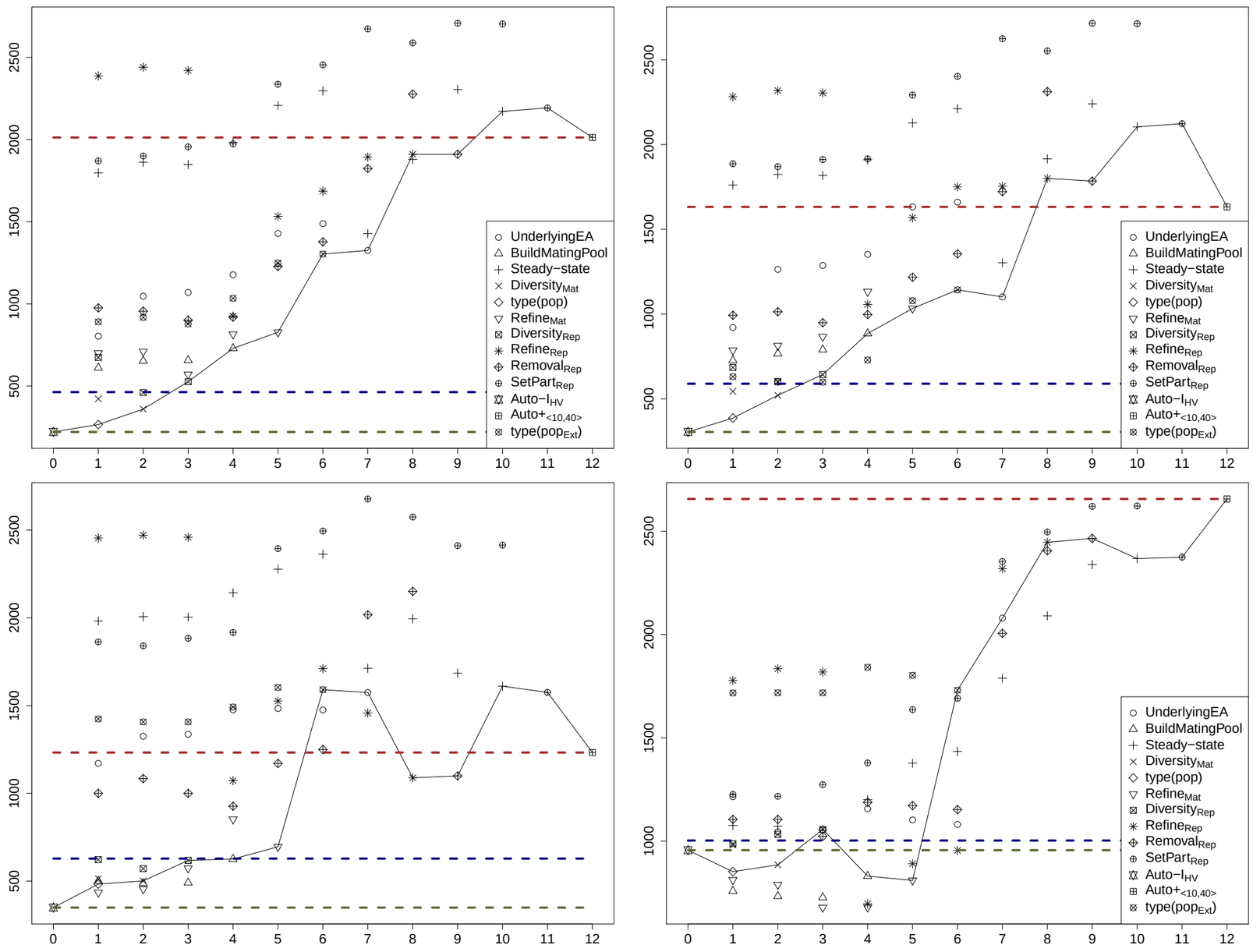


Figure 5.13: Intermediate configurations of the ablation on the $\langle 10, 40 \rangle$ scenario between Auto-I_H (source) and AutoMOEA+$_{\langle 10, 40 \rangle}$ (target), evaluated according to the proposed multi-objective tuning formulation (top left) or the individual performance metrics: I_H^{rpd} (top right), $I_{\epsilon+}$ (bottom left) and I_{IGD} (bottom right).

5.4 Conclusions

Multi-objective evolutionary algorithms (MOEA) have become the method of choice for tackling multi-objective optimization problems (MOPs), and as a result a vast literature with a number of different algorithm proposals can be identified. In general, a set of diverse algorithms is firstly proposed and then commonalities are identified through unified model studies. More importantly, such studies help evaluate the actual effectiveness of individual algorithmic components, which are later used for the proposal of a novel generation of improved MOEAs. More recently, a number of diverse proposals of MOEAs specifically designed for tackling many-objective optimization problems (MaOPs) has arisen due the practical limitations faced by traditional MOEAs on MaOPs. In a recent state-of-the-art evaluation, however, he have identified that more recent algorithms have not been able to properly improve over existing ones, and that on truly many-objective scenarios it was not possible to select a single dominating algorithm.

In this chapter, we have extended in a number of ways a component-wise unified model (a template) we have recently proposed, improving it to comprehend (i) different underlying evolutionary algorithms (EAs), (ii) the decomposition-based MOEA design paradigm, and (iii) archive-specific bounding techniques. More importantly, all the novel components of our template can be freely combined in human-reasonable ways, but its major application concerns the automatic design of MOEAs for MOPs and MaOPs. In fact, we have empirically demonstrated that the AutoMOEA+ algorithms devised in this chapter improve over the previously identified state-of-the-art in MOEAs for continuous optimization. Specifically, we have designed AutoMOEA+ algorithms for each of the 12 experimental scenarios considered, and in all scenarios they have either (i) outperformed existing MOEAs for all metrics, or (ii) matched their performance for some metrics and significantly surpassed it for the remaining ones. In particular, the biggest challenge concerned this contradiction between metrics, which were far stronger for $M = 10$ or $FE_{\max} = 40\,000$ scenarios. However, we have shown that the multi-objective formulation of the tuning problem we propose and adopt is an effective approach to this issue.

In addition to proposing novel algorithms, a second goal of this chapter was to demonstrate that it is possible to automate the component-wise analysis of MOEAs. The primary contribution of such an approach is to explain the actual performance contributions of individual algorithmic components high-performing MOEAs present, i.e., to deconstruct their designs. Overall, we have shown that the original components of the framework were instrumental to its performance, specially the DE/target-to-best/1 scheme proposed in this chapter and the possibility of using different preference relations for all main components a MOEA presents. A secondary, but also very relevant contribution was to demonstrate interactions between components and (i) components, (ii) performance metrics, (iii) experimental factors, and (iv) numerical parameters. In particular, we have shown the existence of design space plateaus, some of which can trick configurators and designers into wrongly believing that a given component is not relevant if conclusions are based on a single performance metric. More importantly, this is compelling evidence that explains why the improvements from more recent MOEAs specifically designed for many-objective optimization have been either small or questionable. In addition, it shows that manually designing an effective MOEA according to a set of diverse performance metrics is a challenging task, but that the automatic design methodology coupled with the multi-objective tuning formulation we employ here is able meet this challenge.

CHAPTER 6

Conclusions

Multi-objective optimization is a well-established research field that models real-world problems with high accuracy at the cost of increased computational complexity. Among the main approaches to solving a multi-objective problem (MOP) is the use of metaheuristics, and in particular the most mature research work from this area concerns multi-objective evolutionary algorithms (MOEAs). The research on MOEAs has provided a number of significant advances to the MO research itself as reviewed in this thesis, ranging from performance assessment theory and practice, to devising a number of effective and relevant algorithmic components reused by many other multi-objective metaheuristic approaches. In this thesis, we have pushed the research on MOEAs and on multi-objective optimization in general even further. In particular, the existing MOEA literature consists of many different proposals that have been considered from a high-level perspective. More precisely, few are the works in the literature that try to analyze MOEAs from a unified model perspective, identifying elements in these proposals that are either similar or equivalent, and taking advantage of such insights. Furthermore, proposals towards a high-level unified model are also instrumental to several advancements, such as a better understanding of similarities and differences between algorithms, or if an effective algorithmic component from one MOEA could benefit a different MOEA.

In fact, given the level of the MOEA research, it is rather strange that recent efforts in this direction cannot be identified in the literature. We conjecture two main reasons for this. Firstly, there have rarely been a number of proposals of frameworks to make MOEAs easier for practitioners to apply when dealing with a novel scenario. However, the main emphasis of these works has been on problem reusability, with very little advancements towards composability. Although regretable, this fact is understandable since a flexible MOEA framework implementation is both (i) theoretically difficult, since the literature presents many structurally different proposals, (ii) practically challenging, as implementing systems with a high flexibility level while maintaining efficiency is difficult to accomplish, and; (iii) potentially undesirable, given that many algorithmic component combinations would not make sense from a human perspective. The second fact that helps explain the lack of a unified model is the constant evolution of MOEA design, which is never properly matched by rigorous performance assessments that could help narrow down the algorithmic components and MOEA designs that are indeed potentially effective. In fact, the advent of many-objective optimization has made this task even more challenging, as many novel algorithmic concepts have been proposed over a short span of time in an effort to improve MOEA effectiveness on

this type of problems.

The work described in this thesis overcomes these challenges by (i) proposing a novel conceptual view of MOEAs that is at the same time flexible, efficient, and representative; (ii) empirically demonstrating that this approach can improve over current MOEA engineering, and; (iii) producing a set of relevant insights that further increase the effectiveness of MOEAs on its main application domains, specially on continuous optimization. In particular, in order to accomplish the goals of this thesis we have relied on existing research on automatic algorithm design, to which we have also significantly contributed by refining the existing approaches in this field. More precisely, this thesis is among the first works identified in the literature to apply either automatic algorithm configuration or design to multi-objective optimization scenarios. During the course of this work, we have identified several unresolved practical issues arising from MO complexities that other researchers had never reported. In fact, our work is instrumental to further encouraging the adoption of automatic engineering in multi-objective optimization, and equally important to machine learning MOEA research. Concretely, in this thesis we have both (i) produced and made available a large amount of experimental data, and (ii) used this data in an effective way to produce relevant contributions, such as the automatic design of state-of-the art MOEAs for continuous optimization.

In the remainder of this chapter, we initially individually address the main contributions of this work. Later, we discuss the potential future works that originate from the work described in this thesis.

6.1 Contribution summary

The main contributions of this thesis are listed below:

1. **A component-wise MOEA algorithmic framework based on an unified template:** We have proposed a composable MOEA framework based on a representative, flexible, and practical template. First, the proposed framework is the most representative currently available as it encompasses (i) different MOEA design paradigms, namely dominance-, indicator-, and decomposition-based; (ii) different underlying evolutionary algorithms (EAs), with the most relevant for continuous optimization being already integrated, and; (iii) the possibility of using archives either as part of the evolutionary process or externally, a modeling that allows us to instantiate MOEAs as different as PAES, SPEA2, and SMS. In fact, our framework allows from a single template the instantiation of more MOEAs than any other existing framework or unified model. Currently, over 15 MOEAs can be instantiated, and we believe that an even greater amount will be instantiable as more components are implemented.

   Concerning flexibility, the proposed framework is built upon the notion of composability. Specifically, our approach comprises a repository of algorithmic components (building blocks) and the template that defines how they can be combined in a flexible, yet humanly-reasonable way to produce algorithmic designs (MOEAs). In particular, the flexibility of our approach is best evidenced by the preference relation definition we adopt, as well as by the possibility of using different preference relations on the different components of the algorithms, as in more recent MOEAs. As this thesis has shown, this possibility can be instrumental for the performance of MOEAs in given application scenarios.

   Finally, the proposed framework is practical since instantiating MOEAs becomes as trivial as setting a few, command-line, categorical parameters. In addition, MOEA components are implemented with a clear separation from problem-specific components, and so reusability is maximized from a practitioner's perspective. More importantly, coupled with automatic algorithm engineering approaches, the proposed framework can be used on novel and existing application domains to

(i) conduct experimental analyses, and (i) generate high-performing algorithms or portfolios with little human effort.

2. **The empirical demonstration that automatically designed component-wise MOEAs can outperform manually designed MOEAs:** To demonstrate the potential of our proposed framework for template-based automatic algorithm design, we have conducted an initial feasibility investigation where we automatically designed AutoMOEAs for several relevant application problems. In particular, we have considered (i) unconstrained continuous optimization, the primary application domain for MOEAs, and (ii) several multi-objective variants of the permutation flowshop problem (PFSP), one of the most relevant scheduling problems found in the combinatorial optimization literature. Besides differing as to structural characteristics, the MOPs selected also range as to the number of objectives (2–5) and problem variables they present, making this assessment yet more representative. Results have shown that the performance of the AutoMOEAs was both better and more robust than the MOEAs from which their components have been gathered. More precisely, AutoMOEAs were considered statistically significantly better for all application problems considered, with the exception of a PFSP variant known to present very particular characteristics and for which further investigation is required.

   Besides generating novel, high-performing AutoMOEAs for the traditional setups commonly adopted, we have also assessed the robustness of the template-based automatic design approach by considering other experimental setups that are also relevant. For continuous optimization, besides the traditional setup where a maximum number of function evaluations (FEs) is given to algorithms, we have also considered a runtime-constrained setup to evaluate whether the computational overhead posed by components would affect the design of the AutoMOEAs and the performance of MOEAs in general. Furthermore, we have also considered the possibility of using different benchmark sets for tuning and testing, to assess the generality of the AutoMOEAs. For both setups, results demonstrated that AutoMOEAs were able to consistently outperform existing MOEAs, and an analysis of their structure produce relevant insights, such as the high cost-benefit ratio of the $I_{\epsilon+}$ to guide the search, as this component presents little computational overhead but provides valuable information to MOEAs. Regarding the PFSP, we have assessed the possibility of designing a single AutoMOEA for optimizing multiple variants and have identified a high-performing, competitive MOEA design that outperforms existing MOEAs.

3. **A comprehensive performance analysis of multi- and many-objective EAs, a concrete first step towards defining a state-of-the-art in MOEAs for continuous optimization:** In order to demonstrate that AutoMOEAs can improve over the state-of-the-art in MOEAs for continuous optimization, we have conducted a large-scale, comprehensive performance assessment encompassing several groups of MOEAs, ranging from historically relevant to specifically designed for MaOPs. More importantly, we have considered the broadest experimental setup for an assessment of MOEAs to date. Concretely, we have considered over 15 MOEAs independently proposed from the different MOEA design paradigms, always selecting high-performing algorithms from each paradigm. In addition, in this work we have originally proposed a conceptual separation between the multi-objective components used by MOEAs and the underlying EAs with which they are coupled, effectively expanding the representativeness of the algorithms considered. Finally, we have considered 12 experimental scenarios comprising different number of objectives (2, 3, 5, and 10) and stopping criteria (2 500, 10 000, and 40 000 FEs). In particular, the different stopping criteria we have adopted allowed us to analyze how MOEAs rank in different real-world situations, simulating problems with computationally expensive FE computation, as well as problems where the computational overhead posed by MOEA components matter. Even more important, we have used automated algorithm configuration tools to jointly select the underlying EA and numerical

parameters for the MOEAs considered, as well as a diverse set of performance metrics.

Given the broad scope of our work, a number of relevant insights were produced. Concerning experimental factors, we have (i) assessed the importance of tuning, specially when different stopping criteria are used; (ii) analyzed how different problem characteristics respond at different rates to the effects of dominance resistance, and (iii) empirically verified the correlations between performance metric contradictions and the number of objectives considered. Regarding MOEA performance, we have observed a pattern in results for scenarios with less than ten objectives, where SMS and IBEA have consistently outperformed the other algorithms for all metrics. By contrast, the remaining MOEAs presented great variability w.r.t. performance metrics, problem features, and number of objectives. More importantly, for these scenarios we have seen that recently proposed MOEAs such as HypE and NSGA-III displayed little improvement (if any) over their immediate predecessors. In contrast to the pattern seen for scenarios with up to five objectives, ten-objective results showed that (i) no single MOEA could be simultaneously considered best for all metrics; (ii) the computational overhead posed by SMS components greatly reduced its performance, whereas IBEA remained effective and robust, and; (iii) NSGA-III is a very competitive algorithm, but only according to the metric it was designed for. Nonetheless, the overall performance of MOEAs on this scenario was rather poor for concave problems, a fact that reinforced the need for alternative approaches.

4. **The automatic design of state-of-the-art multi- and many-objective EAs for continuous optimization:** We have demonstrated that the proposed framework can be used for the automatic design of state-of-the-art MOEAs. Specifically, our investigation targeted the primary application domain of MOEAs, namely unconstrained continuous optimization. The AutoMOEA+ algorithms produced in this work proved robust and effective, outperforming MOEAs in many scenarios across all metrics. More importantly, we have analyzed how the AutoMOEA+ designs differed from the original AutoMOEAs, effectively demonstrating how the structure of high-performing application-specific algorithms can only be achieved if enough flexibility is provided. In particular, we have observed that all but the ten-objective AutoMOEA+ algorithms were DE-based, and that several algorithmic components were changed to suit this different underlying EA.

   Concerning the performance comparison to the state-of-the-art, the few scenarios where AutoMOEA+ algorithms were not the top-performing algorithms for all metrics comprised large FE budgets and number of objectives. For these scenarios we have refined the automatic design methodology, effectively investigating the role of the performance metrics used for tuning. In the scenario with five objectives, using the $I_{\epsilon+}$ instead of the I_H^{rpd} metric led to a novel AutoMOEA+ that proved competitive against all MOEAs for all metrics, and outperformed them according to specific metrics. The same approach could not be adopted for the scenario with ten objectives, since the original AutoMOEA+ had already been designed to optimize the $I_{\epsilon+}$ indicator. We have then proposed a novel tuning metric, namely the hypervolume of the performance metrics considered. Effectively, this metric allows the configurator to search the design space in a multi-objective way, without favoring specific performance metrics. Results showed that the novel AutoMOEA+ algorithm devised for this scenario outperformed all other algorithms for all metrics considered. More importantly, these results reinforce the view that the performance assessment of MOEAs is also a multi-objective problem, and that expecting a single performance metric to account for all the desirable characteristics an approximation front should produce is possibly unfeasible.

5. **Experimental analyses concerning the effectiveness and interactions of individual MOEA components on a set of relevant benchmark problems:** We have used the proposed framework to demonstrate that automated algorithm analysis is possible when a component-wise template is adopted. More precisely, we have used ablation, an automated analysis technique originally proposed for the context of automatic algorithm configuration to assess the effectiveness of individual algorithmic components of different MOEAs. Concretely, we have deconstructed MOEAs and AutoMOEAs into their building blocks, and tested algorithmic designs obtained from altering individual components. Besides the natural advantage of being automated, such an analysis can also be relevant for the investigation of component interactions and how experimental factors affect the design of MOEAs. The first investigation of this kind was conducted for the PFSP and is provided as an appendix. In particular, we have performed an ablation analysis for each problem variant between two high-performing MOEAs, namely IBEA and NSGA-II. In addition, we have considered different template abstractions to investigate whether the traditional approaches of modifying component bundles led to different conclusions compared to our approach of dealing with individual components. We have also investigated how to solve the question of what numerical parameters to use for the intermediate designs produced during ablation. Results have shown that the individual component approach we adopt in this work leads to more insights and better-performing intermediate designs, and that these designs tend to be more effective when the numerical parameters from the best-performing algorithm are reused.

   The other two investigations using ablation we have conducted concern multi-objective continuous optimization. The first, provided as an appendix, assesses the effectiveness of different DE-based MOEAs. Specifically, our investigation demonstrated that one of the default components used by many high-performing DE algorithms did not contribute to their effectiveness, and in some cases rendered their performance worse. The second investigation considered the designs of the AutoMOEA+ algorithms, to assess (i) whether their components could help improve the performance of existing, high-performing MOEAs, (ii) the actual contribution and the possible interactions between their components, and; (iii) the effects of different experimental factors on the performance of components, specifically the number of objectives, the function evaluation (FE) budget MOEAs were given, and the tuning metric used for the automatic design process. Results have shown that the application-specific components were instrumental for the performance of the AutoMOEA+, but that a single component could contribute positively according to a metric but negatively according to another. Regarding component interactions, ablation identified how the use of DE changed the need for external archives, given the possibility of using larger population sizes and selection operators that provided less convergence pressure than when GAs were adopted. Finally, we have empirically assessed the challenge of designing a MOEA that is robust for metrics as contradicting as the ones typically used in multi- (and specially many-) objective optimization, a fact that makes the performance of the AutoMOEA+ algorithm yet more remarkable.

6.2 Future work

The research described in this thesis opens a broad horizon for future work. Below, we discuss the ones we find to be most promising ones.

1. **Establishing an open source community:** One appealing feature of open source software frameworks is that many people can simultaneously collaborate to improve these frameworks. In academia, such communities are also important to increase the flow of research works on a given topic, maximizing its impact. For this reason, besides making the tools proposed in this work available for the MOEA research community, we intend to create an open source community of AutoMOEA users and collaborators that can help extend the number of available algorithmic

components and problem-specific representations and operators. Being a practical methodology, we also intend to narrow the gap to the real-world industrial applications that could directly benefit from this research.

2. **Feature-specific benchmark sets for performance assessment and machine learning:** Although many relevant insights were produced in this thesis, an in-depth investigation of the correlations between problem features and individual MOEA components could not be carried out due to the characteristics of the existing benchmark sets of multi-objective continuous problems. In particular, these benchmark sets are limited in the number of functions they present, and even more limited if one wants to have statistical soundness for a feature-wise analysis. A very pressing future work topic for this thesis is to propose large feature-specific benchmark sets, with enough problems that share a given feature, but vary as to the others can be identified. In particular, such a benchmark set would be instrumental not only for performance assessments, but for the application of many machine learning approaches such as algorithm selection to the research on MOEAs.

3. **Automatic MOEA selection to maximize robustness:** The performance of the AutoMOEA and AutoMOEA+ algorithms on the problems investigated in this thesis is undoubtedly remarkable, but specific problem features have demonstrated that having a single MOEA optimize a set of structurally different problems is not yet the best possible approach. By contrast, the research on algorithm selection has shown that combining algorithm selection techniques with algorithm portfolios is an effective approach to maximize robustness. Using the framework proposed in this thesis and feature-specific benchmarks, one could automatically design MOEAs particularly effective for each feature, and comprise a MOEA portfolio for algorithm selection.

4. **Automatic MOEA design for anytime optimization:** As demonstrated in the several experimental analyses conducted in this thesis, MOEAs present little robustness to different stopping criteria. In other words, one may say that MOEAs as traditionally designed present poor *anytime performance* [196]. So far, the most common approaches to improving the anytime optimization of multi-objective algorithms concerns online parameter adaptation. Another existing approach is to model anytime optimization as a bi-objective tuning problem, where one of the objectives refers to effectiveness (solution quality in single-objective optimization or quality indicators in multi-objective optimization) and the other refers to efficiency (runtime or function evaluations). Results from this approach are promising, and indicate that automatic MOEA design could produce even better results. Specifically, while the parameter space of a traditional configuration problem is restricted, the flexibility introduced by automatic design offers an increased potential for locating anytime configurations (designs). Even more promising is an alternative research approach that considers sequential algorithm portfolios. Coupled with the automatic design approach adopted in this work, one could automatically devise a set of MOEAs to maximize time/FE windows, and their sequential use would ensure anytime performance.

5. **Designing state-of-the-art MOEAs for combinatorial optimization:** The initial investigation on the feasibility of the automatic MOEA design for combinatorial optimization demonstrated that AutoMOEAs can consistently outperform existing MOEAs on the PFSP. However, MOEAs are not among the state-of-the-art for this problem, nor for the most relevant combinatorial optimization problems from the multi-objective literature. Nonetheless, the performance of memetic algorithms on single-objective combinatorial optimization suggests that MOEAs could become state-of-the-art for these problems if properly designed. This belief is made stronger when one analyzes the performance of population-based, multi-objective local search algorithms such as Pareto local search. In more detail, PLS is a frequent component of the algorithms that comprise the state-of-the-art for the most relevant combinatorial multi-objective optimization problems, such as the

traveling salesman problem [162], the permutation flowshop problem [70], and the multi-objective knapsack problem [164].

6.3 Concluding statement

The research on multi-objective evolutionary algorithms is central in the optimization literature, and has produced a number of important contributions to the advancements of multi-objective metaheuristics. Nonetheless, the algorithm engineering methodology adopted has gradually slowed down the pace with which novel concrete contributions have been produced. The fact that best illustrates this issue concerns the small, or even questionable, performance improvements from more recent algorithms over well-established ones. In this thesis, we have taken concrete steps towards a novel MOEA algorithm engineering approach. Specifically, we have proposed the currently most flexible and representative unified model for MOEAs, and used it as the basis to implement a MOEA algorithmic framework from which effective algorithms can be automatically designed and analyzed.

Our hope is that the arguments and results provided in this thesis stir a collective effort from the MOEA community towards approaches that can make the most of the potential offered by the remarkable proposals from this field. In addition, many of the future work possibilities discussed in this chapter will require significant collaborations from research groups with different backgrounds and expertise. In the future, we expect to see MOEA designers concerned solely with the proposal of effective algorithmic components, knowing that the practical, experimental aspects of their research has been made easier as a result of our work.

APPENDIX A

A proof-of-concept ablation analysis on the MO-PFSP

In this appendix, we conduct a proof-of-concept investigation to demonstrate the benefits of integrating a component-wise view of MOEAs and iterative analysis tools such as ablation. To do so, we consider two high-performing algorithms that share a similar structure but use rather different individual algorithmic components, namely NSGA-II and IBEA. More precisely, both MOEAs are (i) based on GAs, (ii) use the same preference relation for mating and environmental selection, and (iii) employ elitism. Conversely, we remark two important differences in the components used by these algorithms, depicted in Table 3.4. First, the preference relation used by NSGA-II is based on dominance depth and crowding distance, whereas IBEA uses solely binary indicators to compute scores for each individual of the population. Second, the elitism in NSGA-II follows a one-shot removal policy, whereas IBEA uses a sequential approach. We next present the ablation analysis methodology and describe how we apply it in this appendix.

A.1 Ablation analysis description and setup

We conduct an iterative analysis to understand which algorithmic components cause the main performance differences between the selected MOEAs. More precisely, given a source and a target algorithms, this analysis can be seen as a path relinking in the design space and it has been applied in the context of automatic algorithm configuration before by Fawcett and Hoos [76]. The main motivation for this analysis is to get insight into the contribution of specific components on algorithm performance. We do so by generating intermediate algorithmic designs between the two algorithms. At each step, we modify all individual algorithmic components in which the two algorithms differ, and follow the path that has the maximum impact on performance. In this way, the analysis of the intermediate designs allows us to understand the actual contribution of the individual components to the performance of the algorithm. A number of important setup issues require attention when conducting an ablation analysis, which we detail below.

Defining source and target algorithms is a task generally done in function of the performance of the algorithms and what one expects to investigate during ablation. For instance, using the algorithm with worse performance as source (*worst-to-best* approach) provides insights into which components from the best performing algorithm could help improve the performance of poor performing designs

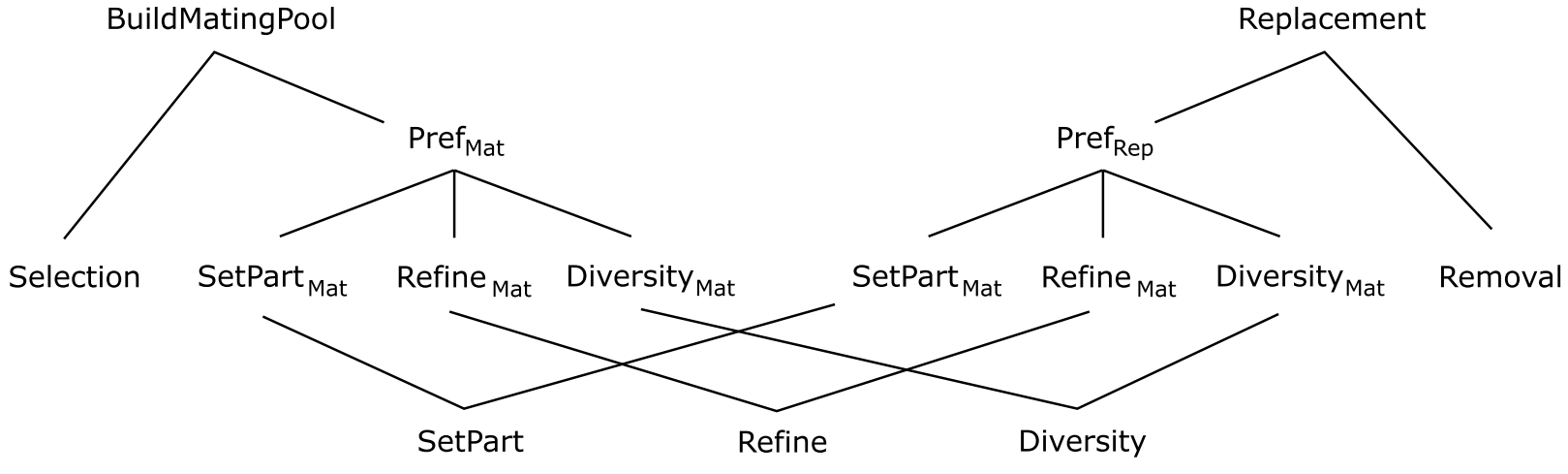


Figure A.1: Different abstraction levels used for the ablation analyses conducted in this appendix.

and, more importantly, to what extent. By contrast, when the goal of the investigation is to primarily identify high-performing intermediate designs that could possibly outperform the best-performing algorithm, the *best-to-worst* approach is more suited, as evidenced by the literature of path-relinking for combinatorial optimization [95]. In this appendix, we adopt the first approach given that our goal is to better understand how components affect performance and how components from high-performing algorithms could improve poor-performing ones. In three out of four PFSP scenarios (C_{max}-TFT, TFT-TT, and C_{max}-TFT-TT) we use NSGA-II as source and IBEA as target designs given that the former outperforms the latter[1]. On C_{max}-TT, IBEA is used as source and NSGA-II as target.

Component-wise template granularity refers to how the differences between source and target algorithmic designs are treated. For instance, different abstraction models for the ablations conducted in this appendix are depicted in Figure A.1. Using a little refined template granularity one could see the differences between IBEA and NSGA-II as components BuildMatingPool and Replacement. Refining the granularity of this model a bit further one sees that, in this particular case, changing component BuildMatingPool equals changing component Preference$_{Mat}$, as Selection is equal for both source and target algorithms. Conversely, Replacement can be further decomposed into two differing components Removal and Preference$_{Rep}$. Considering preference relations, one could (i) regard that components SetPart, Refinement, and Diversity should be changed altogether in both Preference$_{Mat}$ and Preference$_{Rep}$, as done in early MOEAs, or (ii) allow Preference$_{Mat}$ and Preference$_{Rep}$ to become different as in more recent MOEAs.

In this appendix, we simultaneously consider different granularity levels to demonstrate how this affects the ablation analysis. In particular, we consider all the components depicted in Figure A.1. We refer to components that cannot be further decomposed as *atomic*, and the remaining as *composite*. Moreover, a composite component may be discarded from the ablation options if at a given ablation step it becomes equivalent to another composite or atomic component. For instance, if Removal has already been used at some point in the ablation path, component Replacement becomes equivalent to component Preference$_{Rep}$, and hence it is discarded to avoid redundancy. The same can happen for component SetPart (or, analogously, for components Refinement or Diversity) in case the SetPart component from either Preference$_{Mat}$ or Preference$_{Rep}$ has already been used in the ablation path. Finally, the granularity refinement level also affects the convergence speed of the ablation procedure. To prevent concealing potentially relevant insights, we move along the ablation path by first identifying the statistically equivalent intermediate designs, and then selecting the one that changes the least possible components in the current step[2]. For example, if at a given step the

[1] Or is equivalent to, but presents lower rank sums, as in C_{max}-TFT-TT.
[2] In other words, we select the component that represents the most refined granularity model.

ablation identifies the three most-promising intermediate designs are obtained by changing components Replacement, Preference_{Rep}, or SetPart, we select the latter as it changes the least components possible.

Parameter configuration is an important issue in ablation as the changes in the algorithmic designs are likely to require different parameter settings. For instance, a component may be computationally demanding and require a smaller population size than previously adopted. However, given the number of intermediate designs produced during ablation, running a proper tuning campaign for each design is in general prohibitive. Several approaches can be taken, such as (i) using the parameter settings from either source or target algorithms for all intermediate designs or (ii) re-tuning each intermediate design with a limited tuning budget. In this work, we adopt the first approach, and consider the numerical parameters from both source and target algorithms. In more detail, we conduct two different ablation analyses for each scenario, the first using numerical parameters from the source algorithm and the second using the parameters from the target algorithm. However, we noticed that adopting numerical parameters from the target algorithm often produced more interesting insights (and better-performing intermediate designs). For brevity, we only present the results of this type of ablation. Nonetheless, when discussing the ablation results for a particular scenario, we make brief remarks about the ablation differences observed between the two setups.

Intermediate design evaluation can consider the whole test set or more computationally efficient approaches such as racing. In this appendix we adopt the former approach as the runtime-constrained setup we use for the PFSP makes this approach computationally cheap. The evaluation of an intermediate design is performed in the same way described in Section 3.3.

In the following section we present and discuss results for the different PFSP variants. In particular, we remark that many similarities were observed between the ablation for the variants where IBEA was used as target algorithm, as discussed next.

A.2 Ablation results

A general pattern can be observed when one analyzes the results from the three PFSP variants where NSGA-II is used as source and IBEA as target, namely $\text{C}_{\max}$-TFT, TFT-TT, and $\text{C}_{\max}$-TFT-TT. First, using parameters from IBEA provides much better performing intermediate designs. Concretely, many intermediate designs are able to improve even over IBEA, a remarkable finding. In addition, in most ablation steps the components that produce the most significant performance changes are atomic, providing further evidence for our claim about the importance of a refined component-wise approach to MOEAs. Finally, for all three scenarios the best-performing intermediate design is obtained by removing the dominance depth set-partitioning relation from the Preference_{Mat} component of NSGA-II. This result is consistent with the Preference_{Mat} components selected by irace for the PFSP AutoMOEAs, and reveals that a randomized mating selection approach is an effective strategy in the context of most PFSP variants. In particular, we believe this is explained by the computational overhead saved from not computing set partitioning or refinement metrics, which results in more function evaluations in runtime-constrained scenarios such as this one. More importantly, we observe a major performance difference between the intermediate designs where only the SetPart component from Preference_{Mat} (hereon called SetPart_{Mat}) is removed and the designs where the SetPart component from Preference_{Rep} (hereon called SetPart_{Rep}) is removed. In particular, removing SetPart_{Rep} leaves the algorithm with a Preference_{Rep} component that only accounts for diversity, and hence it cannot converge. However, the design choice of using an empty SetPart_{Mat} component and a non-empty SetPart_{Rep} component as selected by ablation would not be possible if different template granularities were considered (if only SetPart was considered, for instance).

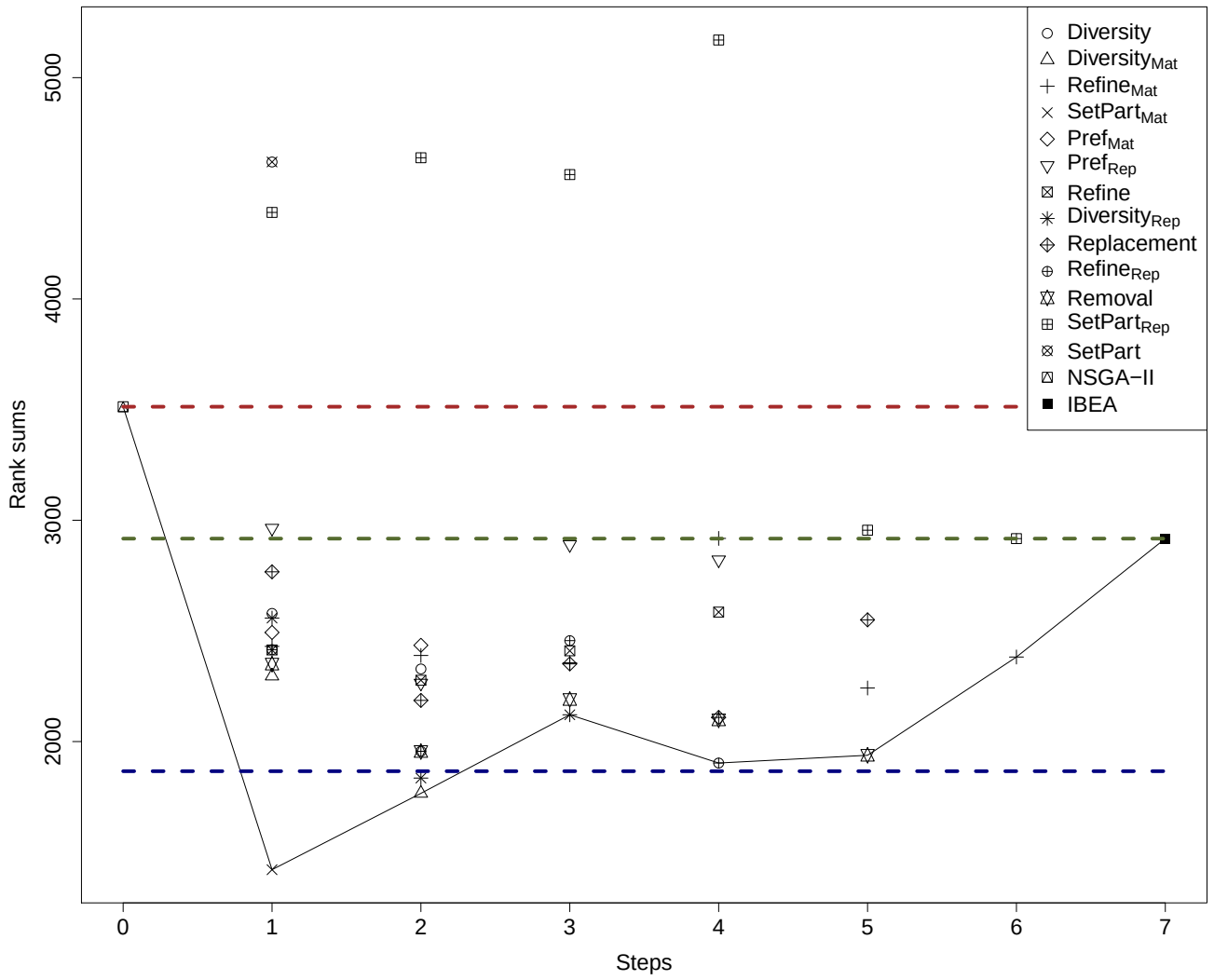


Figure A.2: Intermediate configurations tested for C_{max}-TFT, considering NSGA-II as source (step 0) and IBEA as target (step 7) algorithms. The solid line connects the changes that caused the largest performance improvement. Dashed lines represent the rank sums of source and target algorithms, and the rank sum threshold for statistically significant difference w.r.t. the best ranked design.

As previously discussed, these experimental results further reinforce the need for fine-grained, flexible component-wise MOEA frameworks. We next proceed to a variant-wise analysis.

C_{max}-TFT: All intermediate configurations tested in the analysis of C_{max}-TFT are shown in Fig. A.2. The y-axis represents the rank sums. The x-axis contains the steps of the procedure. In step 0, only the source algorithm is depicted, in this case NSGA-II. In step 1, we modify all possible components that differ between NSGA-II and IBEA, according to the different granularity levels previously explained, thus generating thirteen algorithms. In general, many components could be changed leading to improvements in the performance of NSGA-II, but clearly the best-performing intermediate design is obtained by altering component SetPart$_{Mat}$. As discussed above, the worst performances in step 1 are observed when SetPart$_{Rep}$ is changed (options SetPart$_{Rep}$ and SetPart). Over the following steps, it is not possible to further improve the performance of the intermediate design selected in step 1. On step 2, for instance, we notice the performance loss due to removing Diversity$_{Mat}$ (the Diversity component of Preference$_{Mat}$). Once again, this result is consistent with the design of AutoMOEA$_{C_{max}\text{-TFT}}$, given that crowding distance was selected by irace. It is also interesting to notice that changing Diversity$_{Rep}$ leads to a performance similar to that of changing Diversity$_{Mat}$, although changing both diversity components at once (option Diversity) leads to a significant decrease in performance, since in this case the algorithm would be completely insensitive to diversity. On step 3 it is also not possible to improve or to maintain an equivalent performance when changing any components, but replacing Diversity$_{Rep}$ is the change that least affects the algorithm. At this step, we remark that this intermediate design is nearly a multi-start approach, differing only from a complete random search by the SetPart$_{Rep}$ component that enforces

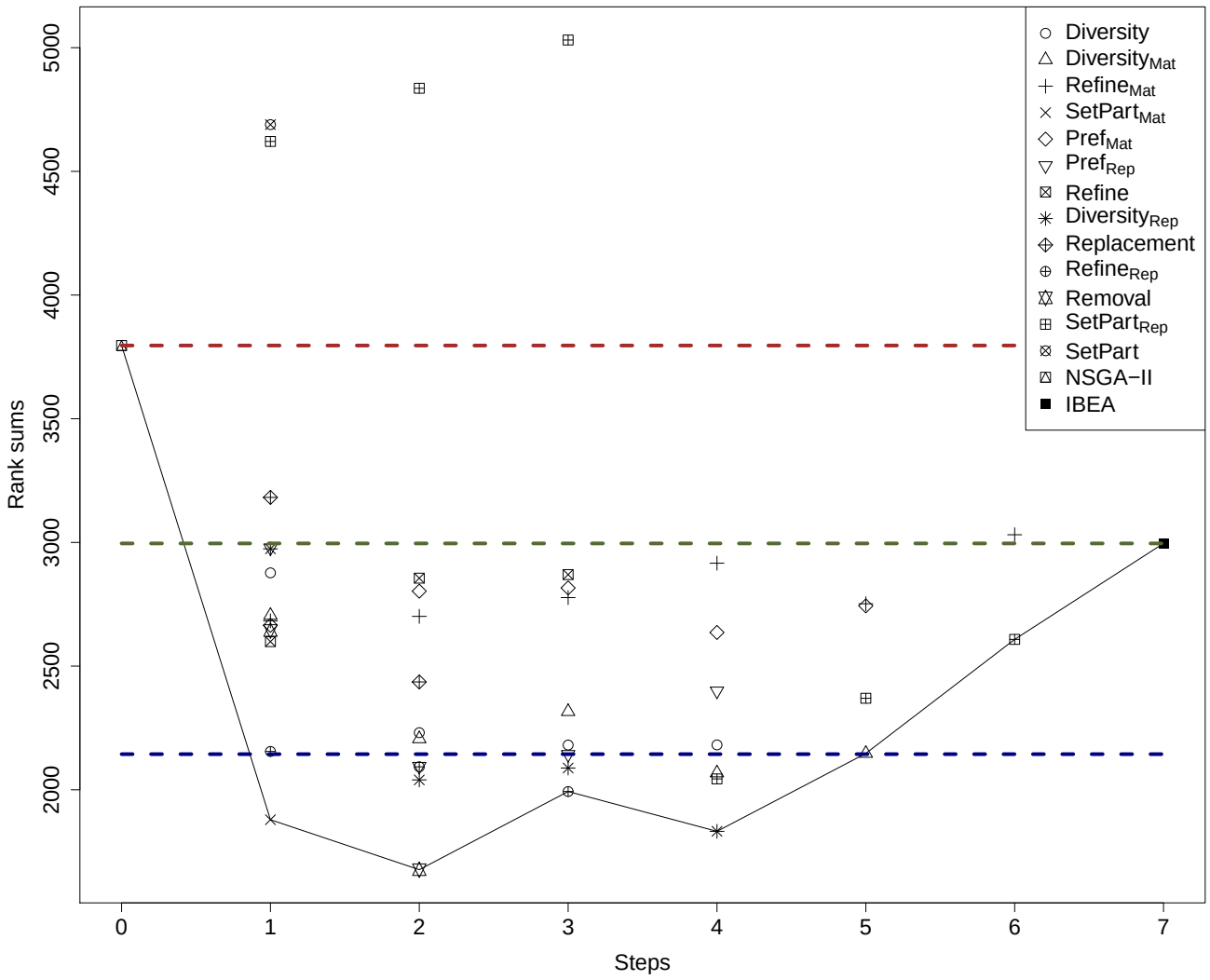


Figure A.3: Intermediate configurations tested for TFT-TT, considering NSGA-II as source (step 0) and IBEA as target (step 7) algorithms. The solid line connects the changes that caused the largest performance improvement. Dashed lines represent the rank sums of source and target algorithms.

convergence. However, the little computational overhead this intermediate design presents makes it better performing than IBEA, which had previously been considered the best-performing traditional MOEA for this variant. Nonetheless, we see the great difference in rank sums between this design and the more elaborate design selected at step 1.

At step 4, component Refinement$_{Rep}$ from IBEA is added, slightly improving performance. The next steps in the ablation procedure are only able to find worsening MOEA designs. At step 5, we notice that changing component Removal is the modification that least worsens the performance of the algorithm. In fact, this is an interesting example of how the benefits of the more accurate sequential removal from IBEA can be counterbalanced by the computation overhead it poses on runtime-constrained scenarios[3]. Finally, steps 6 and 7 show that using the Refinement$_{Mat}$ approach for IBEA or the SetPart$_{Rep}$ approach from NSGA-II worsen results even further. Altogether, the ablation path reveals that the best intermediate design for this variant is a combination of components from NSGA-II and IBEA, and that many intermediate designs are able to outperform the best-performing traditional MOEA used as target design.

TFT-TT: All intermediate configurations tested in the analysis of TFT-TT are shown in Fig. A.3, where once again NSGA-II is used as source design (step 0) and IBEA as target design (step 7). As previously discussed, SetPart$_{Mat}$ is the component that leads to the most significant performance improvement in step 1. In contrast to the C$_{max}$-TFT, it is still possible to further improve the configuration obtained at step 1 by replacing component Removal. In fact, for this variant the

[3]In particular, we remark that the sequential removal only affects SetPart$_{Rep}$ as for dominance depth both removal policies are equivalent.

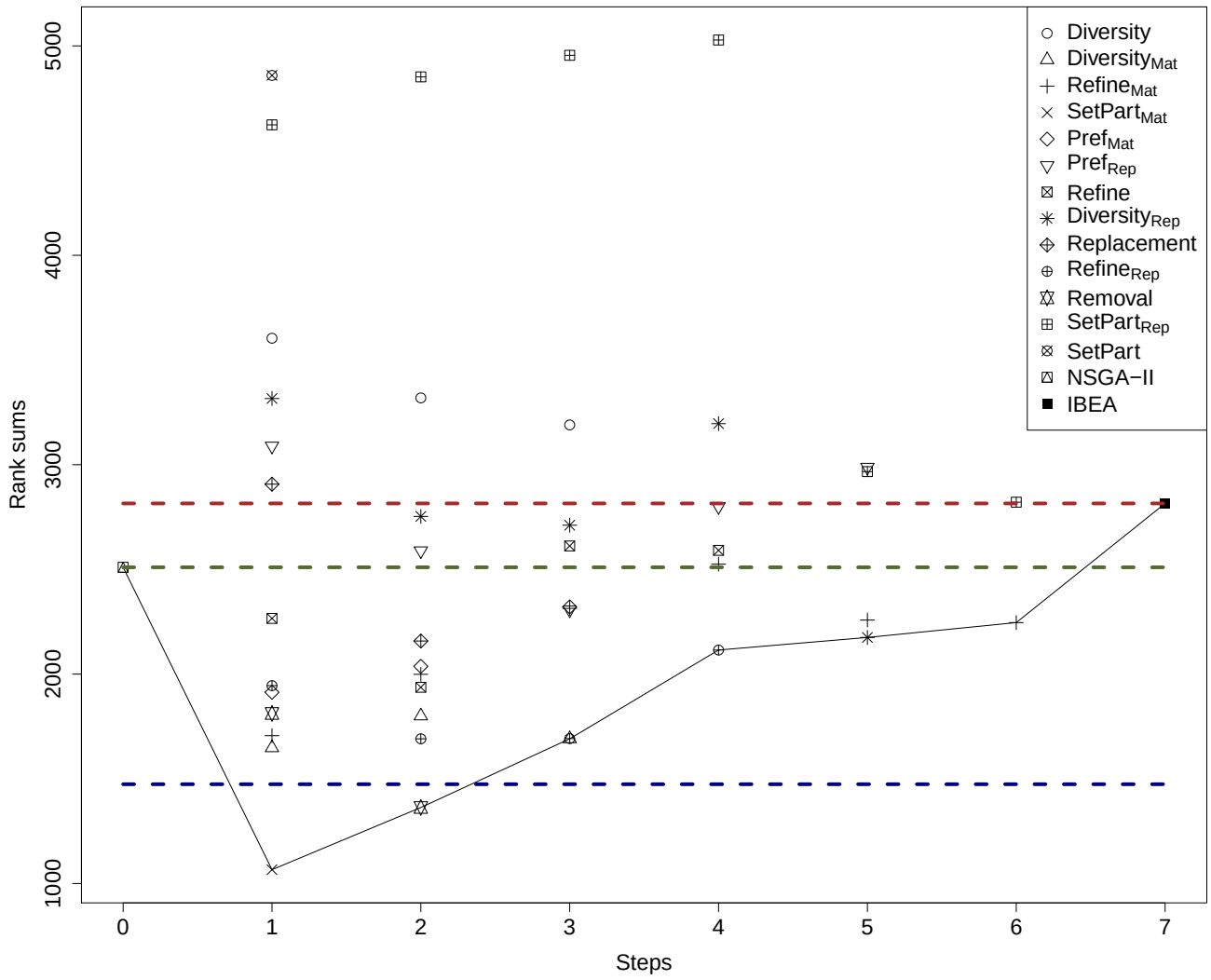


Figure A.4: Intermediate configurations tested for C_{max}-TFT-TT, considering NSGA-II as source (step 0) and IBEA as target (step 7) algorithms. The solid line connects the changes that caused the largest performance improvement. Dashed lines represent the rank sums of source and target algorithms.

configuration obtained at step 2 is the best-performing of all ablation, obtaining considerably lower rank sums than the target design. Given the design of step 1, this performance improvement is explained by a more accurate computation of Diversity$_{Rep}$, that indeed compensates the additional overhead posed by this removal policy. Next (step 3), no change to the current design leads to performance improvements, but adding the Refinement$_{Rep}$ component from IBEA is the change that least affects performance. It is interesting to notice that the following modification (step 4) is removing the Diversity$_{Rep}$ component from the design selected at step 3. In particular, this indicates that for this variant having crowding distance computed after using the binary epsilon indicator does not compensate the computational overhead posed by this diversity metric.

The next steps in the ablation path comprise worsening designs, respectively obtained by removing crowding distance from Preference$_{Mat}$ (step 5), dominance depth from Preference$_{Rep}$ (step 6), and finally by adding the binary epsilon score computation for Preference$_{Mat}$ (step 7). Once again, these results demonstrate that some components from each MOEA are not effective for the TFT-TT, and that a customized design is required to improve MOEA performance.

C_{max}-TFT-TT: All intermediate configurations tested in the analysis of C_{max}-TFT-TT are shown in Fig. A.4, where once again NSGA-II is used as source design (step 0) and IBEA as target design (step 7)[4]. Similar to the two previously discussed variants, SetPart$_{Mat}$ is the component that leads to the most significant performance improvement in step 1. Other similarities can be identified with the

[4]In particular, we remark that no statistically significant difference could be observed between IBEA and NSGA-II, and hence IBEA was selected as target design due to its smaller rank sum. However, we notice from the final ablation plot that the situation gets inverted when many similar intermediate designs are considered, with NSGA-II presenting lower rank sum than IBEA. Nonetheless, this does not affect our analysis.

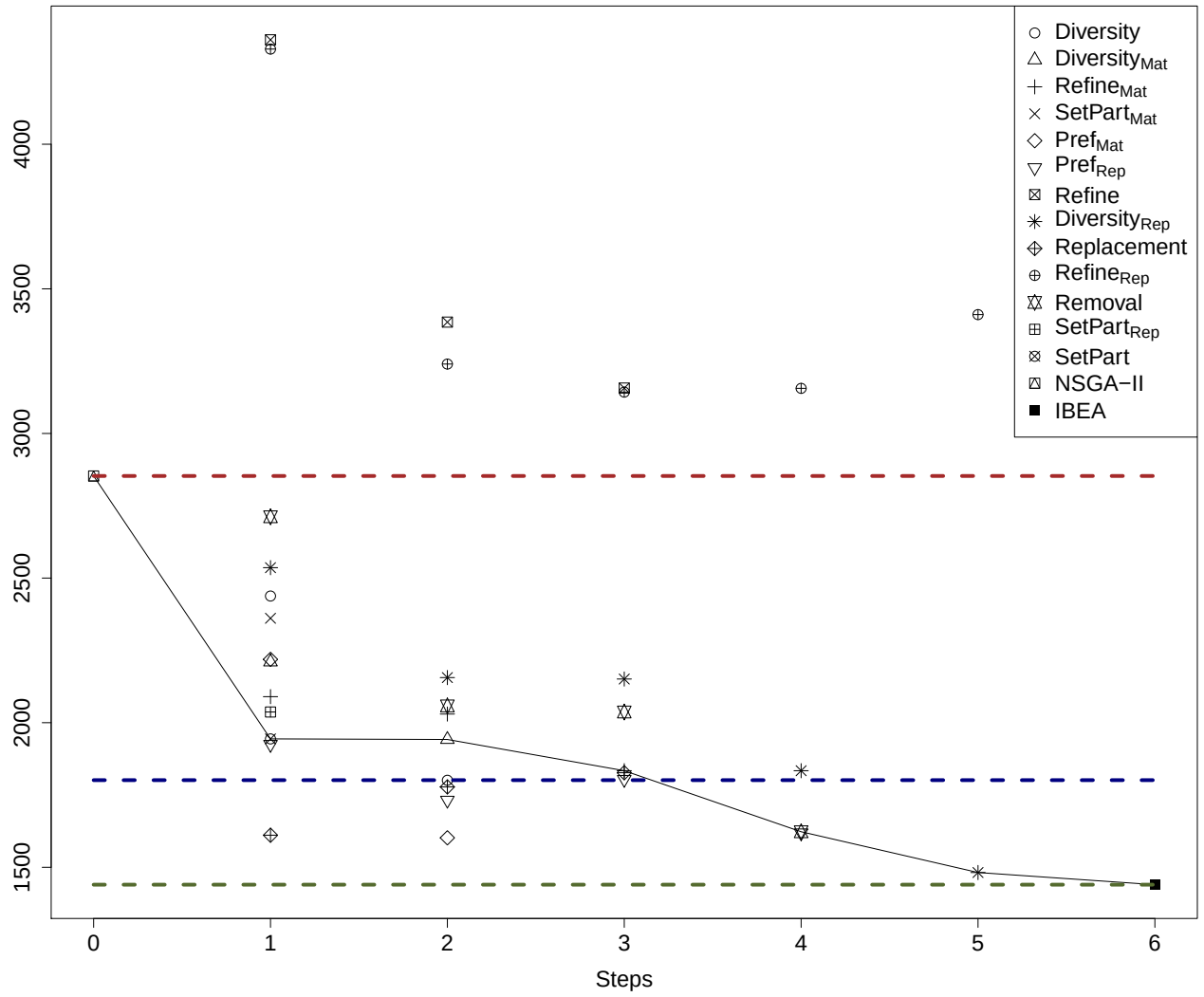


Figure A.5: Intermediate configurations tested for C_{max}-TT, considering IBEA as source (step 0) and NSGA-II as target (step 7) algorithms. The solid line connects the changes that caused the largest performance improvement. Dashed lines represent the rank sums of source and target algorithms.

previous scenarios, in that (i) only worsening designs can be obtained after step 1, like in C_{max}-TFT, and; (ii) Removal is the component selected at step 2. At step 3, an interesting observation concerns the very similar rank sums achieved by either removing Diversity$_{Mat}$ or Refinement$_{Rep}$, although the sequential changes of these two components lead to a significant performance loss, as observed in step 4. At step 5, removing Diversity$_{Rep}$ affects the performance of the algorithm very little since Refinement$_{Rep}$ has already been changed, a similar situation to what had been previously observed for the TFT-TT. In this case, though, adding a Refinement$_{Mat}$ component (step 6) does not alter the performance of the algorithm, indicating that for this variant the binary indicator mating approach of IBEA compensates the overhead it poses. Finally, removing SetPart$_{Rep}$ leads back to the original IBEA (step 7).

C_{max}-TT: All intermediate configurations tested for this variant are shown in Fig. A.5, a plot that differs considerably from the ablation plots discussed for the previous variants. This is partially explained by the difference in source and target designs, as IBEA is used as source (step 0) and NSGA-II is used as target (step 5). Nonetheless, this is the only variant where we observe that changing composite components provides more significant performance differences than changing atomic components. In addition, we are not able to identify any intermediate design that is better-performing than the target design. More importantly, this is the only variant where adopting numerical parameters from either source or target algorithms does not majorly influence the ablation path. We then proceed to a step-wise analysis as done for the previous variants. At step 1, the three ablation options that are considered equivalent are Replacement, Preference$_{Rep}$, and SetPart, indicating that for this variant the environmental selection approach of IBEA can be improved in a number different ways. As previously explained, we select SetPart as this is the composite option

that affects the least atomic components ($\mathsf{SetPart}_{Mat}$ and $\mathsf{SetPart}_{Rep}$). It is also interesting to observe that, differently from the previous variants, introducing a set-partitioning relation to $\mathsf{Preference}_{Mat}$ does not lead to performance losses (see the rank sum differences between $\mathsf{SetPart}_{Rep}$ and $\mathsf{SetPart}$ at step 1). Next (step 2), we see that changing composite components would again lead to performance improvements, but our selection criterion leads to changing only $\mathsf{Diversity}_{Mat}$. It is important to remark, though, that even changing composite components would not lead to an intermediate design better-performing than the target algorithm. From this step on, the ablation path becomes rather smooth, with selected changes of atomic components introducing small performance gains. The first such change (step 3) concerns removing $\mathsf{Refinement}_{Mat}$. In a sense, the combination of changes selected at steps 2–3 are the analogous situation to what happened for previous variants, where changes to a refinement and a diversity components were also conducted sequentially. At step 4 component $\mathsf{Removal}$ is changed, revealing that its computational cost no longer justifies its search benefits. Finally, at step 5 component $\mathsf{Diversity}_{Rep}$ is changed, once again followed by a change in component $\mathsf{Refinement}_{Rep}$. Overall, one sees that diversity and refinement components are directly connected, and that other template granularity models proposed for ablation purposes could try to consider them as a composite component, apart from set-partitioning.

A.3 Conclusions

Although restricted, the proof-of concept investigation conducted in this appendix demonstrated the effectiveness of coupling a component-wise view of MOEAs, such as the one proposed in this thesis, with iterative design-space analysis tools, such as the ablation analysis methodology. Concretely, we have considered the four PFSP variants adopted in this thesis and two high-performing MOEAs, namely IBEA and NSGA-II, deconstructing these algorithms into different composability models to investigate the effectiveness and interactions of specific components. The number of insights obtained is significant, the most important being (i) the need for algorithm customization required by an application domain that is rather structurally different from the one for which MOEAs are generally proposed; (ii) the effects of ablation setup factors, such as considering different granularity models and different numerical parameters used by intermediate designs, and (iii) the contrast between algorithmic components that interact and should be jointly modified, versus atomic components that should be individually changed to maximize the performance of a given design. In particular, we have shown for three out of the four variants considered in this investigation that it is possible to find algorithmic designs that improve over traditional MOEAs by modifying even a single algorithmic component. More importantly, the major contribution of this investigation is to demonstrate that flexible algorithmic engineering methodologies such as the one proposed in this work have the potential to overcome the current drawbacks faced by practitioners and researchers applying MOEAs to domains rather different from the ones they had been designed for.

APPENDIX B

A scenario-wise analysis of MOEA performance

The performance assessment of the state-of-the-art in MOEAs conducted in Chapter 4 produced a number of relevant insights that were discussed in that chapter. In this appendix, we conduct an in-depth analysis of each of the experimental scenarios considered in that assessment. In more detail, we first group results by the number of objectives, making comments on the overall results by means of a rank sum analysis. We then proceed to boxplot analysis of all metrics grouped by the number of FEs given to MOEAs.

B.1 Two-objective problems

Results for two-objective problems are given in Table B.1. For each row, we sort algorithms according to their rank sums when ran for the given number of FEs and assessed by the given performance metric. Algorithms highlighted (**boldface**) present rank sums considered statistically significantly lower than the others according to Friedman's test with 99% confidence level. As we will shortly discuss in more detail, we can observe three groups of algorithms according to their performance throughout scenarios. In general, SMS and IBEA are the algorithms that present best performance. The second group of algorithms comprises NSGA-II, SPEA2, HypE, MOEA/D and NSGA-III, which also present overall good performance, but are affected by specific function characteristics or by the FE budget they are given. Finally, the last group is formed by MO-CMA-ES and MOGA. The former often presents poor performance, in part due to its necessity of large FE budgets; the latter is consistently worse than almost all other MOEAs, both due to its lack of elitism and its lack of both limit-stability and limit-optimality.

We then proceed to a more detailed discussion grouping results by the number of function evaluations.

2500 FEs. Figure B.1 shows the performance of all MOEAs on selected problems when given 2 500 FEs. For brevity, we focus the discussion on the plots that better illustrate the rankings given in Table B.1. Moreover, the benchmark problems we consider here can be grouped according to similar characteristics or difficulty they pose for MOEAs, and the problems we depict in Figure B.1 are representative of each of these groups, as we will detail. Given the low budget of FEs, it is not surprising that many MOEAs are unable to converge to the optimal front. In general, MOEAs display good performance on the easiest DTLZ problems, namely DTLZ2 and DTLZ5. The exception to this pattern are the worst-ranked MOEAs, i.e., MOGA, MO-CMA-ES, and MOEA/D.

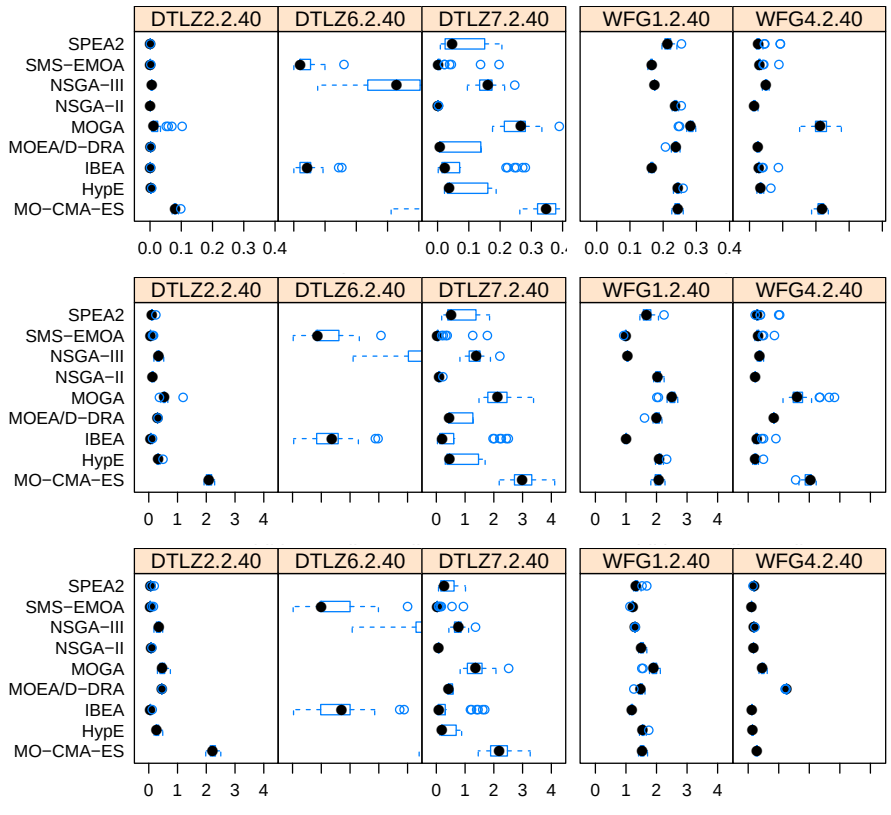


Figure B.1: Performances of MOEAs given 2500 FEs on selected two-objective problems with 40 variables. From top to bottom, I_H^{rpd}, $I_{\epsilon+}$ and I_{IGD}.

In fact, we remark that for this couple problems MO-CMA-ES ranks far worse than all other MOEAs according to all metrics. In addition, while MOEA/D performs slightly better than MOGA according to the I_H^{rpd} and the $I_{\epsilon+}^1$, it is outperformed according to the I_{IGD}. Another important observation is that, except for MO-CMA-ES, the I_H^{rpd} performance of all MOEAs look fairly similar, with differences being noticed mostly on the remaining indicators.

By contrast to the good performance displayed by nearly all MOEAs on the easiest DTLZ problems, their performance on the hardest problems (DTLZ1, DTLZ3, and DTLZ6) is astonishingly poor. In fact, DTLZ6 is the only problem where a few MOEAs are still able to get close to the actual front. The number of local optimal fronts make these problems too difficult for MOEAs when given a low budget of FEs, as we have discussed in Chapter 4. The only algorithms that are able to display reasonable performance are the best-ranked ones, namely SMS and IBEA. The moderately difficult DTLZ problems are the ones that present bias (DTLZ4) or a multi-modal, disconnected front (DTLZ7). The performance patterns on each of these problems is unique, so we address them individually. For DTLZ4, most MOEAs get trapped in biased regions of the search space, with SMS and IBEA being the only algorithms that are able to properly approximate the front on the majority of their runs. In addition, we remark that the performance of MO-CMA-ES according to the distance-based metrics is very poor. Concerning DTLZ7, SMS and NSGA-III are the algorithms that display best performance, being able to accurately approximate the Pareto optimal front on most their runs. In addition, MOEA/D is also able to display considerably good performance according to the I_H^{rpd}, but the distance-based metrics disagree. Again, MO-CMA-ES presents rather poor performance, being far worse than MOGA.

As for the WFG problems, these can be grouped into two major difficulty levels. The first, and harder, is formed by the two convex problems WFG1–2. As one can see from the plots for WFG1, no MOEA is able to reach the actual front. Nonetheless, SMS is the best-performing algorithm for this group of problems, followed by IBEA on WFG1 and by NSGA-III on WFG2. We also remark that the performance metrics disagree on these convex problems: while the I_H^{rpd} and the $I_{\epsilon+}^1$ show a large performance difference between MOEAs, the I_{IGD} shows a much narrower scenario. Nevertheless, all metrics agree that much improvement could be achieved by MOEAs in general. The second

Table B.1: Sum of ranks (in parenthesis) depicting the performance of MOEAs on two-objective problems.

2500 FEs									
I_H^{rpd}	**SMS**	**IBEA** (11)	NSGA-II (456)	SPEA2 (971)	MOEA/D (1972)	HypE (2124)	NSGA-III (2396)	CMA (5139)	MOGA (5622)
$I_{\epsilon+}$	**SMS**	**IBEA** (159)	NSGA-II (1806)	SPEA2 (2048)	HypE (2071)	NSGA-III (2446)	MOEA/D (3976)	CMA (5766)	MOGA (6017)
I_{IGD}	**SMS**	IBEA (490)	NSGA-III (2979)	SPEA2 (3025)	HypE (3152)	NSGA-II (3172)	CMA (5494)	MOEA/D (6026)	MOGA (6386)
10000 FEs									
I_H^{rpd}	**IBEA**	**SMS** (113)	SPEA2 (1149)	NSGA-II (1846)	MOEA/D (2819)	HypE (3593)	CMA (3731)	NSGA-III (4017)	MOGA (6682)
$I_{\epsilon+}$	**SMS**	IBEA (425)	SPEA2 (894)	NSGA-II (1993)	MOEA/D (3091)	HypE (3401)	CMA (3749)	NSGA-III (4231)	MOGA (6740)
I_{IGD}	**SMS**	SPEA2 (711)	IBEA (1387)	HypE (2382)	NSGA-II (2774)	MOEA/D (4237)	NSGA-III (5068)	CMA (5240)	MOGA (7297)
40000 FEs									
I_H^{rpd}	**SMS**	IBEA (453)	SPEA2 (791)	NSGA-II (1795)	MOEA/D (2269)	NSGA-III (3492)	CMA (3818)	HypE (3888)	MOGA (6788)
$I_{\epsilon+}$	**SMS**	SPEA2 (918)	IBEA (1197)	NSGA-II (2613)	MOEA/D (3053)	NSGA-III (3208)	CMA (3393)	HypE (4295)	MOGA (7060)
I_{IGD}	**SMS**	SPEA2 (731)	IBEA (2404)	MOEA/D (2833)	NSGA-III (3039)	NSGA-II (3409)	HypE (4252)	CMA (4446)	MOGA (7364)

Table B.2: Sum of ranks (in parenthesis) depicting the performance of MOEAs on three-objective problems.

2500 FEs									
I_H^{rpd}	**SMS**	IBEA (743)	HypE (1516)	MOEA/D (2421)	SPEA2 (3464)	NSGA-II (3950)	NSGA-III (4068)	CMA (4928)	MOGA (6727)
$I_{\epsilon+}$	**SMS**	**IBEA** (53)	MOEA/D (2456)	HypE (3102)	NSGA-II (3420)	NSGA-III (3580)	SPEA2 (4421)	CMA (5495)	MOGA (6352)
I_{IGD}	**IBEA**	MOEA/D (650)	SMS (711)	NSGA-II (1434)	NSGA-III (2710)	SPEA2 (3592)	HypE (3944)	CMA (4883)	MOGA (5420)
10000 FEs									
I_H^{rpd}	**SMS**	IBEA (556)	MOEA/D (1805)	HypE (2290)	SPEA2 (2302)	CMA (3616)	NSGA-II (4378)	NSGA-III (4627)	MOGA (7029)
$I_{\epsilon+}$	**SMS**	IBEA (494)	SPEA2 (2516)	CMA (2968)	HypE (3132)	MOEA/D (3552)	NSGA-III (4253)	NSGA-II (4885)	MOGA (7323)
I_{IGD}	**IBEA**	SMS (654)	SPEA2 (1041)	MOEA/D (1622)	HypE (2960)	NSGA-II (3640)	CMA (4240)	NSGA-III (4264)	MOGA (6886)
40000 FEs									
I_H^{rpd}	**SMS**	IBEA (715)	MOEA/D (1575)	SPEA2 (2715)	HypE (3510)	CMA (3604)	NSGA-II (4269)	NSGA-III (4276)	MOGA (7163)
$I_{\epsilon+}$	**SMS**	IBEA (994)	CMA (2614)	SPEA2 (2708)	MOEA/D (3137)	NSGA-II (4450)	NSGA-III (4774)	HypE (5284)	MOGA (7610)
I_{IGD}	**MOEA/D**	SMS (269)	SPEA2 (763)	IBEA (1668)	NSGA-II (2780)	CMA (3118)	NSGA-III (3984)	HypE (4815)	MOGA (6686)

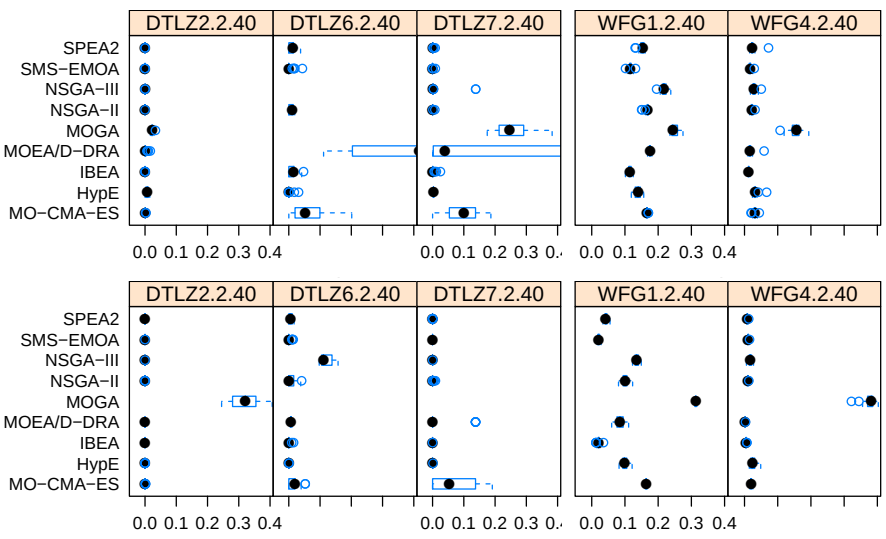


Figure B.2: Performances of MOEAs on selected two-objective problems with 40 variables. Since all metrics agree, we display only the I_H^{rpd}. On top, performance for 10 000 FEs. On the bottom, performance for 40 000 FEs.

group of WFG problems presents moderate difficulty for a few MOEAs. Although none is able to reach the front with the reduced number of FEs, results presented by MOEA/D, MO-CMA-ES, and MOGA are far worse than the ones presented by the remaining MOEAs. Once again, we see a considerable performance difference for a few algorithms depending on the metric considered, namely MO-CMA-ES and MOEA/D. For the I_H^{rpd} and the $I_{\epsilon+}^1$, the performance of MOEA/D is not very worse than most MOEAs, while MO-CMA-ES is the worst-ranking algorithm. By contrast, the I_{IGD} indicates the opposite.

10 000 FEs. The performance of all MOEAs when given 10 000 FEs is shown on Fig. B.2 (top). In particular, since the plots for all metrics are very similar, we only depict the results for the I_H^{rpd}. The first important difference we notice concerns the group of easy DTLZ problems, represented by DTLZ2. More specifically, the only MOEA that is still unable to reach the optimal front is MOGA. Concerning the hardest problems, the performance of all MOEAs is now much better on DTLZ6, but DTLZ1 and DTLZ3 remain unfeasible for all MOEAs. We also remark that the performance of MO-CMA-ES and the decomposition-based algorithms are particularly affected by the characteristics of this problem. Although the three MOEAs fail to correctly approximate the front, NSGA-III shows much worse performance than the other two, and MOEA/D is outperformed by a large margin by MO-CMA-ES. Finally, for the moderately difficult DTLZ problems (DTLZ4 and DTLZ7), most MOEAs are able to reach the optimal front, except for MOGA and MOEA/D on both problems, NSGA-III on DTLZ4, and MO-CMA-ES on DTLZ7.

When we analyze the performance of the algorithms on the WFG problems, we see that all algorithms benefit from the extra FEs on the hardest functions (WFG1–2). However, while SMS and HypE are able to correctly approximate the optimal front of WFG2, no MOEA is able to so on WFG1. Similarly to the previous scenario, SMS is the best-performing algorithm on this couple problems. Once again, NSGA-III is outperformed by all MOEAs but MOGA, indicating that this algorithms lacks robustness w.r.t. different problem characteristics. As for the remaining WFG functions, the performance of all MOEAs is greatly improved, with most of the indicator- and dominance-based algorithms performing nearly equivalently. The only exception to this pattern is MOGA, which still performs poorly.

40 000 FEs. The substantial increase in the number of function evaluations given to MOEAs reflects in performance improvements from most MOEAs which had failed to converge in previous scenarios, as we can see in Fig. B.2. On the easiest DTLZ, all MOEAs but MOGA converge to the actual fronts except for MOGA. On the moderate, MOEA/D and NSGA-III still face difficulties on DTLZ4,

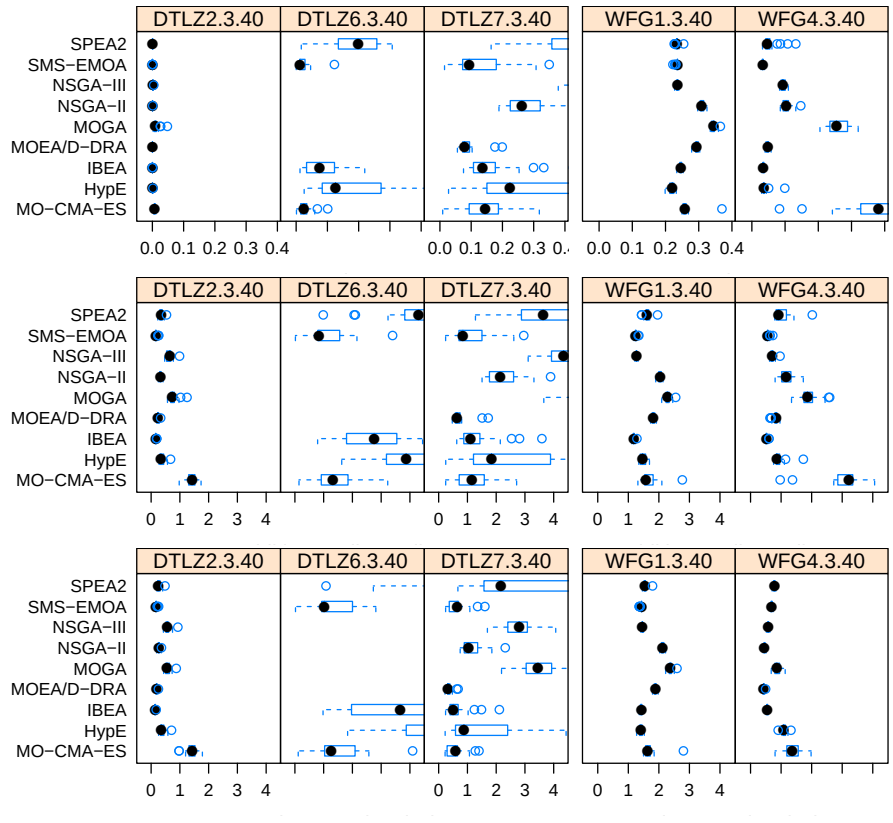


Figure B.3: Performances of MOEAs given 2 500 FEs on selected three-objective problems with 40 variables. From top to bottom, I_H^{rpd}, $I_{\epsilon+}$ and I_{IGD}.

whereas MO-CMA-ES struggles on DTLZ7. On the hardest, no MOEA is able to reach the optimal front for DTLZ1 and DTLZ3, indicating that these problems are probably unfeasible for MOEAs when moderate number of variables are used. Another important observation is that even with this increased FE budget, NSGA-III fails to accurately approximate the Pareto optimal front for DTLZ6. The same happens to MO-CMA-ES and MOEA/D on the larger n_{var} values.

Concerning WFG functions, most MOEAs are able to converge to the actual fronts on the WFG3–WFG9 functions, MOGA being the exception. The same performance improvements can be observed for WFG1, with SMS, IBEA, and SPEA2 reaching excellent results. By contrast, a lot of variability can be seen on the results of the best-performing MOEAs on WFG2 according to both the I_H^{rpd} and the $I_{\epsilon+}^1$, HypE being the exception. Finally, the decomposition-based and MO-CMA-ES display the worst performance on both these problems.

B.2 Three-objective problems

The rank sum analysis of the performance presented by all MOEAs on three-objective problems is given in Table B.2. In general, the same patterns observed for the two-objective problems can be seen in this case, i.e., algorithms can be clustered into three different groups according to performance. Once again, SMS and IBEA comprise the best-performing group, but this time only MOGA comprises the worst-performing one. In addition, overall performances of HypE, MOEA/D and MO-CMA-ES are greatly improved. In the case of the MO-CMA-ES, we notice that both this algorithm and SPEA2 display much worse performance when given a low FE budget. The opposite happens to HypE, as we discuss in more detail next.

2 500 FEs. Figure B.3 shows the performance of all MOEAs on selected problems when given 2500 FEs. For the easiest DTLZ problems (DTLZ2 and DTLZ5), MOEAs are able to display good performance according to the I_H^{rpd}, but the distance-based metrics favor SMS and IBEA over all other algorithms. In particular, MO-CMA-ES is the worst-performing algorithm, although we remark NSGA-III and performs as poorly as MOGA. The increase in the number of objectives

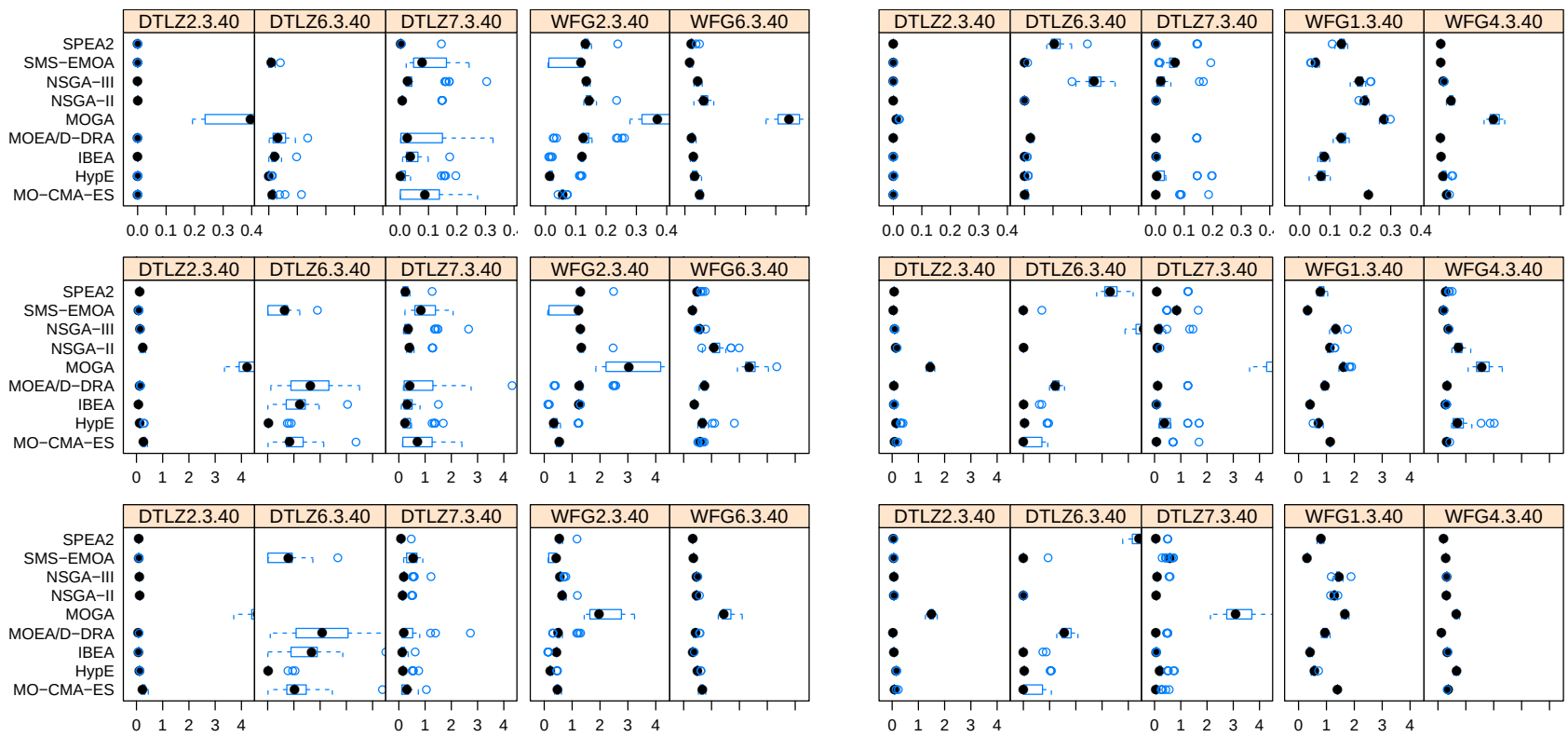


Figure B.4: Performances of MOEAs given 10 000 FEs (left) and 40 000 FEs (right) on selected three-objective problems with 40 variables. From top to bottom, I_H^{rpd}, $I_{\epsilon+}^1$, and I_{IGD} performances.

poses more difficulty for MOEAs also for the moderate DTLZ functions (DTLZ4 and DTLZ7). Once again, SMS and IBEA are the best-performing algorithms, although we see that, particularly for DTLZ7, SPEA2 faces a significant challenge, whereas MO-CMA-ES appears to deal with its characteristics quite favorably. In fact, this algorithm outperforms all other MOEAs on DTLZ7 when $n_{\text{var}} = 30$, but is not able to maintain its top-performing behavior on larger n_{var} values. Concerning metrics, two very different patterns are observed. For DTLZ4, the distance-based metrics agree and show that many MOEAs have much room to improve, whereas the I_H^{rpd} would seem to indicate that all algorithms have successfully converged. By contrast, on DTLZ7 the opposite happens, with I_H^{rpd} results being worse than the distance-based metrics. For the hardest DTLZ problems, two different situations are observed. For DTLZ1 and DTLZ3, again no MOEA is able to approximate the fronts. For DTLZ6, however, the performance of some MOEAs is actually better than for the two-objective DTLZ6, in particular HypE, MO-CMA-ES, and SPEA2. In fact, MO-CMA-ES is only outperformed by SMS regardless of n_{var}.

Concerning the WFG problems, the group comprising the hardest ones (WFG1 and WFG2) pose a challenge similar to that observed on $M = 2$ scenarios. No algorithm is able present reasonable results whatever the performance metric considered on both problems, although some algorithms clearly perform better than others. The only exception to this pattern are SMS and IBEA, which perform well according to all metrics on WFG2 with $n_{\text{var}} = 30$, and HypE which performs well on the same problem but only according to the I_H^{rpd}. For the remaining WFG problems (represented by WFG4) we notice that algorithms are unable to reach the Pareto optimal front, although results can be considered reasonable given the limited FE budget MOEAs are allowed to use. Nonetheless, the only algorithm that is not affected by specific problem characteristics from this WFG subset is IBEA. In fact, the concave WFG problems are directly responsible for the good ranking of MOEA/D according to the I_{IGD}. In addition, the same indicator ranks SMS as the third-best MOEA, a significant difference w.r.t. the rest of the metrics. This fact is explained by the performance of SMS on the problems that present parameter-dependent bias, i.e., WFG7–9, where this MOEA performs slightly worse than the top two ranked algorithms.

10 000 FEs. Boxplots depicting the performance of MOEAs when ran for 10 000 FEs are given in

Fig. B.4 (left). We initially make a few remarks concerning performance metrics. On the DTLZ benchmark, all metrics agree, although two important exceptions can be observed. Firstly, on DTLZ6 the I_H^{rpd} agrees with the other metrics, although it seems to indicate a better performance from MOEAs in general than the remaining metrics. The same pattern is observed on DTLZ7, but now it is the I_{IGD} that indicates a better performance than the other metrics. This also happens on WFG2, the other disconnected problem, and on the concave WFG problems. On the remaining convex WFG problems (WFG1 and WFG3), all metrics agree.

In general, the approximation fronts produced by MOEAs now present much better performance, whatever the metric considered. For the easiest DTLZ problems, all MOEAs are now able to accurately approximate the fronts. On the moderate ones, we see two different situations. On DTLZ4, improvements are more clear on the distance-based metrics, whereas I_H^{rpd} improvements are mostly seen when $n_{\text{var}} = 50$. Conversely, on DTLZ7 many MOEAs display major performance gains. The most significant exception is SMS, which is not able to improve over its performance when given $FE_{\max} = 2\,500$. In fact, on this particular problem SMS would outrank only MOEA/D, which presents great variability, and MOGA. For the hardest DTLZ functions, over half of the MOEAs is able to present reasonable results on DTLZ6, but none on DTLZ1 and DTLZ3. However, even on DTLZ6 we remark that the MOEAs that display very good performance on smaller n_{var} values face much more difficulties when this factor is increased. Concerning WFG problems, all algorithms show performance improvements on all functions, but none is able to reach the actual fronts. The major exception is WFG2, where some MOEAs improve their performances on extreme n_{var} values at the cost of a worsened performance on $n_{\text{var}} = 40$.

40 000 FEs. The increase in the computational budget reflects in different performance improvement rates that vary according to the group of problems considered, as shown in Fig. B.4 (right). On both the easiest and the moderate DTLZ problems, most algorithms are able to properly approximate the optimal fronts. The most surprising exception is SMS, which fails to converge on DTLZ7 even when given this increased FE budget. On the hardest WFG problems, algorithms are also able to display improved performances, although on the WFG2 problem this is only observed more clearly on specific n_{var} values. Nonetheless, the only MOEAs that correctly approximate the optimal front on this problem are SMS and IBEA, and none is able to do so on WFG1. Finally, on the concave WFG problems all MOEAs show improvements, in particular according to the distance-based metrics, and even more so according to the I_{IGD}. Nonetheless, MOEAs are unable to converge to the actual fronts, even though some algorithms get very close to that goal. Finally, we remark that the lower relative performance of IBEA according to the I_{IGD} is explained by the improvements of other MOEAs on the WFG concave functions rather than by a worsening in the performance of IBEA.

B.3 Five-objective problems

As discussed in Chapter 4, the increase in the difficulty level of the problems is much more substantial for some problems than for others. For the DTLZ problems, we discard DTLZ1 and DTLZ3 from our analysis, as no MOEA is able to produce reasonable results for any of them. We then re-categorize the remaining DTLZ problems into two groups: (i) the hardest, comprising DTLZ6 and DTLZ7, which now becomes more difficult than DTLZ6, and; (ii) the moderate, comprising DTLZ4 and the formerly easy DTLZ2 and DTLZ5. Concerning the WFG benchmark, regardless of the number of function evaluations, the performance metric and the MOEA considered, we notice that the convex problems, which before posed significant difficulty for all algorithms, now become easier to solve than the concave ones. Initial considerations aside, the rank sum analysis of the performance presented by all MOEAs on five-objective problems is given in Table B.3. Based on their performance, MOEAs can be clustered into nearly the same

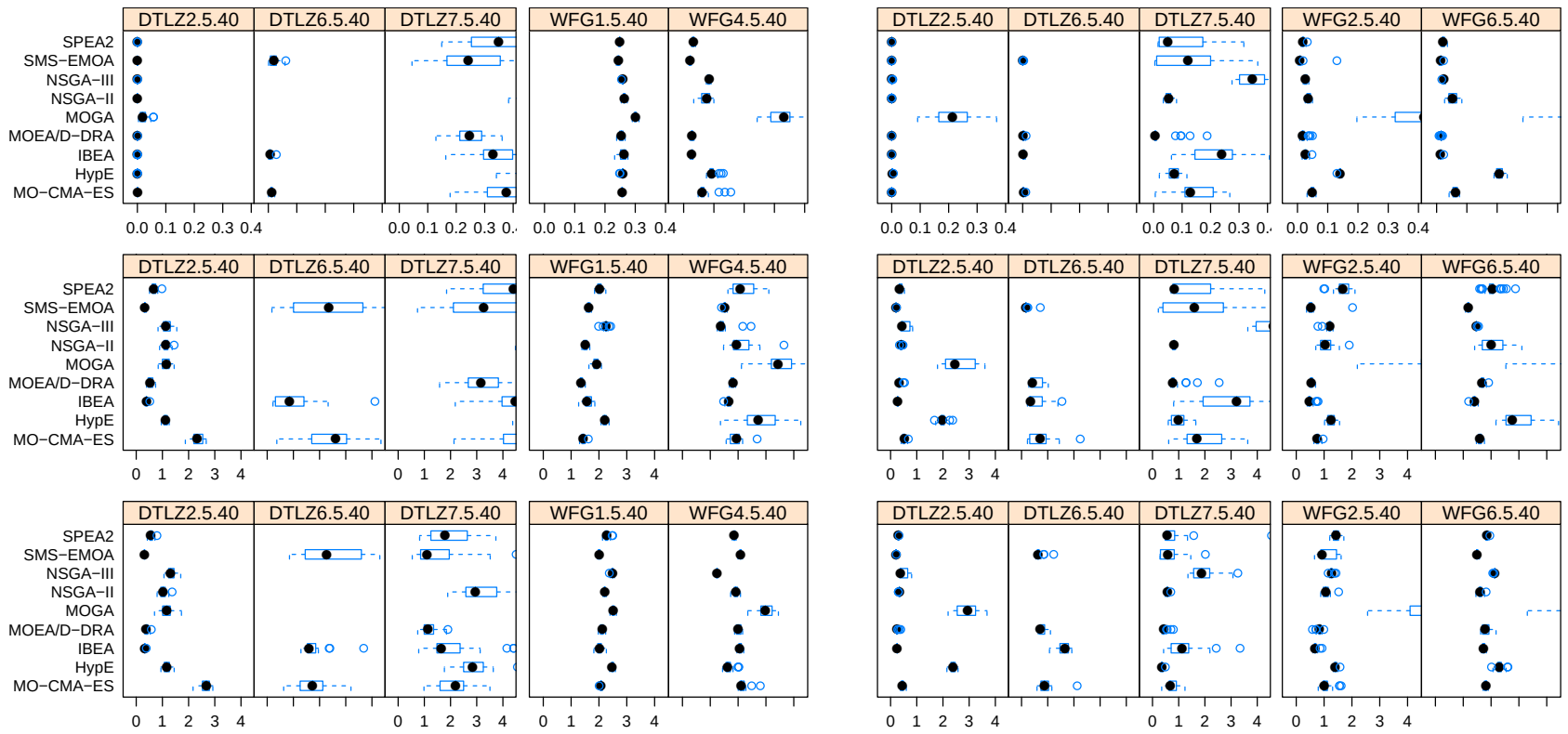


Figure B.5: Performances of MOEAs given 2 500 FEs (left) and 10 000 FEs (right) on selected five-objective problems with 40 variables. From top to bottom, I_H^{rpd}, $I_{\epsilon+}$ and I_{IGD}.

three groups we highlighted for three-objective problems. The major exception concerns HypE, which is now often the highest-ranked of the MOEAs that comprise the intermediate-performance group. In addition, we notice that the rankings of NSGA-III vary considerably depending on the metric considered on $FE_{\max} = 2\,500$ scenarios. We then proceed to a detailed discussion grouped by FE budget.

2 500 FEs. Boxplots depicting the performance of all MOEAs on three-objective problems when given 2 500 FEs are shown in Fig. B.5 (left). From the distance-related metrics one can confirm what we previously discussed about DTLZ2, as clearly this problem becomes a challenge for most MOEAs when a limited number of FEs is allowed. In fact, the only algorithms to converge to the actual front of DTLZ2 are SMS and IBEA. We remark that this problem is a clear example of the different metrics behavior, as the I_H^{rpd} would make it seem that this is an easy problem for all MOEAs. The performance displayed by MOEAs on the other moderate DTLZ problems (DTLZ4–5) is very similar to the one depicted for DTLZ2, confirming that the number of objectives affects the difficulty level of the problems in different degrees. In general, the other MOEA that performs nearly equivalently to the two best is MOEA/D. The most important exception concerns NSGA-III, which ranks first according to the I_{IGD} metric on DTLZ5.

For the hardest DTLZ problems (DTLZ6–7), we see two similar, yet slightly different situations. In common, no MOEA is able to converge to the actual Pareto front with the limited number of FEs they are given. However, while SMS, IBEA, and MO-CMA-ES are the algorithms that perform best on DTLZ6, MOEA/D is best-performing algorithm on DTLZ7 when all metrics are considered. Nonetheless, we make a few remarks concerning SMS. First, on DTLZ6 its performance is very affected by the increase in n_{var}, and so is the performance of MO-CMA-ES. Second, while the I_H^{rpd} indicates that the performances of SMS and MOEA/D are similar on DTLZ7, the distance-based metrics show a big gap between these two algorithms.

Concerning the WFG benchmark, we start our analysis with the non-concave problems, i.e., the convex WFG1–2 and the mixed linear-convex WFG3. In general, no MOEA is able to correctly approximate the optimal fronts and that it is difficult to extract patterns from those plots, but we focus on two important observations. First, on WFG2–3 SMS performs quite poorly according to the I_H^{rpd}, but much better according to the distance-based metrics. Second, although many MOEAs

Table B.3: Sum of ranks (in parenthesis) depicting the performance of MOEAs on five-objective problems.

2500 FEs									
I_H^{rpd}	**SMS**	IBEA (1402)	MOEA/D (1603)	SPEA2 (3375)	NSGA-II (4245)	CMA (4274)	NSGA-III (4731)	HypE (4765)	MOGA (7463)
$I_{\epsilon+}$	**SMS**	IBEA (745)	MOEA/D (1221)	NSGA-III (2898)	NSGA-II (3766)	SPEA2 (3785)	HypE (4108)	CMA (4365)	MOGA (5588)
I_{IGD}	**SMS**	**NSGA-III** (57)	**IBEA** (242)	MOEA/D (610)	HypE (1034)	SPEA2 (1052)	NSGA-II (1931)	CMA (3214)	MOGA (4503)
10000 FEs									
I_H^{rpd}	**SMS**	MOEA/D (1471)	IBEA (1535)	SPEA2 (3295)	CMA (3374)	NSGA-III (3897)	NSGA-II (4130)	HypE (5811)	MOGA (7562)
$I_{\epsilon+}$	**SMS**	IBEA (1713)	MOEA/D (2569)	CMA (2588)	NSGA-II (4086)	NSGA-III (4124)	SPEA2 (4701)	HypE (5907)	MOGA (7664)
I_{IGD}	**SMS**	IBEA (1898)	MOEA/D (2119)	NSGA-II (2329)	CMA (2515)	SPEA2 (3579)	HypE (5040)	NSGA-III (5225)	MOGA (7398)
40000 FEs									
I_H^{rpd}	**SMS**	IBEA (1154)	MOEA/D (1761)	CMA (2915)	SPEA2 (2995)	NSGA-III (3325)	NSGA-II (4188)	HypE (5826)	MOGA (7363)
$I_{\epsilon+}$	**SMS**	IBEA (173)	CMA (1610)	MOEA/D (1650)	SPEA2 (3310)	NSGA-II (3988)	NSGA-III (4436)	HypE (5777)	MOGA (7097)
I_{IGD}	**IBEA**	MOEA/D (244)	SMS (827)	SPEA2 (1720)	CMA (2074)	NSGA-II (2737)	NSGA-III (5087)	HypE (5189)	MOGA (6780)

Table B.4: Sum of ranks (in parenthesis) depicting the performance of MOEAs on ten-objective problems.

2500 FEs									
I_H^{rpd}	**SMS**	IBEA (827)	CMA (1919)	NSGA-II (2965)	NSGA-III (3543)	SPEA2 (4257)	HypE (5525)	MOEA/D (6218)	MOGA (7505)
$I_{\epsilon+}$	**MOEA/D**	IBEA (370)	SMS (2092)	NSGA-II (2471)	CMA (2791)	NSGA-III (3315)	SPEA2 (3498)	HypE (4878)	MOGA (6297)
I_{IGD}	**IBEA**	NSGA-III (847)	SPEA2 (1272)	CMA (1289)	SMS (1915)	NSGA-II (2228)	HypE (3315)	MOEA/D (4441)	MOGA (5562)
10000 FEs									
I_H^{rpd}	**IBEA**	SMS (222)	CMA (1116)	NSGA-III (2326)	SPEA2 (2532)	NSGA-II (3241)	HypE (4846)	MOEA/D (5114)	MOGA (6919)
$I_{\epsilon+}$	**MOEA/D**	IBEA (1258)	SMS (2794)	CMA (3250)	NSGA-III (3347)	NSGA-II (4045)	SPEA2 (4562)	HypE (5214)	MOGA (6922)
I_{IGD}	**NSGA-III**	**IBEA** (36)	SPEA2 (646)	NSGA-II (1776)	SMS (1828)	CMA (1987)	HypE (2557)	MOEA/D (4424)	MOGA (5151)
40000 FEs									
I_H^{rpd}	**IBEA**	SMS (897)	SPEA2 (1401)	CMA (1878)	NSGA-III (2416)	NSGA-II (2576)	HypE (4899)	MOEA/D (5077)	MOGA (7073)
$I_{\epsilon+}$	**MOEA/D**	**IBEA** (220)	NSGA-III (1836)	SMS (2309)	NSGA-II (2986)	SPEA2 (3252)	CMA (3705)	HypE (4581)	MOGA (6518)
I_{IGD}	**IBEA**	NSGA-III (928)	SPEA2 (942)	NSGA-II (2710)	HypE (2937)	CMA (3725)	SMS (4129)	MOEA/D (5513)	MOGA (6584)

perform similarly across these functions, the overall best-performing MOEAs are SMS, IBEA, and MOEA/D. As for the concave WFG problems, the patterns in the boxplots depend considerably on the given metric. According to the I_H^{rpd}, NSGA-III shows the best performance, although many MOEAs display similar performance when $n_{\text{var}} = 50$. By contrast, the distance-based metrics favors SMS, followed by IBEA and NSGA-III on the $I_{\epsilon+}^1$, and by IBEA, MOEA/D, and SPEA2 on the I_{IGD}.

10 000 FEs. The increase in the number of function evaluations given to MOEAs reflects on performance improvements in nearly all problems and according to all metrics, as shown in Fig. B.5 (right). For all problems considered, no MOEA is able to properly approximate the optimal fronts. Concerning the DTLZ functions, we notice that the same group of MOEAs is able to display the best performance among all algorithms on problems DTLZ2 and DTLZ4–6 when all metrics and n_{var} values are considered altogether, namely SMS, IBEA, MOEA/D and MO-CMA-ES. By contrast, on DTLZ7 most algorithms perform similarly according to the I_{IGD}, but the remaining metrics favor MOEA/D considerably over the other MOEAs.

Regarding the non-concave WFG problems represented in Fig. B.5 by WFG1, we see different performance patterns. For WFG1 and WFG3, all metrics agree and indicate that SMS is the best-performing algorithm, followed by SPEA2 on WFG1 and by MOEA/D on WFG3. By contrast, performance differences on WFG2 are very clear when one compares results on $n_{\text{var}} = 30$ and $n_{\text{var}} = 50$. Overall, SMS, IBEA, and MOEA/D can be considered the best-performing algorithms for this function, but the performance of SMS according to the I_{IGD} is much worse than for the other metrics. As for the concave WFG problems, we see again these three algorithms being considered best, but we remark that NSGA-III would have made it into that group if not for its performance according to the I_{IGD}.

40 000 FEs. Boxplots depicting MOEA performances when given 40 000 FEs are shown in Fig. B.6. Overall, the performance gains are clear for all MOEAs according to all metrics, except for the concave WFG problems where no significant I_H^{rpd} improvements can be seen. Nonetheless, once again no MOEA is able to successfully approximate the actual fronts, even though many MOEAs get very close to it on DTLZ2. Concerning the moderate DTLZ problems, the best-performing MOEAs get reasonably close to the actual fronts, namely SMS, IBEA, MOEA/D, and MO-CMA-ES, although the latter two lose performance on DTLZ4 when n_{var} is increased. These same four MOEAs repeats their good performance on DTLZ6. However, while other MOEAs were able to show reasonable results on the moderate problems, this time the performance of the remaining algorithms is quite poor. Finally, a peculiar result can be observed on DTLZ7: algorithms that displayed good performance on most other problems, namely SMS, IBEA, MOEA/D, and NSGA-III, are outranked by other MOEAs such as SPEA2, NSGA-II, and HypE.

On the non-concave WFG problems, many MOEAs show significant performance improvements, although it becomes clear that MOEAs are unable to converge to the actual fronts on these problems. Once again, results on WFG1 are fairly consistent across the different metrics. SMS ranks well on all these problems and across all metrics, but while it is the best-performing MOEA for WFG1–2, it is outperformed by MOEA/D on WFG3. In addition, IBEA also ranks well on WFG2. As for the concave WFG problems, we notice major discrepancies between performance metrics. For instance, SMS and NSGA-III rank very well according to the I_H^{rpd}, but on the distance-based metrics the performance of NSGA-III is not as competitive, and the same happens for SMS according to the I_{IGD}. Similarly, MOEA/D performs well according to the I_H^{rpd} and to the I_{IGD}, but is not as competitive according to the $I_{\epsilon+}^1$. Overall, the only MOEA that can be considered as well-performing according to all metrics is IBEA.

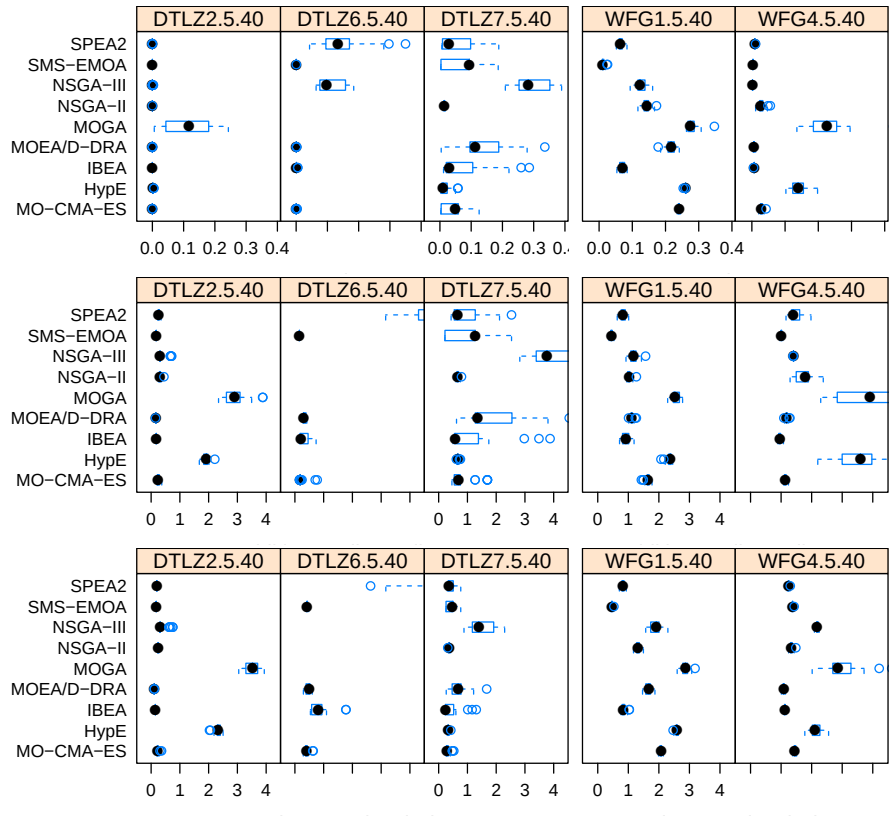


Figure B.6: Performances of MOEAs given 40 000 FEs on selected five-objective problems with 40 variables. From top to bottom, I_H^{rpd}, $I_{\epsilon+}$ and I_{IGD}.

B.4 Ten-objective problems

The large increase in the number of objectives leads to important changes in the rank sum analysis we show in Table B.4. First, as previously discussed, the disparity in rankings according to the different metrics becomes considerable. One could even say that selecting a best-performing MOEA becomes itself a multi-objective task, since only IBEA is able to rank well according to multiple metrics. Even so, other algorithms outrank it on specific metrics, corroborating that metrics play a critical role both for design and assessment on truly many-objective scenarios.

A very important observation that yet had not been reported in the literature is the low rank sums achieved by NSGA-II and SPEA2 in all ten-objective scenarios. Two precautions taken in this thesis are directly related to this good performance. First, both algorithms benefit directly from proper tuning, as the numerical parameters selected by irace differ considerably from the default adopted in the literature. Second, SPEA2 uses DE as underlying algorithm when given 10 000 and 40 000 FEs, reinforcing the need to consider different underlying EA algorithms when proposing a MOEA. Next, we detail our discussion for each budget considered.

2 500 FEs. Results depicted in the boxplots given in Fig. B.7 (right) show that the increase in the number of objectives makes several problems too difficult for MOEAs to solve when only a few FEs are allowed. More precisely, the only problem groups for which we see reasonable results are the moderate DTLZ problems (DTLZ2 and DTLZ4–5) and the non-concave WFG problems (WFG1–3). However, we remark that results for the $I_{\epsilon+}$ on WFG3 in general are not as good as on WFG1–2. On the selected problems depicted on Fig. B.7 (right), we see a contrast between metrics in opposite directions. While on DTLZ2 the performance of MOEAs look perfect, but the distance-based metrics show many differences between algorithms. In fact, the only MOEAs that consistently demonstrate good performance on the moderate DTLZ problems are SMS, IBEA, and MOEA/D. As for the hardest DTLZ problems, we see very poor performances from all MOEAs, with two major exceptions. First, I_H^{rpd} results for DTLZ6 when $n_{\text{var}} = 31$ look promising for SMS, IBEA, and MO-CMA-ES, but we remark that the distance-metrics disagree with this, except for SMS. Second, I_{IGD} results for SMS on DTLZ7 look reasonable, but the remaining metrics contradict

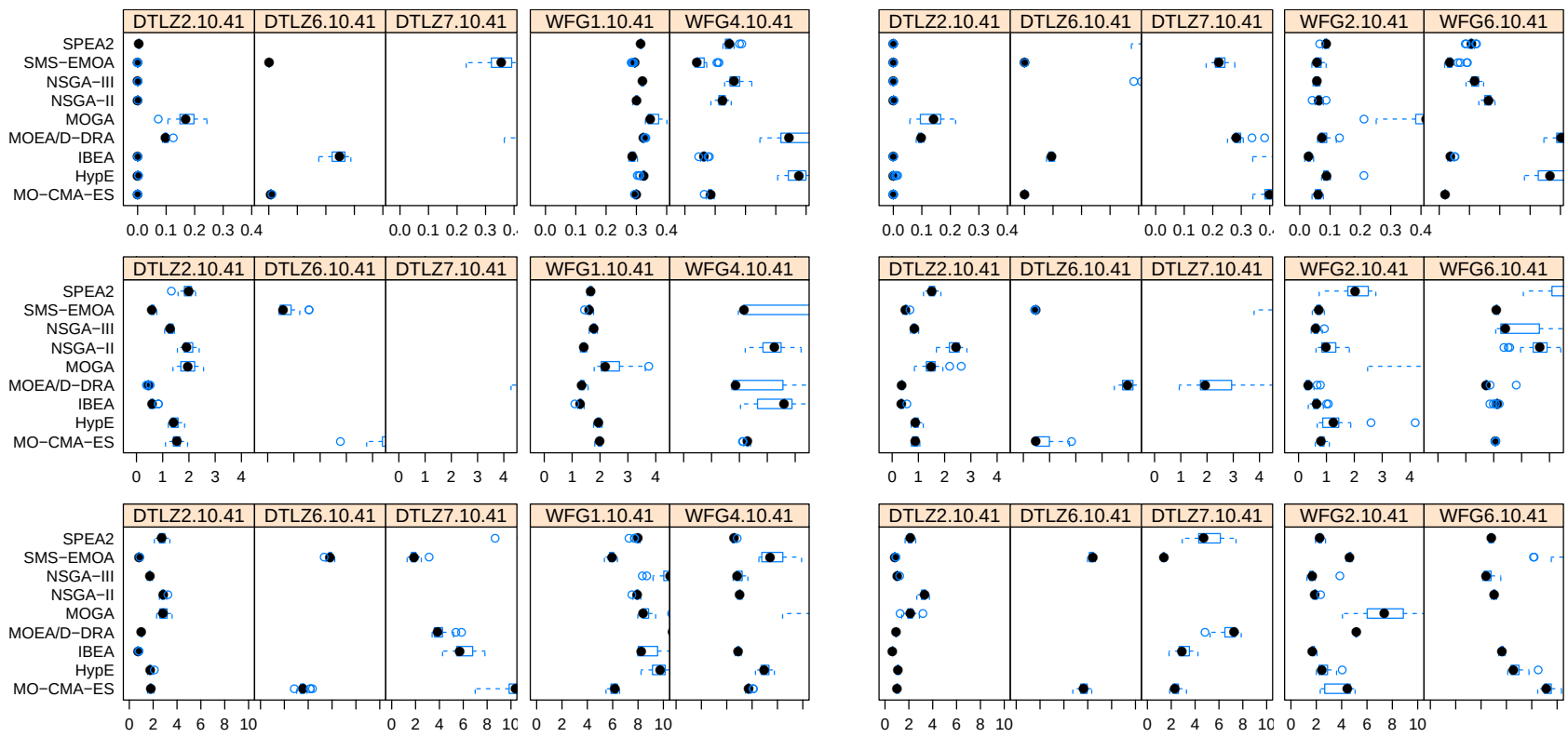


Figure B.7: Performances of MOEAs given 2 500 FEs (left) and 10 000 FEs (right) on selected ten-objective problems with 41 variables. From top to bottom, I_H^{rpd}, $I_{\epsilon+}$ and I_{IGD}.

these results.

Concerning the non-concave WFG problems, we see three very different situations. First, on WFG1 the I_H^{rpd} and the I_{IGD} agree that results are far from reasonable, but the $I_{\epsilon+}^1$ indicates otherwise. By contrast, on WFG3 the indicator that suggests a good performance from MOEAs in general is the I_{IGD}. In this case however, the three metrics disagree completely, since the I_H^{rpd} would indicate a reasonable performance and the $I_{\epsilon+}^1$ results are very poor. Finally, the only of these problems where all metrics agree is WFG2. As a result, it is pretty difficult to select the best-ranked algorithm in all these problems. As for the concave WFG problems, we notice yet again that all metrics disagree. In particular, the distance-based metrics denote a very poor performance from all MOEAs, specially the I_{IGD}. In this context, it is more clear that some algorithms are unable to simultaneously satisfy multiple metrics than to indicate the best MOEA. Overall, the major conclusion drawn from these experiments is that, when faced with computationally expensive problems that present a large number of objectives, MOEAs can only be expected to produce reasonable results for the ones that present particular features.

10 000 FEs. Results shown in Fig. B.7 (right) are very similar to the results discussed on the previous section, although improvements can be seen for most problems according to all metrics. On the moderate DTLZ problems, most MOEAs now get much closer to the actual fronts, the negative examples being the dominance-based algorithms. It is also important to remark that MOGA is surprisingly better on these functions than the remaining MOEAs from its paradigm according to the distance-based metrics. On DTLZ6, SMS and MO-CMA-ES now display good performance according to both I_H^{rpd} and $I_{\epsilon+}^1$, although mostly on smaller n_{var} values. On DTLZ7, $I_{\epsilon+}^1$ results are very poor for all MOEAs except for MOEA/D. By contrast, the only MOEA one could recommend based on the I_H^{rpd} and the I_{IGD} is SMS.

Concerning the non-concave WFG problems, results are again unique. For WFG1, the only metric that points to a reasonable performance from MOEAs in general is the $I_{\epsilon+}^1$. As for WFG2, metrics again consistently disagree, with IBEA, MOEA/D, and NSGA-III being the algorithm of choice for each metric, respectively. Nonetheless, we remark that results from half of the MOEAs in this problem are quite good, even if they are not able to fully approximate the actual front. Finally, on

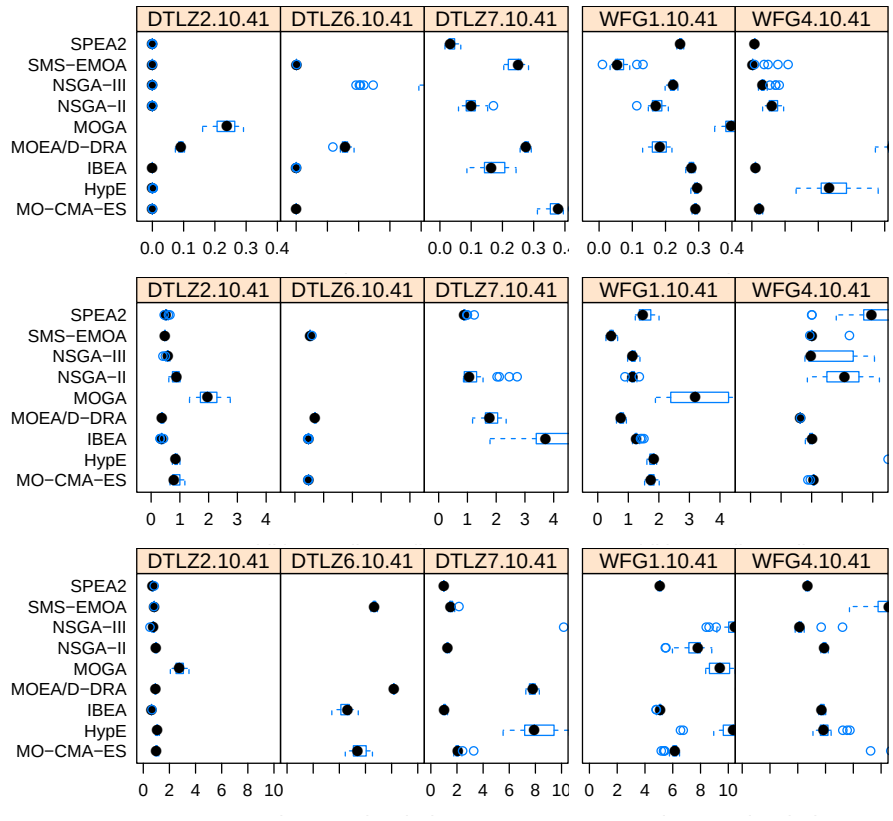


Figure B.8: Performances of MOEAs given 40 000 FEs on selected ten-objective problems with 41 variables. From top to bottom, I_H^{rpd}, $I_{\epsilon+}$ and I_{IGD}.

WFG3 results are still far from good, but can be considered much better than the ones for WFG1. Surprisingly, the MOEA of choice for this problem is NSGA-II. As for the non-concave problems, once again it is not possible to recommend a single MOEA due to the disparity between different metric rankings. Given the I_H^{rpd}, the $I_{\epsilon+}^1$, and the I_{IGD}, the algorithms of choice would be SMS, MOEA/D, and NSGA-III, respectively.

40 000 FEs. The increase in the number of FEs given to MOEAs this time results in several significant performance improvements only for the I_H^{rpd}, as seen in Fig. B.8. In particular, we remark that all MOEAs benefit from the increase FE budget on most problems. The only exception concerns the WFG1, where IBEA, MOEA/D, and MO-CMA-ES are unable to improve over their previous results. Concerning the distance-based metrics, improvements can be seen only on the moderate DTLZ functions, although the I_{IGD} also shows minor improvements for the hardest functions. We remark, though, that we observe in a few occasions that an improvement according to a given metric might result in a worsening on another. Such is the case with WFG1, for instance, where the algorithms that improve the most on the I_H^{rpd} worsen on the I_{IGD}.

Overall, results indicate that even given a large number of FEs no single MOEA is able to perform consistently well on all problems. In addition, it happens very often that the best MOEA for a given problem is only considered the best for that particular problem, or that MOEAs are only the best-performing algorithms according to a given metrics. Altogether, these results suggest that the task of a general-purpose MOEA for unconstrained continuous is probably unfeasible when dealing with truly many-objective optimization.

APPENDIX C

Component-wise multi-objective differential evolution

Differential evolution (DE) [213] plays an important role in single-objective optimization and has led to the development of a number of effective optimization algorithms for both constrained and unconstrained continuous problems [54]. In particular, one of the most attractive features of DE is its simplicity and its ability to outperform classical genetic algorithms (GAs) [193]. As a result, a number of research proposals have extended DE algorithms to tackle multi-objective optimization problems (MOPs) in the Pareto sense [54, 143, 201]. In general, extensions follow different paths on how to adapt DE to deal with Pareto optimality, and these stand-alone algorithms have been compared to well-known GA-based algorithms such as NSGA-II [60] or SPEA2 [234] to test their effectiveness. Interestingly, two research groups independently proposed the same DE algorithm at about the same time: DEMO [201] and GDE3 [143]. To highlight the effectiveness of this algorithm, we remark that it ranked among the top five best-performing algorithms at the 2009 CEC competition on multi-objective optimization [229].

In the most comprehensive study conducted so far on DE for multi-objective optimization, Tušar and Filipič [219] have considered DEMO as a template for instantiating DE algorithms. Concretely, DEMO uses DE for exploring the decision space, but uses the environmental selection strategy of NSGA-II. The authors then considered the possibility of using other environmental selection approaches, and compared three top-performing GA-based algorithms, NSGA-II, SPEA2, and IBEA [232] with DE versions of these algorithms, aliased DEMO$^{\text{NS-II}}$, DEMO$^{\text{SP2}}$ and DEMO$^{\text{IB}}$. By performing pairwise comparisons between algorithms that differ only in the underlying search mechanism (GA or DE), the DE operators were shown to obtain more accurate approximations of the Pareto front and DEMO$^{\text{SP2}}$ was found to best balance convergence and diversity [218].

We extend here this excellent earlier work by carrying out a more profound component-wise analysis [26, 27] of the design of DE algorithms for MOPs. Our analysis shows that a more fine-grained view of DE components can lead to new insights. In the original analysis only the environmental selection strategy was a component to be set in the DEMO template. However, the DE-part of DEMO differs from traditional GAs in more than one component. In addition to the *DE variation operator*, there is an *online replacement strategy*, i.e., newly generated solutions are compared to existing solutions as soon as they are created, enforcing a higher convergence pressure. In fact, the latter component was found to be the key improvement of DEMO over earlier DE adaptations to MOPs [201]. However, when we consider the DEMO versions that use environmental selection strategies from IBEA and SPEA2 instead of the

Algorithm 6 componentWiseDE template

```
1: Initialize(pop)
2: repeat
3:     Variate(pop)
4:     Reduce(pop)
5: until termination criteria met
Output: pop
```

original DEMO algorithm that uses the environmental selection from NSGA-II, we show that the online replacement strategy is not always beneficial to the effectiveness of the DEMO versions. In other words, while DEMO was an improvement over existing NSGA-II based DE algorithms because of its online replacement strategy, the other DEMO versions present the same (or, sometimes, worse) performance than versions of IBEA and SPEA2 that simply use the DE variation operator.

Furthermore, we consider several factors that affect the conclusions in the original analysis. First, in the original paper, the quality indicator used by IBEA and DEMO$^{\text{IB}}$ was the binary hypervolume difference, whereas strong evidence points to a better performance of IBEA when using the binary epsilon indicator [25, 232]. Second, the analysis conducted in the original paper was done using the default parameter settings traditionally adopted by the EMO community for the benchmarks considered. However, we have recently shown that tuning the numerical parameters of EMO algorithms can significantly improve their performance [25], altering their relative performance. Finally, although the original paper considered a representative number of benchmark functions, they all used the same number of variables. In this work, we consider several different problem sizes to ensure scalability issues do not compromise the generality of our results.

The remainder of this appendix is organized as follows. Section C.1 presents our component-wise approach to differential evolution, and how we instantiate both DE-based and GA-based algorithms using a flexible template. In addition, it also presents the intermediate algorithmic designs we use in this work to understand the contribution of the individual DE components we consider. The experimental setup used for this assessment is given in Section C.2. We split the discussion of the results in two parts. In Section C.3, we compare algorithms grouped by environmental selection strategy. In Section C.4, we compare all algorithms among themselves and to a well-known efficient EMO algorithm, SMS [19]. We do so to put the results in perspective, since we have recently shown that SMS performs consistently well for the experimental setup considered here [25]. Finally, we conclude and discuss future work in Section C.5.

C.1 Differential evolution from a component-wise view

Several articles in the literature propose how to adapt DE algorithms to multi-objective optimization. However, the differences among most of these algorithms are quite small. From a very high-level perspective, multi-objective DE algorithms can be represented using the template defined by Algorithms 6 and 7. The general template displayed in Algorithm 6 could actually represent any of the most used evolutionary computation approaches (GA, DE or evolution strategies). Starting from an initial population (line 1), variation operators and environmental selection are applied to a population to promote evolution, until a given stopping criterion is reached.

In DE algorithms, the variation procedure is carried out as displayed in Algorithm 7. The DE operator produces a trial vector from an existing target vector of the population. Although the single-objective optimization literature presents many different strategies for this operation, the multi-objective DE algorithms proposed so far use the *DE/rand/1/bin* approach [213]. The most significant difference between the existing DE proposals is encapsulated in procedure OnlineReplace (line 3). In earlier algo-

Algorithm 7 DE variation

Input: pop
1: **repeat**
2: *trial* ← DE_operator(*target*)
3: OnlineReplace(pop, *target, trial*)
4: **until** #offspring produced

Algorithm 8 GA variation

Input: pop
1: pool ← Select(pop)
2: pop_{new} ← GA_operators(pool)
3: pop ← pop ∪ pop_{new}

rithms, the trial vector $\vec{x}_{\text{trial}}$ only replaced the target vector $\vec{x}_{\text{target}}$ if $\vec{x}_{\text{trial}}$ dominated $\vec{x}_{\text{target}}$. In this case, no environmental replacement is necessary, since the population size is always constant. Later, algorithms considered the option of adding the trial vector to the population in case both trial and target vectors were nondominated. In this case, the population size might double at each iteration, and hence environmental replacement strategies are employed after the variation is concluded, to reduce the population to its original size. While this prevents algorithms from early stagnation, it may as well slow down their convergence. We refer to these two replacement versions as *online replacement strategies*, since trial solutions may replace target solutions during the variation stage, before the actual population management represented by procedure Reduce happens. However, some multi-objective DE algorithms do not consider online replacement at all. In this case, solutions are created by the DE operator, but are only compared to the population altogether, when procedure Reduce is executed. These three different options for online solution replacement are listed in the bottom part of Table C.1.

The three different DEMO versions considered by Tušar and Filipič [219] can be easily instantiated using this template as follows (all three versions use DE variation and (non)dominance online solution replacement):

DEMO$^{\text{NS-II}}$ uses environmental selection strategy proposed for NSGA-II, i.e., nondominated sorting with tie-breaking according to crowdedness.

DEMO$^{\text{SP2}}$ uses the environmental selection strategy proposed for SPEA2, i.e., sorting according to dominance strength and tie-breaking according to nearest neighbor density estimation.

DEMO$^{\text{IB}}$ uses the environmental selection strategy proposed for IBEA, i.e., sorting according to the binary ϵ-indicator ($I_{\epsilon+}$).

In an analogous fashion, the original GA-based algorithms NSGA-II, SPEA2 and IBEA can be instantiated using the same template. To do so, instead of a DE-based variation, we use a traditional GA variation approach, outlined by Algorithm 8. The mating selection (line 1) is done according to the fitness of the individuals, which is computed using the same strategies adopted for the environmental replacement in the respective GA-based algorithms. Besides the previously discussed algorithms, the component-wise template presented here could also be used to instantiate other algorithms. We will discuss this in more detail in the next section.

As it can be seen, the three original DEMO versions [219] comprise more than a single atomic DE-related algorithmic component. Concretely, it is a combination of the DE variation operator and an online replacement strategy. Although the DEMO versions of NSGA-II, SPEA2, and IBEA have indeed shown performance improvements over the original algorithms, it remains unclear how each of these individual components contribute to these performance gains. To properly assess the effectiveness of these components, we propose a set of intermediate algorithmic designs: **DE$^{\text{NS-II}}$**, **DE$^{\text{SP2}}$**, and **DE$^{\text{IB}}$** which are identical to the DEMO variants except that they do not use online solution replacement.

Moreover, the only difference between these DE versions and the original versions of NSGA-II, SPEA2 and IBEA is the use of the DE variation operator. For instance, considering the case of NSGA-II, DE$^{\text{NS-II}}$, and DEMO$^{\text{NS-II}}$, the first uses traditional GA selection and variation, while the latter two use DE variation. However, while DEMO$^{\text{NS-II}}$ may replace solutions as soon as they are created, DE$^{\text{NS-II}}$

Table C.1: Algorithmic options of a component-wise multi-objective DE template.

Component	Domain	Description
Variate	DE variation, GA variation	Underlying variation options
Reduce	NSGA-II, SPEA2, IBEA	Environmental selection approaches
OnlineReplace	dominance, (non)dominance, none	Online solution replacement criterion (this component only takes effect when DE variation is used)

replaces solutions only at the environmental selection stage (procedure Reduce of Algorithm 6). In the next section, we present the experimental setup in which we use these intermediate designs to properly investigate the effectiveness of the DE operators used by the different DEMO versions.

C.2 Experimental setup

The benchmark sets we consider here include all unconstrained DTLZ [61] and WFG [112] functions (DTLZ1–7 and WFG1–9). Since both benchmark sets offer scalability as to the number of variables and objectives, we explore this feature to increase the representativeness of our investigation. We consider versions of these problems with three and five objectives. Concerning the number of variables n, we consider problems with $n \in \{20, 21, \ldots, 60\}$. Furthermore, to ensure that numerical parameters do not affect our performance assessment of the DE components, we initially tune all algorithms, but we use disjoint sets for tuning and testing to prevent overfitting. More precisely, we use problems with sizes $n_{testing} = \{30, 40, 50\}$ for testing, and problems with sizes $n \in \{20, 21, \ldots, 60\} \setminus n_{testing}$ for tuning. For both testing and tuning, experiments are run on a single core of Intel Xeon E5410 CPUs, running at 2.33GHz with 6MB of cache size under Cluster Rocks Linux version 6.0/CentOS 6.3. The remaining details about tuning and testing are given below.

Tuning setup. The automatic parameter configuration tool we use in this work is irace [159]. Although it was originally proposed for configuring single-objective optimization algorithms, it can be adapted for multi-objective optimization by using the hypervolume indicator [157]. Concretely, for each problem considered by irace, candidate configurations are run for a maximum number of function evaluations (10 000, following [25]). The approximation fronts they produce are then normalized to the range $[1, 2]$ to prevent issues due to dissimilar domains. Finally, we compute the hypervolume for each front using $r_i = 2.1$, $i = 1, \ldots, M$ as reference point, where M is the number of objectives considered.

The parameter space we consider for tuning all algorithms is given in Table C.2. Parameter μ applies to both DE-based and GA-based algorithms. The following six parameters (λ, t_{size}, p_c, p_m, η_c, η_m) only apply to GA-based algorithms. In particular, we highlight that all GA-based algorithms use SBX crossover and polynomial mutation, as commonly done in the literature [19, 61, 112]. Parameter t_{size} controls the size of the deterministic tournament used for mating selection. The probability of applying the crossover operator to a given pair of individuals is controlled by parameter p_c. Analogously, the probability of applying the mutation operator to a given individual is controlled by parameter p_m. In addition, we consider two different mutation schemes: (i) *bitwise*,

Table C.2: Parameter space for tuning all MOEAs for continuous optimization.

		GA variation				*DE variation*	
Parameter	$\mu = \lvert\texttt{pop}\rvert$	$\lambda = \lvert\texttt{pop}_{\texttt{new}}\rvert$	t_{size}	p_c, p_m	η_c, η_m	CR	F
Domain	$\{10, 20, \ldots, 100\}$	1 or $\lambda_r \cdot \mu$ $\lambda_r \in [0.1, 2]$	$\{2, 4, 8\}$	$[0, 1]$	$\{1, 2, \ldots, 50\}$	$[0, 1]$	$[0.1, 2]$

which sets the mutation probability per variable $p_v = 1/n$; and (ii) *fixed*, where p_v becomes a parameter $\in [0.01, 1]$. Finally, η_c and η_m are the distribution indices for the SBX crossover and polynomial mutation, respectively. The remaining two parameters (CR and F) in Table C.2 concern DE variation. They control the number of variables affected by the operator (parameter CR) and the strength of the changes (parameter F).

There are two additional parameters that concern only SPEA2 and IBEA. The original version of SPEA2 contains an additional parameter k for its k-th nearest neighborhood density estimation strategy in the mating selection. Here, besides the default value, which is computed according to the population size and we denote with $k_{\text{method}} = \textit{default}$, we also give irace the possibility of configuring k directly, with $k \in \{1, 2, \ldots, 9\}$. For IBEA, as previously discussed, several different binary quality indicators can be used. Here we allow irace to select between the two most commonly adopted [232], the binary hypervolume indicator (I_H^-) and the binary ϵ-indicator ($I_{\epsilon+}$). Additionally, irace is given the flexibility to set different quality indicators for mating and for environmental selection if that leads the algorithm to better performance. Algorithms are tuned for each benchmark set (DTLZ or WFG) and for each number of objectives (3 or 5); that is, for each algorithm X, we obtain four tuned variants: X_{D3}, X_{D5}, X_{W3} and X_{W5}. For brevity, the tuned settings for all algorithms considered in this work are provided as supplementary material [29].

Testing setup. For comparing the tuned algorithms, we run each algorithm 25 times and evaluate them based on the relative hypervolume of the approximation fronts they produce w.r.t. the Pareto optimal fronts. Since the latter are typically infinite, we generate, for each problem instance, a Pareto front with 10 000 Pareto-optimal solutions following the methodology described in the papers where the benchmarks were proposed [61, 112]. Given an approximation front A generated by an algorithm when applied to a problem instance and the Pareto front P of the same problem instance, the relative hypervolume of A equals $I_H(A)/I_H(P)$. A relative hypervolume of 1.0 means the algorithm was able to perfectly approximate the Pareto front for the problem considered.

The comparison is done visually by means of boxplots, and analytically through rank sums. Since we generate a large set of results, we only discuss the most representative ones here. In particular, we focus the discussion on the WFG benchmark and provide the analysis on the DTLZ benchmark as supplementary material [29]. Additionaly, due to the large amount of results we produce, we present here the results for $n = 40$. Similar results were found for $n \in \{30, 50\}$, and are also provided as supplementary material.

C.3 Experimental analysis grouped by environmental selection

To investigate how each algorithm component individually affects the performance of the different DEMO versions, we first conduct an analysis where algorithms are grouped by the environmental selection strategy they employ.

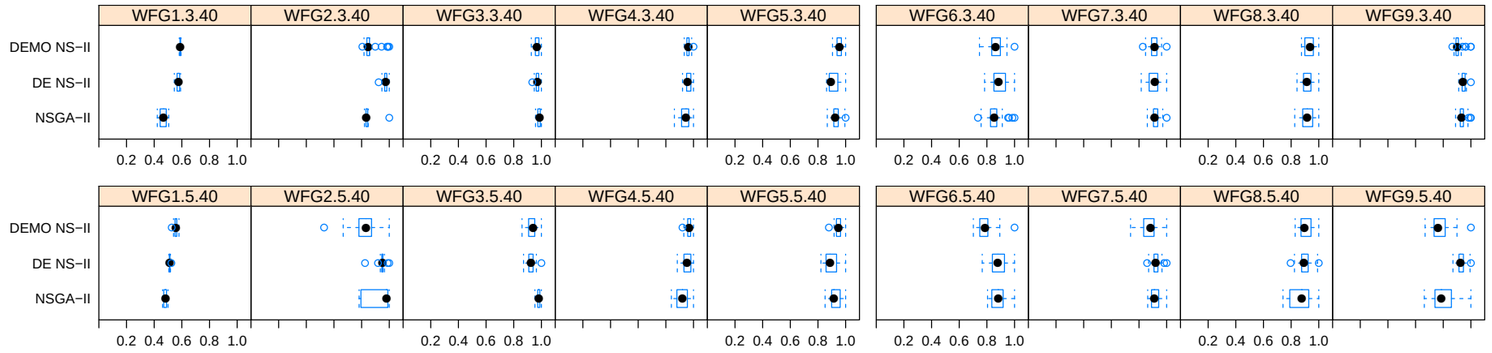


Figure C.1: $I_{H\%}$ boxplots of the MOEAs using the selection strategy of NSGA-II (WFG problems, 40 variables). Top: 3-obj; bottom: 5-obj.

Figure C.2: $I_{H\%}$ boxplots of the MOEAs using the selection strategy of SPEA2 (WFG problems, 40 variables). Top: 3-obj; bottom: 5-obj.

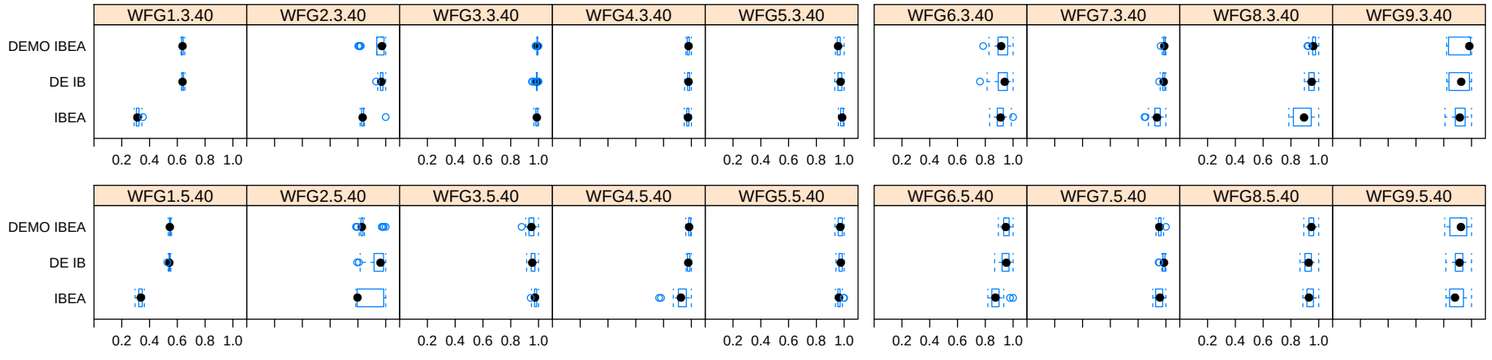


Figure C.3: $I_{H\%}$ boxplots of the MOEAs using the selection strategy of IBEA (WFG problems, 40 variables). Top: 3-obj; bottom: 5-obj.

Table C.3: Sum of ranks depicting the overall performance of algorithms grouped by environmental selection strategy. Algorithms in boldface present rank sums not significantly higher than the lowest ranked for a significance level of 95%.

3 objectives			5 objectives		
DEMO$^{\text{NS-II}}_{\text{W3}}$ (1259.5)	**DE$^{\text{NS-II}}_{\text{W3}}$ (1321)**	NSGA-II$_{\text{W3}}$ (1469.5)	**DE$^{\text{NS-II}}_{\text{W5}}$ (1257)**	DEMO$^{\text{NS-II}}_{\text{W5}}$ (1393)	NSGA-II$_{\text{W5}}$ (1400)
DEMO$^{\text{SP2}}_{\text{W3}}$ (1281)	**SPEA2$_{\text{W3}}$ (1299.5)**	DE$^{\text{SP2}}_{\text{W3}}$ (1469.5)	**DEMO$^{\text{SP2}}_{\text{W5}}$ (1259)**	**DE$^{\text{SP2}}_{\text{W5}}$ (1346.5)**	SPEA2$_{\text{W5}}$ (1444.5)
DE$^{\text{IB}}_{\text{W3}}$ (1212)	**DEMO$^{\text{IB}}_{\text{W3}}$ (1246.5)**	IBEA$_{\text{W3}}$ (1591.5)	**DEMO$^{\text{IB}}_{\text{W5}}$ (1215.5)**	**DE$^{\text{IB}}_{\text{W5}}$ (1225.5)**	IBEA$_{\text{W5}}$ (1609)

NSGA-II, DE$^{\text{NS-II}}$, and DEMO$^{\text{NS-II}}$: The boxplots of the relative hypervolume achieved by the algorithms that use the environmental selection strategy proposed for NSGA-II are given in Figure C.1. For the 3-objective problems (top), we observe very heterogeneous results. For some problems such as WFG7 and WFG8 there is almost no difference between the algorithms, indicating that the DE components are unable to improve the performance of the original NSGA-II. However, for problems such as WFG1, WFG2, WFG4, and WFG6, the performance of NSGA-II can be improved, sometimes by a large margin, such as for WFG1 and WFG2. When we consider the effectiveness of the DE components, we see that sometimes using both components (as in DEMO$^{\text{NS-II}}$) is beneficial (e.g., WFG1, WFG5, and WF8), but for other problems it is better to use the DE variation without the online replacement strategy as in DE$^{\text{NS-II}}$ (e.g., WFG2, WFG6, and WFG9). Particularly for WFG9, using both components simultaneously worsens the performance of NSGA-II. When we aggregate results for all runs and sizes of 3-objective WFG problems in a rank sum analysis (Table C.3), we see that both DE-based algorithms improve over NSGA-II, but no significant difference can be found among DEMO$^{\text{NS-II}}$ and DE$^{\text{NS-II}}$ using Friedman's test at 95% confidence level.

The performance shown by NSGA-II, DE$^{\text{NS-II}}$, and DEMO$^{\text{NS-II}}$ on the 5-objective WFG problems (see Fig. C.1, bottom) is quite different. This time, using both DE components (DEMO$^{\text{NS-II}}$) is only beneficial for problems WFG1, WFG4, WFG5, and WFG8. In the other problems, the online replacement leads to results worse even than the ones achieved by the original NSGA-II. However, when we consider only the DE variation (DE$^{\text{NS-II}}$), we see that the performance of NSGA-II is improved for most functions, except for WFG2 and WFG5. When we aggregate results for all 5-objective problems, we see that DE$^{\text{NS-II}}$ indeed ranks first, with significantly lower rank sums than the remaining algorithms (Table C.3).

SPEA2, DE$^{\text{SP2}}$, and DEMO$^{\text{SP2}}$: The boxplots of the relative hypervolume achieved by the algorithms that use the environmental selection strategy proposed for SPEA2 are given in Figure C.2. This time the 3-objective problems (top) show a more clear separation between problems for which DE components lead to improvements and problems for which they worsen the performance of the original SPEA2. For the first group (WFG1, WFG2, and WFG6), we see that there is no pattern as to whether the online replacement is a suitable component for improving SPEA2. However, for the problems where DE components do not lead to performance improvements, typically the version that uses online replacement (that is, DEMO$^{\text{SP2}}$) shows better results than the version that does not use it (that is, DE$^{\text{SP2}}$). When we aggregate results for all 3-objective problems, we see that SPEA2 and DEMO$^{\text{SP2}}$ show equivalent results, while DE$^{\text{SP2}}$ shows significantly higher rank sums than both.

For the 5-objective WFG problems (see Figure C.2, bottom), the online replacement component plays a more important role than in the 3-objective problems. For most problems, the performance

of DE^{SP2} and $DEMO^{SP2}$ is quite different: while $DEMO^{SP2}$ outperforms SPEA2 for most problems, DE^{SP2} worsens the performance of SPEA2 for nearly half of the problems considered. The main exception is WFG2, where DE^{SP2} has the best performance among all algorithms. When all 5-objective problems are considered (Table C.3), $DEMO^{SP2}$ ranks first with rank sums significantly lower than DE^{SP2} and SPEA2, which respectively rank second and third. Despite its erratic behavior, DE^{SP2} also presents significantly lower rank sums than SPEA2.

IBEA, DE^{IB}, and $DEMO^{IB}$: The boxplots of the relative hypervolume achieved by the algorithms that use the environmental selection strategy proposed for IBEA are given in Figure C.3. The results for the 3-objective problems (top) achieved by these indicator-based versions are far more homogeneous than the results shown before for NSGA-II and SPEA2 environmental selection strategies. In almost all situations, DE^{IB} and $DEMO^{IB}$ perform nearly identically. Moreover, the DE-based variants always outperform the GA-based version, except for problems WFG3–WFG5, where the original IBEA was already very effective. These results indicate that, for 3-objective problems, the online replacement component is not an effective component when combined with the indicator-based environmental selection strategy proposed by IBEA.

The results for the 5-objective problems (see Figure C.3, bottom) are somehow consistent with the results on the 3-objective problems. However, on the 5-objective problems, online replacement leads to performance changes. For some problems, such as WFG2 and WFG7, DE^{IB} finds better results than $DEMO^{IB}$. The opposite happens for problems WFG8 and WFG9. When we aggregate across all problems (Table C.3), we see that these two algorithms get nearly the same rank sum, and that IBEA gets significantly worse rank sums.

Overall, the DE operator leads algorithms to better results on problems WFG1, WFG2, WFG6, and WFG9. As common characteristics, WFG1 and WFG2 present convex geometry, WFG1 and WFG9 present some form of bias, and WFG6 and WFG9 present a complex non-separable reduction [112]. As for the online replacement component, the only problem for which we can say that it is beneficial is the WFG8 problem. However, since the DE operator typically worsens the performance of the original algorithms for this problem, we see that the online replacement is only weakening the effects of the DE operator. Although these results might seem to contradict the results presented by the authors of DEMO, we see that the environmental selection strategy from NSGA-II represents a special case here. $DEMO^{NS\text{-}II}$ in fact improves over $DE^{NS\text{-}II}$ and NSGA-II, particularly for functions where NSGA-II faces difficulties [112]. However, this is most likely explained by the poor performance of NSGA-II rather than by the effectiveness of the online replacement strategy.

C.4 Comparison to SMS

In this section we compare all algorithms with SMS. For the3-objective problems (Figure C.4, top) we see that, in general, the DE-based algorithms are never clearly worse than SMS, except for the WFG6 problem. Particularly for WFG1 and WFG2, the differential evolution operator leads to a significant performance improvement. However, the online replacement is not effective for these two problems regardless of the environmental selection strategy employed, and often worsens the performance of the algorithms. When we aggregate across all 3-objective problems considered (Table C.4), we see that DE^{IB} and $DEMO^{IB}$ achieve significantly lower rank sums than all other algorithms. $DEMO^{SP2}$ and SPEA2 rank second, along with SMS. These results confirm that DE algorithmic components can indeed lead to significant performance improvements, but that the interactions between them and the environmental selection are also significant.

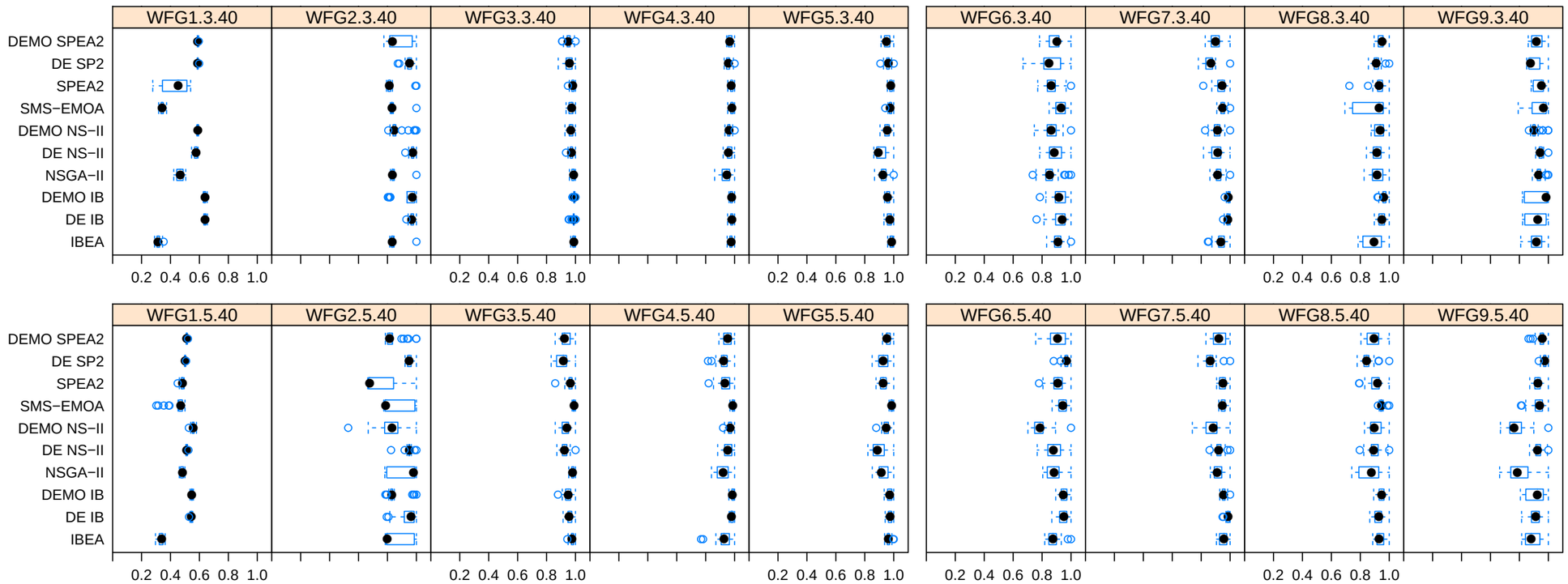


Figure C.4: Achieved $I_{H\%}$ boxplots: 3-objective (top) and 5-objective (bottom) WFG problems with 40 variables.

Table C.4: Sum of ranks depicting the overall performance of all algorithms. ΔR is the critical rank sum difference for Friedman's test with 95% confidence. Algorithms in boldface present rank sums not significantly higher than the lowest ranked.

3 objectives ($\Delta R = 271$)	5 objectives ($\Delta R = 265$)
DE^{IB}_{W3} (2532)	**DE^{IB}_{W5} (2493)**
$DEMO^{IB}_{W3}$ (2535)	**$DEMO^{IB}_{W5}$ (2506)**
$DEMO^{SP2}_{W3}$ (3738.5)	SMS_{W5} (2891.5)
$SPEA2_{W3}$ (3764.5)	$IBEA_{W5}$ (3930)
SMS_{W3} (3798)	$DEMO^{SP2}_{W3}$ (3932.5)
$IBEA_{W3}$ (3924)	$DE^{NS\text{-}II}_{W5}$ (4089)
$DEMO^{NS\text{-}II}_{W3}$ (3972.5)	DE^{SP2}_{W5} (4123.5)
$DE^{NS\text{-}II}_{W3}$ (4094.5)	$SPEA2_{W5}$ (4271.5)
DE^{SP2}_{W3} (4325.5)	$DEMO^{NS\text{-}II}_{W5}$ (4426.5)
$NSGA\text{-}II_{W3}$ (4440.5)	$NSGA\text{-}II_{W5}$ (4461)

The comparison between all algorithms for 5-objective problems is given in Figure C.4 (bottom). This time the environmental selection strategy becomes very important for the effectiveness of the algorithms. As expected, dominance-based approaches (NSGA-II and SPEA2) are not as effective for many-objective scenarios, and hence even the DE versions of these algorithms are not able to perform as well as the indicator-based algorithms. However, the performance improvements provided by the DE variation to IBEA is such that both DE^{IB} and $DEMO^{IB}$ become the top-performing algorithms, even though IBEA itself did not perform as competitively as SMS. These results indicate that, if coupled with proper many-objective search mechanisms, DE algorithmic components can possibly improve state-of-the-art algorithms, such as SMS.

C.5 Conclusions

This appendix has examined how the individual components of DE interact with the components of various EMO algorithms. In particular, we studied the underlying variation operator (GA or DE), the environmental selection strategy (NSGA-II, SPEA2, or IBEA), and the use of an online replacement strategy. For the DTLZ benchmark, results presented a ceiling effect, and hence we focused our analysis on the WFG benchmark. For both three or five objectives, results showed that the DE-operator improves the algorithms in most problems and that there is a strong interaction between this component and environmental selection. However, for the online replacement component, results almost always indicated that this component is not effective, except when combined with NSGA-II environmental selection.

These results represent a significant contribution of our investigation. Before our work, it was believed that the online replacement component was critical to the effectiveness of multi-objective DE algorithms [201]. Furthermore, this result reinforces the value of the component-wise design approach, particularly the argument that components should be jointly investigated to account for interactions.

APPENDIX D

Automatic MOACO design for the bi-objective knapsack

Multi-objective ant colony optimization (MOACO) algorithms have been applied to multi-objective combinatorial optimization problems (MCOPs) since more than 10 years [5, 15, 63, 85, 121, 156]. The interest in MOACO algorithms may be explained by the practical relevance of multi-objective problems and by the positive results that have been achieved with these algorithms. The available MOACO algorithms provide a large number of different design choices that allow the instantiation of a huge number of structurally different MOACO algorithms. Recently, López-Ibáñez and Stützle [157] proposed a MOACO framework that implements most of those design possibilities. The automatic configuration tool *Iterated F-race* (irace) [14, 159] was used to automatically generate MOACO algorithms for the bi-objective traveling salesman problem (bTSP). The authors showed that the automatic configuration of a generic MOACO framework produced better results than the MOACO algorithms from the literature used to build the framework. In this appendix, we continue the investigation of the effectiveness of this approach by extending the MOACO framework to deal with the bi-objective bidimensional knapsack problem (bBKP).

The bBKP is a popular benchmark problem in multi-objective optimization [163, 233]. Moreover, four different MOACO algorithms have been proposed for the bBKP [5]. The bBKP has also some properties that make it interesting for further exploring the possibilities of the automatic design of MOACO algorithms from a flexible framework. In particular, the representation of solutions is different from the TSP, pheromone information is represented by a vector instead of a matrix, and the structure of the solution space is quite different from the TSP. This chapter shows that the proposed method for the automatic design of MOACO algorithms also works for the bBKP. The proposed method is able to generate, with little effort from the human designer, MOACO algorithms that are clearly better than those proposed earlier for the bBKP, even after tuning the ACO settings of the MOACO algorithms from the literature and improving significantly their performance.

The remainder of this chapter is organized as follows. Section D.1 reviews the basic ACO concepts and in particular the underlying ACO algorithms used by the framework. Section D.2 reviews the bBKP and the existing MOACO algorithms for this problem. Next, Section D.3 describes the MOACO framework, highlighting the extensions we implement for its application to the bBKP. Section D.4 describes the experimental setup, while experimental results are discussed in Section D.5. We conclude in Section D.6.

D.1 Ant colony optimization

Ant colony optimization (ACO) comprises a set of ant algorithms proposed for several different optimization domains, such as combinatorial, continuous, dynamic and constrained. In common, all ACO algorithms employ agents, namely *artificial ants*, that construct solutions at each iteration. The natural inspiration of ACO algorithms lies in the informed constructive procedure they use. In nature, foraging ants deposit a chemical substance called pheromone over the path they traverse. This pheromone influences the movement direction of other ants: the greater the presence of pheromone, the greater the chance an ant will follow that path. In optimization, besides pheromone information, ants also evaluate heuristic information before making a decision. For the TSP, for instance, the probability $p(e_{ij})$ of selecting an edge e_{ij} of the construction graph is given by:

$$p(e_{ij}) = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{h \in \mathcal{N}_i} \tau_{ih}^{\alpha} \cdot \eta_{ih}^{\beta}} & \text{if } j \in \mathcal{N}_i \\ \\ 0 & \text{otherwise} \end{cases} \tag{D.1}$$

where α and β are numerical parameters that respectively control the influence of the pheromone (τ_{ij}) and the heuristic information (η_{ij}), and $\mathcal{N}_i$ is the set of states reachable from state i. Moreover, the pheromone information in ACO algorithms is also subject to evaporation and reinforcement. Since algorithms differ on how to operationalize these characteristics, we detail them separately. Among the ant algorithms devised for combinatorial optimization, three major proposals form a representative sample of this metaheuristic, namely Ant System (AS) [66], Ant Colony System (ACS) [64] and $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [215]. As all discrete ACO algorithms, Ant System was originally proposed for the traveling salesman problem (TSP), but the general idea of the algorithm is applicable to any combinatorial problem. In AS, pheromone is only reinforced after all solutions have finished building their solutions. More precisely, all ants perform a pheromone deposit, the value of which is calculated as:

$$\Delta\tau_{ij}^{k} = \frac{1}{L_k} \tag{D.2}$$

where L_k stands for the length of the tour built by ant k[1]. Given a total of m ants, the *global pheromone update* used at iteration t is defined as

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t, t+1) \tag{D.3}$$

where ρ stands for the evaporation rate.

The original AS algorithm initially presented high-quality results for the TSP, but further experimental analysis on larger instances showed it reached very poor results compared to the state-of-the-art then [215]. Several improvements have since been proposed. The other two major ACO algorithms, ACS and $\mathcal{MM}$AS, differ from AS regarding the number of ants allowed to contribute to the global pheromone update. Two strategies are commonly adopted: the ant that constructed the best solution of the iteration (*iteration-best*) or the ant that constructed the best solution so far (*best-so-far*). Moreover, both algorithms present unique distinctive features. In $\mathcal{MM}$AS, pheromone limits are subject to bounds, τ_{max} and τ_{min}, and are initialized to a very high number. To compute these bounds, $\mathcal{MM}$AS first determines the value of τ_{max}. In their original proposal, Stützle and Hoos [215] demonstrate that the maximum pheromone amount any edge can have, τ_{max}, is given by Equation D.4, where L^{opt} stands for the tour length of the optimal solution. Since this information is generally not known, $\mathcal{MM}$AS uses an estimate, $\hat{\tau}_{max}$, given by Equation D.4, where L^{bsf} is the tour length of the best solution constructed so far. The

[1] Notice that only transitions used by the ants are actually reinforced.

lower bound τ_{min} is then computed according to Equation D.4, where ν is a numerical parameter of the algorithm.

$$\tau_{max} = \frac{1}{\rho \cdot L^{opt}} \qquad \hat{\tau}_{max} = \frac{1}{\rho \cdot L^{bsf}} \qquad \tau_{min} = \frac{\hat{\tau}_{max}}{\nu} \tag{D.4}$$

The use of pheromone bounds in $\mathcal{MM}$AS may lead the algorithm to a *convergence* situation. An $\mathcal{MM}$AS algorithm is said to have converged if, for each state of the construction graph, one neighbor state presents a pheromone amount equal to $\hat{\tau}_{max}$, while the information of all the others equals τ_{min}. At this stage, the pheromone bounds are recomputed and the pheromone information of all edges of the construction graph is reinitialized to the new $\hat{\tau}_{max}$.

The difference between ACS and AS is more subtle. Besides the global pheromone update used by AS, ACS algorithms also use a *local pheromone update*, meant to prevent stagnation. After finishing building its solution, each ant updates the pheromone information of the transitions it has traversed, using the formula given by Equation D.5. Furthermore, a different state transition rule is used, allowing the algorithm to behave greedily. At every step of the constructive procedure, the algorithm randomly generates a number q and compares it to a numerical control parameter q_0. If $q \leq q_0$, the best transition is greedily taken. Else, Equation D.1 is used as in the original AS.

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0 \tag{D.5}$$

D.2 ACO algorithms for the bBKP

In this chapter, we tackle the bBKP, which is a widely used bi-objective benchmark problem [163, 233]. The bBKP is a special case of the general multi-objective multidimensional knapsack problem (moMKP), which is formalized as follows:

$$\max \ f^d(x) = \sum_{i=1}^{n} p_i^d x_i \quad d = 1, \ldots, D \qquad \text{s.t.} \quad \sum_{i=1}^{n} w_i^j x_i \leq W_j \quad j = 1, \ldots, m$$

where each item i has D profits and m costs, f^d is the d-th component of the D-dimensional objective vector f, n is the number of items, p_i^d is the d-th profit of item i, w_i^j is the j-th cost of item i, W_j is the j-th capacity of the knapsack, and x_i is a decision variable in $\{0, 1\}$ that controls whether item i is included in the knapsack ($x_i = 1$) or not ($x_i = 0$). The set of feasible solutions is $X \subseteq \{0, 1\}^n$. The bBKP is a special case of the moMKP where $D = m = 2$.

The consequence of the constructive aspect of ACO algorithms is that in order to apply an ant algorithm to a given COP, the problem must first be modeled as a construction graph. For the TSP this modeling is straightforward, since the original graph with cities and distances is already a compatible construction graph. For other problems, this modeling has to be carefully conducted, since it is critical for the performance of the algorithm. In order to apply ACO algorithms to the single-objective BKP, the approach adopted here starts by considering that all items are initially out of the knapsack. At each step, probabilities are computed and one item is chosen to be added to the knapsack. The knapsack capacity is updated, and items which eventually do not fit the knapsack anymore are removed from the candidate list. The solution construction ends when no more items can be added to the knapsack. Furthermore, the pheromone information is defined as a vector, where each component τ_i gives the desirability of adding item i to the knapsack. Each ant k constructs a solution by adding, at each step, item i to the knapsack

with a probability p_i

$$p_i = \begin{cases} \frac{\tau_i^\alpha \cdot \eta_i^\beta}{\sum_{j \in N^k} \tau_j^\alpha \cdot \eta_j^\beta} & \forall i \in N^k, \\ 0 & \text{otherwise}, \end{cases} \tag{D.6}$$

where η_i is a heuristic estimation of the benefit of adding item i, and N^k is a set of candidate items. After each step, the item added to the current solution and those items that do not fit anymore in the remaining capacity of the knapsack are removed from the candidate set. The solution construction stops when the candidate set is empty. After the constructed solutions are evaluated, the pheromone information is updated in two steps. First, pheromone values are evaporated, that is, decreased by a factor ρ. A stronger evaporation makes the ACO algorithm converge faster but may lead to stagnation. Second, the pheromone values corresponding to items present in the best solutions are updated by depositing an amount of pheromone $\Delta\tau$, thus increasing the probability that newly constructed solutions contain those items. Alaya et al. [5] proposed four different algorithms that extend the ACO metaheuristic to the bBKP.

mACO$_1$ has one pheromone vector for each objective, that is, τ^1 and τ^2. Ants are divided in three groups $\lambda \in \{0, 0.5, 1\}$ according to the weight λ they use for aggregating the two pheromone vectors when constructing solutions. The solution construction uses *random aggregation*, that is, at each step the pheromone information to be used is chosen as τ^1 with a probability $(1 - \lambda)$, and as τ^2, otherwise. This means that ants using $\lambda = 0$ or $\lambda = 1$ use only τ^1 or τ^2, respectively. The heuristic information is aggregated by means of *weighted sum aggregation*, that is, $\eta = (1 - \lambda) \cdot \eta^1 + \lambda \cdot \eta^2$, where η^1 and η^2 are the heuristic information corresponding to each objective.

The pheromone update method used by mACO$_1$ is a particular case for $\lambda \in \{0, 0.5, 1\}$ of a method called *best-of-objective-per-weight* (BOW) [157]. In BOW, those solutions generated with the same weight λ are kept in the same list. For the lists of $\lambda \notin \{0, 1\}$, the best solution according to each objective updates the pheromone vector of the corresponding objective. For the list of $\lambda = 0$, only the best solution according to the first objective updates τ^1, whereas for the list of $\lambda = 1$, only the best solution according to the second objective updates τ^2.

Finally, mACO$_1$ uses a particular pheromone deposit. Given the best solution constructed in the current iteration and the best-so-far solution according to objective d (s^d_{ib} and s^d_{bf}, respectively), the amount of pheromone deposited is given by $\Delta\tau^d = \frac{1}{1+f^d(s^d_{\text{bf}})-f^d(s^d_{\text{ib}})}$. We refer to this method as *fobj-mACO*.

mACO$_2$ is identical to mACO$_1$ except for how the multiple pheromone vectors are aggregated. Instead of a random aggregation, mACO$_2$ uses a weighted sum aggregation, that is, $\tau = (1 - \lambda) \cdot \tau^1 + \lambda \cdot \tau^2$.

mACO$_3$ uses only a single pheromone vector. The heuristic information is also a single vector, which is statically computed at the start of the algorithm as $\eta_i = \eta_i^1 + \eta_i^2$. Pheromone information is updated using all nondominated solutions found since the start of the algorithm, that is, the best-so-far archive. Every solution component is rewarded a constant $\Delta\tau = 1$ only once per iteration, regardless of how many times it is present on different solutions.

mACO$_4$ follows mACO$_1$: one pheromone vector per objective, which are aggregated by weighted random aggregation; BOW pheromone update, and pheromone deposit is *fobj-mACO*. However, there is only one weight $\lambda = 0.5$, and one heuristic vector defined as in mACO$_3$.

The mACO algorithms can be instantiated as described above by our MOACO framework [157]. which defines some terms differently from Alaya et al. [5]. For example, following Iredi et al. [121], we define a colony as a group of ants that construct solutions using only their own pheromone information. According to this definition, the four mACO variants are single colony, since the ants share the pheromone information. Nonetheless, the above formulation is equivalent to the original algorithms, and we have confirmed this approach is equivalent to the original [5].

Algorithm 9 MOACO framework

1: **for** each colony $c \in \{1, \ldots, N^{\text{col}}\}$ **do**
2: InitializePheromoneInformation()
3: $\Lambda_c \leftarrow$ MultiColonyWeights()
4: InitializeHeuristicInformation()
5: $A^{\text{bf}} \leftarrow \emptyset$
6: $iter \leftarrow 0$
7: **while** not termination criteria met **do**
8: $A^{iter} \leftarrow \emptyset$
9: **for** each colony $c \in \{1, \ldots, N^{\text{col}}\}$ **do**
10: **for** each ant $k \in \{1, \ldots, N^{\text{a}}\}$ **do**
11: $\lambda \leftarrow$ NextWeight(Λ_c, k, $iter$)
12: $\tau \leftarrow \begin{cases} \text{Aggregation}(\lambda, \{\tau_c^1, \tau_c^2\}) & \text{if multiple } [\tau] \\ \tau_c & \text{if single } [\tau] \end{cases}$
13: $\eta \leftarrow \begin{cases} \text{Aggregation}(\lambda, \{\eta^1, \eta^2\}) & \text{if multiple } [\eta] \\ \eta & \text{if single } [\eta] \end{cases}$
14: $s \leftarrow$ ConstructSolution(τ, η)
15: $A^{iter} \leftarrow$ RemoveDominated($A^{iter} \cup \{s\}$)
16: $A^{\text{bf}} \leftarrow$ RemoveDominated($A^{\text{bf}} \cup A^{iter}$)
17: $A^{\text{upd}} \leftarrow$ ChooseUpdateSet(A^{iter}, A^{bf})
18: **for** each colony $c \in \{1, \ldots, N^{\text{col}}\}$ **do**
19: $A_c^{\text{upd}} \leftarrow$ MultiColonyUpdate(A^{upd})
20: PheromoneUpdate($A_c^{\text{upd}}, N^{\text{upd}}$)
21: $iter \leftarrow iter + 1$
22: **Output:** A^{bf}

D.3 A flexible MOACO framework for the bBKP

In this chapter, we extend the flexible MOACO framework proposed for the bTSP by López-Ibáñez and Stützle [157] to also tackle the bBKP and we automatically instantiate MOACO algorithms. The MOACO framework is able to replicate most MOACO designs proposed in the literature and can generate new MOACO designs by combining components in novel ways. However, its application to the bBKP requires extending it concerning the solution representation and other problem-specific features. Here, we briefly summarize the high-level structure of the framework and its components (see [157] for further details).

The high-level algorithmic scheme of the MOACO framework is given in Algorithm 9. The MOACO framework is a multi-colony algorithm, where each colony c of ants has its own pheromone information and its own set of weights Λ_c for possibly aggregating information. The assignment of weights to colonies is defined by MOACO component MultiColonyWeights. Within each colony, each ant constructs a solution according to pheromone information τ and heuristic information η. Either τ or η may be the result of aggregation. That is, if the pheromone information consists of multiple pheromone vectors, one for each objective, these are aggregated into a single pheromone vector τ by means of MOACO component Aggregation (line 12), using a particular weight λ. If multiple heuristic vectors are used, they are aggregated in a similar way. Which weight is used by each ant may depend on the set of weights of each colony, the particular ant, and the particular iteration. The different possibilities are encapsulated by MOACO component NextWeight (line 11). Once all ants have constructed a solution, the resulting iteration-best archive of nondominated solutions (A^{iter}) is merged into the best-so-far archive (A^{bf}) (line 16). After this step, the pheromone information of each colony is updated in two steps. First, the set of solutions for update (either A^{iter} or A^{bf}), is partitioned among colonies according to component MultiColonyUpdate (line 19). Next, a number of solutions from each set is used to update the pheromone information of each colony in a way defined by component PheromoneUpdate (line 20). The algorithm

Table D.1: Algorithmic components of the MOACO framework

Component	Domain	Description
$[\tau]$	{ single, multiple }	Num. pheromone vectors
$[\eta]$	{ single, multiple }	Num. heuristic vectors
N^{weights}	$\mathbb{N}^+$	Number of weights
Aggregation	weighted sum, weighted product, random	How weights are used to aggregate multiple $[\tau]$ or $[\eta]$
NextWeight	one weight per iteration (1wpi), all weights per iteration (awpi)	How weights are used at each iteration
PheromoneUpdate	nondominated solutions (ND), best-of-objective (BO), best-of-objective-per-weight (BOW)	Which solutions are selected to update the pheromone information
N^{upd}	$\mathbb{N}^+$	Num. solutions that update each $[\tau]$
ChooseUpdateSet	best-so-far (BSF), iteration-best (IB), mixed	Whether the solutions used for update are taken from A^{bf}, A^{iter} or using both alternately
The following components have an effect only when using multiple colonies.		
N^{col}	$\mathbb{N}^+$	Number of colonies
MultiColonyWeights	same ($\cap_{100\%}$), overlapping ($\cap_{50\%}$), disjoint ($\cap_{0\%}$	Whether colonies share all, 50% or no weights.
MultiColonyUpdate	{ origin, region }	How solutions are assigned to colonies
New components added in this work for the bBKP.		
$\tau_{\max}$ method	{ default, value }	Method for calculating $\tau_{\max}$
$\tau_{\min}$ method	{ default, value }	Method for calculating $\tau_{\min}$
$\Delta\tau$	{ constant, fobj-mACO, fobj, MACS }	Method for calculating $\Delta\tau$
η_i	$\{ \frac{\text{profit}}{\text{cost}}, \frac{\sum \text{profits}}{\text{cost}}, \frac{\text{profit}}{\sum \text{costs}} \}$	Heuristic information used

stops when a termination criterion is met, typically a maximum number of iterations or a time limit, and returns the best-so-far archive.

The flexibility of the MOACO framework is given by the alternative definitions of the algorithmic components that specify the key steps in the algorithm. Defining these components in particular ways allows the framework to replicate most of the MOACO algorithms in the literature. A summary of the available alternatives is given in Table D.1. The complete description of all components and their alternatives can be found in the original publication [157]. For brevity, we restrict ourselves here to the new extensions implemented for the bBKP.

Following [157], we use $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [215] as the underlying ACO algorithm that defines details such as the pheromone deposit $\Delta\tau$, and maximum and minimum pheromone levels ($\tau_{\max}$ and $\tau_{\min}$). Here, we have adapted $\mathcal{MM}$AS to the bBKP, but making more flexible the definition of $\Delta\tau$, $\tau_{\max}$ and $\tau_{\min}$ to be able to replicate faithfully the original mACO algorithms for the bBKP. The alternatives implemented for the definition of the pheromone deposit ($\Delta\tau$) are:

fobj, that is, $\Delta\tau^d = f^d(s)$, where τ^d is the pheromone information corresponding to objective d. If only one pheromone vector is used instead of multiple, then $\Delta\tau = f^1(s) + f^2(s)$. This method is the

one used in the original $\mathcal{MM}$AS.

constant, that is, $\Delta\tau^d = 1 - \frac{r^d(s)-1}{N^{\text{upd}}}$, where $r^d(s)$ is the rank of solution s ordered according to objective d and N^{upd} is the number of solutions used to update τ^d. This method is inspired by rank-based ant system [44].

fobj-mACO, this is the method used in mACO_1, mACO_2 and mACO_4.

MACS, that is, $\Delta\tau = f^1(s) \cdot f^2(s)$, which is adapted from MACS [15].

For the definition of the pheromone levels we consider two possibilities. The first is the **default** setting of $\mathcal{MM}$AS, which uses $\tau_{\max} = \frac{\max^{iter}(\Delta\tau)}{\rho}$, where $\max^{iter}(\Delta\tau)$ is the maximum amount of pheromone deposited at iteration *iter* for a single pheromone component, and $\tau_{\min} = \frac{\tau_{\max}}{\nu \cdot n}$, where $\nu \in \mathbb{R}^+$ is a parameter ($\nu = 2$ in $\mathcal{MM}$AS). The second is the **value** setting, where $\tau_{\max}$ and $\tau_{\min}$ are set to two different constant values $\tau_{\max} > \tau_{\min}$. A **value** setting is used in all mACO algorithms.

In addition, we have implemented three alternatives for the heuristic information. For a given objective d and item i, the heuristic information can be either profit divided by cost ($\eta 1_i^d$), which is the one used in the mACO algorithms [5], sum profits divided by cost ($\eta 2_i^d$), or profit divided by sum costs ($\eta 3_i^d$) [163], that is,

$$\eta 1_i^d = \frac{p_i^d}{w_i^d} \qquad \eta 2_i^d = \frac{\sum_{k=1}^{D} p_i^k}{w_i^d} \qquad \eta 3_i^d = \frac{p_i^d}{\sum_{l=1}^{m} w_i^l} \tag{D.7}$$

D.4 Experimental setup

Our experiments are divided in two stages. In a first stage, we automatically configure the ACO settings of the mACO algorithms and compare the resulting configurations with the original settings. This is done to avoid a bias by possibly poor ACO parameter settings of the mACO algorithms. In the second stage, we compare the best configurations with an algorithm automatically instantiated from the MOACO framework.

As the automatic algorithm configuration tool, we use irace [14, 159]. The input of irace is a definition of the parameter space, which may contain categorical and numerical parameters, and a set of training instances. irace was originally designed for single-objective algorithms, but it has been extended to handle the multi-objective case by using the hypervolume quality measure [157] (I_H). The hypervolume is a well-known quality measure in multi-objective optimization [235]. It computes for each approximation set, the volume in the objective space weakly dominated by the approximation set and bounded by a reference point; hence, the larger the hypervolume the better. We use the hypervolume (concretely, the implementation provided by Fonseca et al. [81]) not only in combination with irace, but also to compare the various MOACO algorithms.

For the application of irace, we create a training set of 100 randomly generated instances of the bBKP, following the method proposed by Zitzler and Thiele [233]. These instances have random sizes in the range $n \in \{100, \ldots, 750\}$. For comparing the algorithms, we generate a different test set of 50 bBKP instances for each size $n \in \{100, 250, 500, 750\}$. We include in our test set also the four instances by Zitzler and Thiele [233] of sizes $n \in \{100, 250, 500, 750\}$, called ZTZ instances. All algorithms are implemented in C and all experiments are run on a single core of Intel Xeon E5410 CPUs, running at 2.33GHz with 6MB of cache size under Cluster Rocks Linux version 4.2.1/CentOS 4.

The mACO algorithms were originally run with different termination criteria, that is, a different number of iterations, for each variant [5]. To replicate the original mACO experiments, we consider four different computation time limits in our experiments, which correspond to the mean time taken by each of the four mACO variants measured across 25 independent runs on the four ZTZ instances using the corresponding number of iterations (see Table D.2). Then, we compute a formula that approximates the

Table D.2: Termination criteria used in our experiments.

	$\mathbf{TIME}_1$	$\mathbf{TIME}_2$	$\mathbf{TIME}_3$	$\mathbf{TIME}_4$
Time (s)	$0.00001 \cdot n^2$	$0.00003 \cdot n^2$	$0.0001 \cdot n^2$	$0.001 \cdot n^2$
Equivalent to	9000 solutions of mACO_2	3000 solutions of mACO_1	30000 solutions of mACO_3	300000 solutions of mACO_4

Table D.3: Parameter space for tuning the ACO settings of the mACO algorithms.

Parameter	α	β	ρ	q_0	a_f	τ_max method	τ_min method
Domain	$\{0,\ldots,10\}$	$\{0,\ldots,15\}$	$[0.01,1]$	$[0,0.99]$	$\{1,\ldots,30\}$	$\{$*default*, *value*$\}$ *value* $\in [6,100]$	$\{$*default*, *value*$\}$ *value* $\in [0.01,6]$ $\nu \in [1.5,15]$

computation time obtained for each termination criterion. The four resulting termination criteria are given in Table D.2, sorted from the shortest to the longest time.

Comparisons are conducted using empirical attainment functions (EAFs), boxplots of the hypervolume (I^H) and the unary additive epsilon ($I^{\epsilon+}$) indicators [235], and the Friedman non-parametrical test. In this appendix, only few representative results are given; for the complete set of results and the test and training instances we generated, we refer to the supplementary material [23].

D.5 Experimental analysis

D.5.1 Improving the ACO settings of the mACO algorithms

In the first stage of our analysis, we automatically configure the ACO settings of the four mACO variants. The parameter space given to irace is shown in Table D.3. Parameter a_f is a surrogate parameter of the total number of ants, which is given by $N^\text{a} = a_\text{f} \cdot (0.12 \cdot n + 36)$. N^a is rounded to the closest smaller number divisible by three, because mACO_1 and mACO_2 divide the ants into three groups. We apply irace with a budget of 5 000 independent runs in the tuning phase for each mACO algorithm and for each termination criterion TIME_i. Here, the mACO algorithms use their original heuristic information $\eta 1$ [5]. The resulting 16 configurations of mACO are provided as supplementary material [23]. Here, we focus on the configurations obtained when using TIME_4, which are shown in Table D.4.

We compare all algorithms (original and tuned versions) in terms of the hypervolume. We run all algorithms for all four termination criteria 10 independent times on each of the 200 randomly generated bBKP instances (50 instances per instance size $n \in \{100, 250, 500, 750\}$). We normalize the objective values per instance to the interval $[1, 2]$, with 1 corresponding to the maximum value and 2 to the minimum, and compute the hypervolume using the reference point $(2.1, 2.1)$. To analyze the results, we apply the Friedman test, and its associated post-hoc test for multiple comparisons [51], using the median hypervolume obtained by each algorithm on each instance as values, the instances as the blocking factor and the different mACO algorithms as the treatment factor. In all cases, the Friedman test rejects the null hypothesis of equal performance at a significance level of 0.05. Those algorithms whose ranks differ by more than the critical difference are considered to be significantly different. Table D.5 summarizes the results of applying this statistical analysis for each termination criterion. Ranks obtained by each algorithm are shown in parenthesis. The minimum significant rank difference is displayed between parenthesis on the header of each column. The best algorithm and those that are not significantly different from the best are marked in boldface.

From Table D.5, we observe that mACO_2-tuned is the best performing algorithm for all different TIME_i, whereas mACO_4 performs the worst. This seems to contradict the results reported by Alaya

Table D.4: Settings chosen by *irace* for mACO_i-tuned under TIME_4.

Variant	α $\{0,...,10\}$	β $\{0,...,15\}$	ρ $[0.01,1]$	q_0 $[0,0.99]$	τ_{max} **method** $\{$*default*, *value*$\}$	τ_{min} **method** $\{$*default*, *value*$\}$	a_f $\{1,...,30\}$
$mACO_1$-*tuned*	8	1	0.03	0.03	*value* $= 65$	*value* $= 0.33$	27
$mACO_2$-*tuned*	3	1	0.07	0.10	*default*	*default*, $\nu = 6$	26
$mACO_3$-*tuned*	3	1	0.08	0.18	*value* $= 49$	*value* $= 0.34$	2
$mACO_4$-*tuned*	2	1	0.19	0.19	*default*	*default*, $\nu = 8$	5

Table D.5: Friedman test results for I_H obtained by the mACO algorithms.

I_H **TIME**$_1$ (32.957)	I_H **TIME**$_2$ (31.793)	I_H **TIME**$_3$ (35.433)	I_H **TIME**$_4$ (40.745)
mACO_2-tuned (293)	**mACO_2-tuned (208)**	**mACO_2-tuned (220)**	**mACO_2-tuned (227)**
mACO_1-tuned (319)	mACO_1-tuned (402)	mACO_1-tuned (380)	mACO_1-tuned (373)
mACO_2 (591)	mACO_2 (610)	mACO_2 (644)	mACO_3-tuned (757)
mACO_4-tuned (958)	mACO_3-tuned (973)	mACO_1 (987)	mACO_2 (779)
mACO_3-tuned (1005)	mACO_1 (1036)	mACO_3-tuned (1040)	mACO_4-tuned (1076)
mACO_3 (1202)	mACO_4-tuned (1087)	mACO_4-tuned (1073)	mACO_1 (1238)
mACO_1 (1268)	mACO_3 (1301)	mACO_3 (1287)	mACO_3 (1282)
mACO_4 (1564)	mACO_4 (1583)	mACO_4 (1569)	mACO_4 (1468)

et al. [5], which considered mACO_4 as the best performing variant. The different results are explained because, in their case, mACO_4 constructed 100 times more solutions than mACO_2, which roughly requires 100 times more computational time (Table D.2). By contrast, we compare algorithms using the same computation time limit.

The main conclusion we take from these results is that each tuned mACO algorithm clearly outperforms its corresponding original version for each stopping criterion. Hence, we use these tuned variants for comparing against the automatically generated MOACO algorithm in the next section.

D.5.2 Automatically generating MOACO algorithms for the bBKP

In this second stage of our analysis, we automatically configure all parameters of the MOACO framework. In particular, for the parameters specific to the underlying ACO algorithms, we use the same parameter space as for the mACO algorithms (Table D.3). For the multi-objective components, we consider all alternatives described in Table D.1, plus the following ranges: $N^{\text{col}} \in \{1, 2, 5\}$ and $N^{\text{upd}} \in \{1, \ldots, 10\}$. Since N^{a}, the number of ants, has to be divisible by N^{col}, and the result be divisible by N^{weights} (when *awpi* is used), N^{a} was always rounded to the largest smaller number divisible by 10. The weights are defined as, $N^{\text{weights}} \in \{0.2, 5, N^{\text{a}}\}$, when $N^{\text{col}} = 2$, and $N^{\text{weights}} \in \{0.5, 2, N^{\text{a}}\}$, when $N^{\text{col}} = 5$. For single colony versions, only two values were allowed: 0.2 and 0.5. As in the previous section, we apply irace four times, once for each stopping criterion. The budget of each run of irace is 5 000 runs of the MOACO framework. The four resulting configurations are given as supplementary material [23]. Here, we focus on the configuration obtained for TIME_4 (Table D.6).

The analysis of the AutoMOACO configurations shows several commonalities. First, heuristic $\eta 3$ is always chosen, which is different from the one used in the mACO algorithms. Second, the parameter β is always close to the maximum value allowed, thus giving very high importance to the heuristic information. Third, the parameter value of q_0 is also high. This together with the high value of the parameter β implies that most of the items are chosen greedily. Fourth, the number of ants is always very large. For example, 1000 ants are used for instance size 750. As a result, the number of iterations executed by the MOACO algorithm in the given time limits is rather small. It reaches from at most two iterations for the shortest time limits (TIME_1 and TIME_2) to about 60 to 85 iterations for the larger time limit (TIME_4). In the first case, if very few iterations are executed, the algorithm actually behaves

Table D.6: Parameter settings chosen by irace for AutoMOACO: TIME_4.

Parameter	α	β	ρ	q_0	a_f	$\tau_{\max}$	$\tau_{\min}$	N^{col}	N^{weights}	MCWeights	NextWeight	MCUpdate
Value	1	12	0.12	0.57	8	83	2.49	5	N^{a}	$\cap_{50\%}$	*awpi*	*origin*

Parameter	N^{upd}	Selection	Ref.	$\Delta\tau$	$[\tau]$	$[\eta]$	$[\tau]$-Aggreg.	$[\eta]$-Aggreg.	Heuristic
Value	10	BO	BSF	*constant*	*multiple*	*multiple*	*product*	*sum*	$\eta 3$

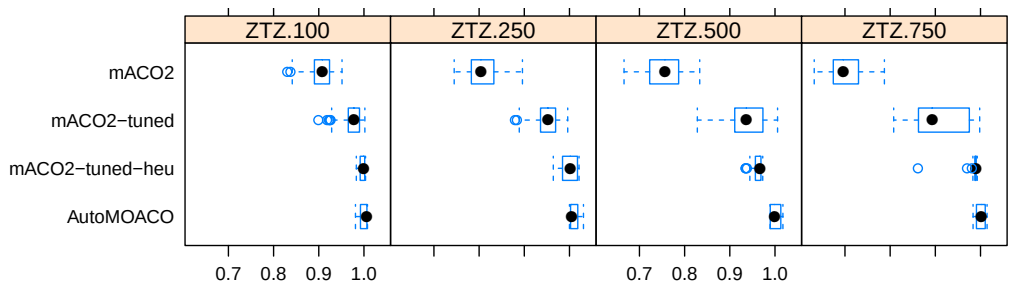


Figure D.1: Boxplots of the I_H indicator for several MOACO algorithms with TIME_4.

Table D.7: Friedman test results for I_H for various MOACO algorithms.

I_H **TIME**$_1$ (14.74)	I_H **TIME**$_2$ (11.416)	I_H **TIME**$_3$ (7.635)	I_H **TIME**$_4$ (5.987)
AutoMOACO (236)	**AutoMOACO (228)**	**AutoMOACO (212)**	**AutoMOACO (204)**
mACO_2-tuned-heu (365)	mACO_2-tuned-heu (373)	mACO_2-tuned-heu (388)	mACO_2-tuned-heu (399)
mACO_2-tuned (611)	mACO_2-tuned (599)	mACO_2-tuned (600)	mACO_2-tuned (597)
mACO_2 (688)	mACO_2 (800)	mACO_2 (800)	mACO_2 (800)

as a greedy construction procedure that performs multiple scalarizations of the bi-objective problem. For the longer time limits, we confirmed that excluding the pheromone information (that is, setting $\alpha = 0$) makes the performance become significantly worse (see supplementary material [23]). This implies that for the larger computation time limits, despite the low number of iterations, the ACO component is effective.

Finally, we compare the performance obtained by the automatically configured MOACO algorithms and the mACO algorithms. Given the high impact of using heuristic information $\eta 3$, we repeated the tuning of each of the mACO variants as described above, but this time leaving open also the choice of the heuristic information. In the following comparison, we consider only the original and the two tuned variants of mACO_2, which are the best mACO variants for each of the time limits. In Fig. D.1 we show boxplots of the hypervolume distribution for the algorithm automatically instantiated from the MOACO framework (AutoMOACO), the original mACO_2, mACO_2 tuned with $\eta 1$ and mACO_2 tuned leaving open the choice of the heuristic information (mACO_2-tuned-heu). The instances shown are the four ZTZ instances. Finally, Table D.7 gives the results of the Friedman test, which is applied as described in Section D.5.1. Clearly, the AutoMOACO algorithm is the top performer, outperforming significantly the other variants. For complete results, we again refer to the supplementary material [23].

D.6 Conclusions

In this chapter, we have extended the MOACO framework [157] to the bBKP and automatically generated MOACO algorithms. The results reported here for the bBKP confirm the previous conclusions obtained in the bTSP, that is, the automatically configured MOACO algorithms outperform the MOACO algorithms from the literature, even after the ACO parameters of the latter have been tuned with the same effort. Interestingly, the MOACO algorithm tuned for very short time limit is rather a repeated stochastic

greedy construction procedure than an ACO algorithm. Although this result may seem counter-intuitive at first, it is, however, a strength of automatic configuration procedures, because they are not biased towards our expectations. The fact that the resulting algorithm is better than the MOACO algorithms proposed in the literature, indicates that the automatic design works as desired, that is, it provides a high-performing algorithm for the given termination criterion. For higher computation time limits, the ACO component of the finally configured algorithm works and contributes to its high performance. Future work should extend the MOACO framework to deal with any number of objectives, and apply the proposed automatic design method to new problems in order to further confirm the above conclusions.

APPENDIX E

Automatic PLS design for the bi-objective knapsack

The efficiency of many successful heuristic algorithms for combinatorial optimization problems is based on the proper use of local search procedures. In fact, many metaheuristics have incorporated the possibility of using local search for example as daemon actions in ant colony optimization and as improvement procedures in genetic algorithms. Pareto local search (PLS) [189] is a straightforward but effective extension of single-objective local search to multi-objective problems. Given a set of solutions, a PLS algorithm consists of selecting one solution at a time and exploring its neighborhood, thus, generating new solutions. These new solutions are added to the initial set, dominated solutions are eliminated, and the algorithm continues until each of the solutions in the solution set has been explored. The performance of PLS algorithms usually tends to depend on (i) the quality of the input solutions, (ii) the definition of the neighborhood structure, (iii) the pivoting rule used for exploring of the neighborhood, and the possible use of candidate lists, and (iv) restrictions on the set of solutions to keep the runtimes manageable. For such reasons, PLS algorithms are well suited for analyzing the impact of design features on the performance of local search procedures for multi-objective combinatorial optimization problems (MCOPs).

In this chapter, we describe and develop a PLS framework, and apply it for the bi-objective bidimensional knapsack problem (bBKP), using the local search components commonly found in the literature. Three sets of experimental analysis are conducted. The first, using a subset of the implemented components (due to the huge number of possible combinations) consists of full factorial designs used to investigate factors and their eventual interactions when PLS is used as a stand-alone optimization method. The experimental setup used aims at isolating the effect of the initial solution set, and the effect of the neighborhood size. Results confirm the dependence of PLS on high-quality input solutions, and that large neighborhoods have to be combined with candidate lists to limit exploration and keep runtimes reasonable. In the second set of experiments, we analyze PLS as a post-optimization method. Two algorithms are used for generating input solutions: (i) a simply greedy procedure, and (ii) AutoMOACO, the automatically generated MOACO algorithm devised in Appendix D for the bBKP. Results show that PLS is able to significantly improve the quality and size of the approximation fronts generated by both algorithms. Finally, the third set of experiments uses all of the implemented components, and consists of automatically generating PLS-based algorithms. Moreover, we also consider the hybridization with the MOACO framework described in Appendix D, thus allowing the automatic design methodology to generate hybridizations of MOACO and PLS. Results show that pure PLS algorithms generate

Algorithm 10 Pareto local search

Input: An initial set of nondominated solutions $\mathcal{A}_0$
1: $\mathsf{explored}(s) \leftarrow \text{FALSE} \quad \forall\, s \in \mathcal{A}_0$
2: $\mathcal{A} \leftarrow \mathcal{A}_0$
3: **repeat**
4: $\quad s \leftarrow \mathsf{NextSolution}(\mathcal{A}_0)$
5: $\quad$ **for all** $s' \in \mathsf{Neighborhood}(s)$ **do**
6: $\quad\quad$ **if** $\mathsf{Acceptance}(s, s', \mathcal{A})$ **then**
7: $\quad\quad\quad \mathsf{explored}(s') \leftarrow \text{FALSE}$
8: $\quad\quad\quad \mathcal{A} \leftarrow \mathsf{Update}(\mathcal{A}_0, s')$
9: $\quad \mathsf{explored}(s) \leftarrow \text{TRUE}$
10: $\quad \mathcal{A}_0 \leftarrow \{s \in \mathcal{A} \,|\, \mathsf{explored}(\text{S}) = \text{FALSE}\}$
11: **until** $\mathcal{A}_0 = \emptyset$
Output: $\mathcal{A}$

high-quality approximation fronts for the bBKP, whereas the hybridization is not an advisable strategy. The remainder of this chapter is organized as follows. Section E.1 introduces the PLS metaheuristic. Section E.2 presents the PLS framework implemented for the bBKP. Section E.3 explains the experimental setup, while the experimental results are discussed in Sections E.4 to E.7. Finally, conclusions and possibilities for future work are discussed in Section E.8.

E.1 Pareto local search

Pareto local search (PLS) is a stochastic local search method for tackling multi-objective problems based on a natural extension of single-objective local search approaches [189]. PLS algorithms use the concept of dominance to extend single-objective local search to multi-objective problems. Starting from an initial set of solutions $\mathcal{A}_0$ (which can also be a singleton), PLS selects at each iteration an unexplored solution $s \in \mathcal{A}$, the current set of non-dominated solutions, and explores the neighborhood of s, generating new solutions. If these new solutions satisfy an acceptance criterion, they are added to $\mathcal{A}$. Once the neighborhood of s is explored, s is flagged so it is not revisited. The algorithm stops when all solutions in $\mathcal{A}$ have been flagged. The three main steps of PLS as shown in Algorithm 10 can be summarized as follows:

Selecting a solution to be explored. The method NextSolution chooses the next solution to be explored. The original PLS chooses the next solution uniformly at random. More recently, other possibilities have been considered, such as selection based on the *optimistic hypervolume improvement* [71], i.e., favoring the selection of solutions that present the best hypervolument contributions (I_H^1) to the solution archive (see Section 2.1).

Exploring the neighborhood. Given a solution s, the method $\mathsf{Neighborhood}(s)$ generates the set of neighbor solutions. Two pivoting rules are commonly used in the literature: (i) *first*, where the neighborhood exploration stops at the first accepted neighbor, or; (ii) *full*, where the neighborhood of s is explored fully and all possible neighbors of s are examined. It is also possible to use combinations of both rules [71]: the first-exploration pivoting rule is used in the beginning to favor intensification until a point is reached when PLS stagnates. Then, the full-exploration pivoting rule is used until the end of the execution of the algorithm.

Accepting new solutions. Given a solution s and a neighbor solution s', the method $\mathsf{Acceptance}(s, s', \mathcal{A})$ determines whether s' is considered an acceptable solution or not. Two common possibilities are: (i) *dominance*, where s' is only accepted if s' dominates s, or; (ii) *nondominance*, where s' is accepted in case s' is nondominated w.r.t. $\mathcal{A}$. The first criterion generates a higher

pressure towards good solutions but it may lead to early stagnation. When using nondominance, the output is likely a well distributed set, but it may lead to high computation times. Again, it is also possible to use combinations of both rules [71]: at the beginning, the archive only accepts dominating solutions. If no acceptable neighbors are found for a given solution, the acceptance condition is relaxed to nondominance and the solution neighborhood is revisited.

Dubois-Lacoste et al. [72] present a review of PLS, highlighting its use both as a stand-alone procedure and in hybrid algorithms. Regarding stand-alone PLS, the authors identify studies focusing on time-limited experiments [152], on anytime behavior [71] and on how to restart or continue the search after PLS converges [6, 67, 89]. Regarding PLS as a post optimization procedure, Dubois-Lacoste et al. [69] have proposed algorithms that are currently state-of-the-art for several bi-objective flowshop problems.

E.2 Applying PLS to the bBKP

In this chapter, a solution x for the bBKP is represented as a list s of size n, where the first $n_s \leq n$ items are considered to be in the knapsack[1]. Furthermore, each solution has as an associated profit vector $\vec{p}_s = (p_s^1, p_s^2)$, where $p_s^c = \sum_{i=1}^{n} p_i^c x_i^s$, $c = 1, 2$, and a load vector $\vec{w}_s = (w_s^1, w_s^2)$, where $w_s^j = \sum_{i=1}^{n} w_i^j x_i^s$, $j = 1, 2$.

Two methods have been implemented for generating the initial solution(s) for PLS: (i) *random*, where one or more random solutions are generated, and (ii) *greedy*, where a set of greedy solutions is generated. Random solutions are generated by choosing an item uniformly at random at each construction step, until no more items can be added due to the capacity constraints. When using greedy solutions, a set of uniformly distributed weights $\Lambda = \{\lambda_1, \ldots, \lambda_z\}$ is generated, where z is the number of input solutions. For each $\lambda \in \Lambda$, a greedy solution is generated using one of the following heuristic functions [163]:

$$\eta_1(i) = \frac{\lambda\, p_i^1 + (1-\lambda)\, p_i^2}{\sum_{j=1}^{m} w_i^j} \qquad \eta_2(i) = \frac{\lambda\, p_i^1 + (1-\lambda)\, p_i^2}{\sum_{j=1}^{m} \frac{w_i^j}{W_j - w_s^j + 1}} \tag{E.1}$$

All actions related to neighborhood exploration are encapsulated in the procedure Neighborhood. This procedure systematically explores the neighborhood of a solution s, returning a set of neighbors. The neighborhood operator used is the r-remove operator, which removes up to r items from the knapsack. In our solution representation, removing one item at the i-th position of the list means exchanging it with the last selected item, i.e., the item at position n_s of the list, and decreasing the value of n_s by one.

Solutions are reconstructed by filling the knapsack with items found at positions $i = n_s + r, \ldots, n$ of the list, in the order they appear. Since biasing the search is important, in the beginning of the algorithm items not selected in the input solution are ordered. Formally, given an input solution s, let $\mathcal{IN}(s) = \{i \,|\, s_i = 1\}$ be the set of n_s items inside the knapsack and $\mathcal{OUT}(s) = \{i \,|\, s_i = 0\}$ be the set of items outside the knapsack. The procedure SolutionOrdering is used to order items $x_i \in \mathcal{OUT}(s)$ in a nondecreasing order according to their heuristic value. The heuristics used for this ordering are the same presented in Eq. E.1. The weights used by the heuristic functions are generated on a *per solution* basis. Given a solution s, we tested nine methods for computing λ (Table E.1). For efficiency, candidate lists are used to constrain the set of items that are considered for removal; in other words, items that are not member of the candidate list are never considered for removal in a current solution. Given a solution s, the candidate list of items for removal contains the L last items of the list, i.e., items at positions $i, n_s - L < i \leq n_s$. Two methods have been used for determining parameter L: (i) *all*, where $L = n_s$, and (ii) *input*, where L is input by the user.

[1] Nonetheless, the implementation also keeps the Boolean vector x, allowing the algorithm to know if an item is in the knapsack or not in $O(1)$.

Table E.1: Methods used for computing the weights used by SolutionOrdering.

Method	**Formula**	**Description**
equal	$\lambda = 0.5$	equal weights
random-discrete	$\lambda \in \{0, 1\}$	uniformly randomly chosen
random-continuous	$\lambda \in [0, 1]$	uniformly randomly chosen
largest-gap	$\lambda = i,\ min(w_s^i)$	privileges dimension with more free space
smallest-gap	$\lambda = i,\ max(w_s^i)$	privileges dimension with less free space
highest-profit	$\lambda = i,\ max(f^i(s))$	privileges objective with the highest value
lowest-profit	$\lambda = i,\ min(f^i(s))$	privileges objective with the lowest value
proportional-same	$\lambda = w_s^1/(w_s^1 + w_s^2)$	proportional to the loads
proportional-opposite	$\lambda = w_s^2/(w_s^1 + w_s^2)$	inversely proportional to the loads

The pseudocode for Neighborhood can be seen on Algorithm 11. C_r stands for a combination of r items, whereas C_* stands for a combinations of any number of items. Accepted checks if the solution s' is acceptable according to the acceptance criteria selected. Finally, the search is controlled by the following pivoting-rules:

1. *remove-first*: given that the first accepted neighbor is generated by the removal of the item found at the i-th position of s, $n_s - L < i \leq n_s$, Neighborhood does not explore the insertion possibilities generated by the removal of items found at the j-th position of s, $\forall j, n_s - L < j < i$;

2. *remove-full*: Neighborhood explores the insertion possibilities generated by the removal of each of the items in the candidate list.

3. *insert-first*: given the insertion possibilities generated by the removal of an item x_i, Neighborhood stops at the first combination of items that produces an accepted neighbor.

4. *insert-full*: given the insertion possibilities resulting from the removal of an item x_i, Neighborhood generates all acceptable neighbors.

The pseudocode for the bBKP-PLS algorithm can be seen on Algorithm 12. Firstly, the algorithm generates initial solutions if none are provided (line 1). Then, a weight λ is generated for each solution $s \in \mathcal{A}_0$ (line 3), and used in the SolutionOrdering procedure (line 4). Finally, PLS is called (line 6). In addition, when Neighborhood checks if a neighbor solution is acceptable regarding the original solution, it also checks if its acceptable w.r.t. to the set $\mathcal{A}$. This is done to ensure that $\mathcal{A}$ is extended by at least one solution if such a solution exists in the neighborhood of s.

E.3 Experimental setup

For the analysis of PLS, we use the same 200 bBKP instances described in Appendix D as a test set. We use a full factorial design, and each configuration is run 10 times per instance. All parameters are analyzed both individually and for possible interactions through plots of the median I_H of the approximation sets they produce. When comparing two algorithms using boxplots of their I_H or plots of the differences of their empirical attainment functions (EAFs) [158], we use the four ZTZ instances previously used in Appendix D, and we run the algorithms 25 independent times per instance. When more algorithms are simultaneously compared, we use the boxplots of the I_H for a sample of 4 instances of each size: the ZTZ instances plus three instances from the test set used for the analysis of PLS. For automatically designing the hybrid AutoMOACO+PLS algorithm, we use irace [14, 159], also with a budget of 5000 experiments.

Algorithm 11 Procedure Neighborhood

Input: An input solution s
1: $\mathcal{N} \leftarrow \emptyset$
2: **for all** $C_r \in$ candlist(s) **do**
3: $s' \leftarrow$ Remove(s, C_r)
4: **for all** $C_* \in \mathcal{OUT}(s)$ **do**
5: $s'' \leftarrow$ Insert(s', C_*)
6: **if** Accepted(s'', s') **then**
7: $\mathcal{N} \leftarrow \mathcal{N} \cup s''$
8: **if** *insertion-first* **then**
9: found $\leftarrow$ true; break;
10: **if** found and *removal-first* **then**
11: break
Output: A set of neighbor solutions $\mathcal{N}$

Algorithm 12 bBKP-PLS

Input: An input method input $\in \{$*random*, *greedy*$\}$
1: $\mathcal{A}_0 \leftarrow$ InitialSolutions(input)
2: **for all** $s \in \mathcal{A}_0$ **do**
3: $\lambda \leftarrow$ GenerateLambda(s)
4: SolutionOrdering(s, λ)
5: $\mathcal{A} \leftarrow$PLS(A_0)
Output: $\mathcal{A}$

All algorithms are implemented in `C` and all experiments are run on a single core of Intel Xeon E5410 CPUs, running at 2.33GHz with 6MB of cache size under Cluster Rocks Linux version 6.0/CentOS 6.3. In this chapter, only few representative results are given. The complete set of results are made available as a supplementary page [24].

E.4 Experiments with stand-alone PLS

In this section, we present the results of the analysis of stand-alone PLS. For this set of experiments, we consider random selection of solutions in PLS and acceptance based on nondominance. To properly isolate the effect of the neighborhood operator, this analysis is divided in two stages: (i) experiments removing one item ($r = 1$), and (ii) experiments removing more than one item ($r > 1$). The parameter space used for this analysis comprises all possible values previously described, summarized in Table E.2.

Removing a single item. The different possibilities of choosing the weights (λ) and the heuristic information (η) do not have a strong influence on the results, hence, we do not present here a detailed analysis of these parameters. Instead, we analyze the initialization method, the length of the candidate list, and the removal and insertion pivoting rules. The results presented here are aggregated across all possible settings of λ and η. Detailed results can be found on the supplementary material. We first analyze stand-alone PLS initialized with a single random solution. Fig. E.1 (left) shows the median I_H on the y-axis, and the instances grouped by sizes on the x-axis. The lines represent the median I_H obtained by different configurations, and are ordered according to performance on the larger instance. It is clear that using candidate lists when $r = 1$ is a bad decision, since the I_H quickly degenerates as the instance size grows. However, as shown on Fig. E.1 (right), the runtimes of configurations that do not use candidate list tend to grow very quickly (y-axis).

The analysis of PLS starting from greedy solutions requires an additional parameter, namely the number of weights used for generating input solutions. Experiments were conducted for 2, 10 and 50

Table E.2: Parameter values used for the analysis of stand-alone PLS.

Parameter	**Values**
η	η_1, η_2
λ	*equal, random-discrete, random-continuous, largest-gap, smallest-gap, highest-profit, lowest-profit, proportional-same, proportion-opposite*
candidate list	*all,* $L = 15$*,* $L = 30$*,* $L = 50$
removal rule	*removal-first, removal-full*
insertion rule	*insertion-first, insertion-full*

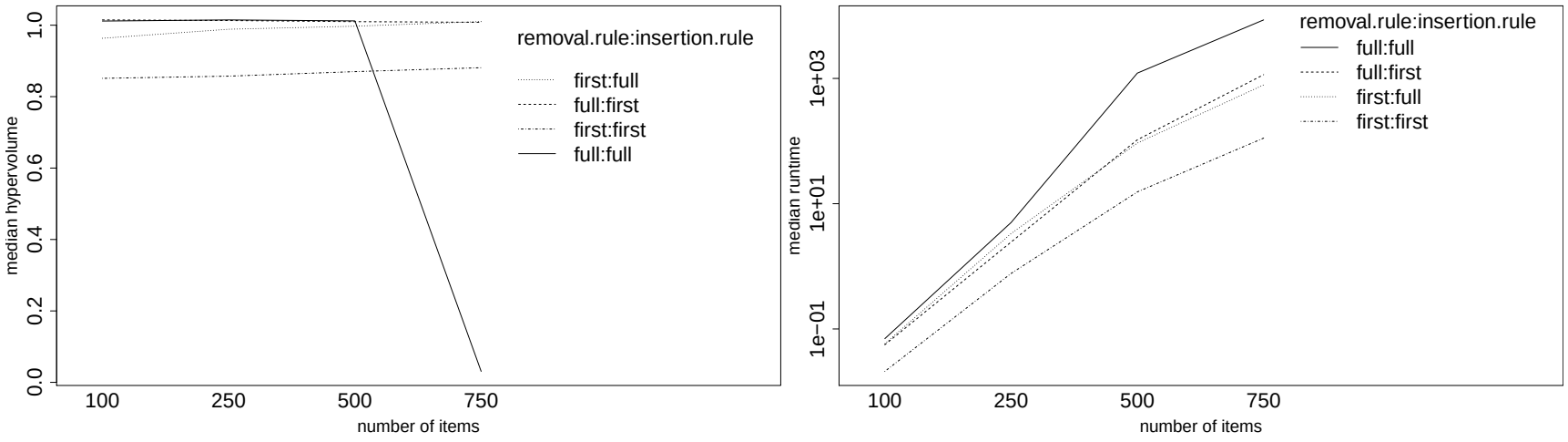


Figure E.1: Median I_H (left) and runtime (right) of different combinations of pivoting rules for PLS starting from a random initial solution and $r = 1$.

input weights, and this parameter proved critical for the performance of the algorithm. When only two input weights are used, the performance of the algorithm is really poor and similar to when a random initial solution is used. On the other hand, using 50 weights not only improves the final result quality, but also helps the algorithm converge faster compared to other settings. Therefore, we focus on the experiments using 50 initial weights for generating the greedy solutions. Also in this case, not using candidate lists leads to long runtimes. However, the solution quality does not degenerate when larger values of L are adopted. Fig. E.2 (left) shows the median I_H (y-axis) grouped by instance sizes (x-axis). The best performing versions are the ones that use $L = \{30, 50\}$, *removal-full* and *insertion* $\in$ {*first*, *full*} ($L = 30$ not shown here due to space reasons). When analyzing runtimes (see also Fig. E.2, right), a candidate list of size 50 combined with removal-full and insertion-first is a setting that takes computation times similar to those that are used as time limits in the analysis of state-of-the-art algorithms [22, 163].

Removing more than one item. For the analysis of PLS with $r > 1$, the same parameter space used in the previous experiments is adopted. However, given that in the previous experiments with $r = 1$ we observed that the parameters used for SolutionOrdering (that is, the heuristic and the values of λ) behave very similarly, we narrowed down the number of configurations to be tested by selecting η_1 as the heuristic function and *highest-profit* as the method for defining λ. The experiments were limited to a maximum runtime of 5 hours per run.

Figure E.3 shows the results for $r = 2$. In terms of solution quality, there is a big difference between configurations that use *removal-first* and *insertion-first* and the ones that do not. Moreover, the best configurations for small instance size become much worse with larger instance sizes. The reason is that those configurations reach the CPU-time limit of 5 hours and were stopped before completion. In fact, the only configurations with runtime lower than 1 000 seconds are the ones that either (i) use a candidate list ($L \leq 50$) combined with *removal-first* and *insertion-first*, or; (ii) use a candidate list with $L = 15$ combined with *removal-first* and *insertion-full*.

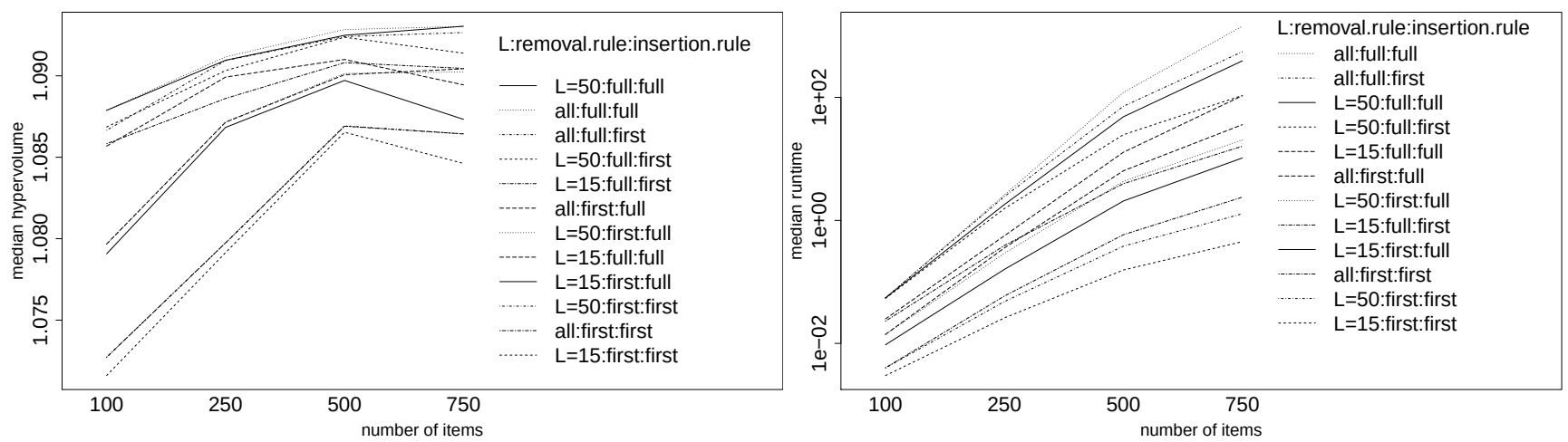


Figure E.2: Median I_H (left) and runtime (right) of different combinations of pivoting rule and candidate list sizes for PLS starting from greedy solutions and $r = 1$.

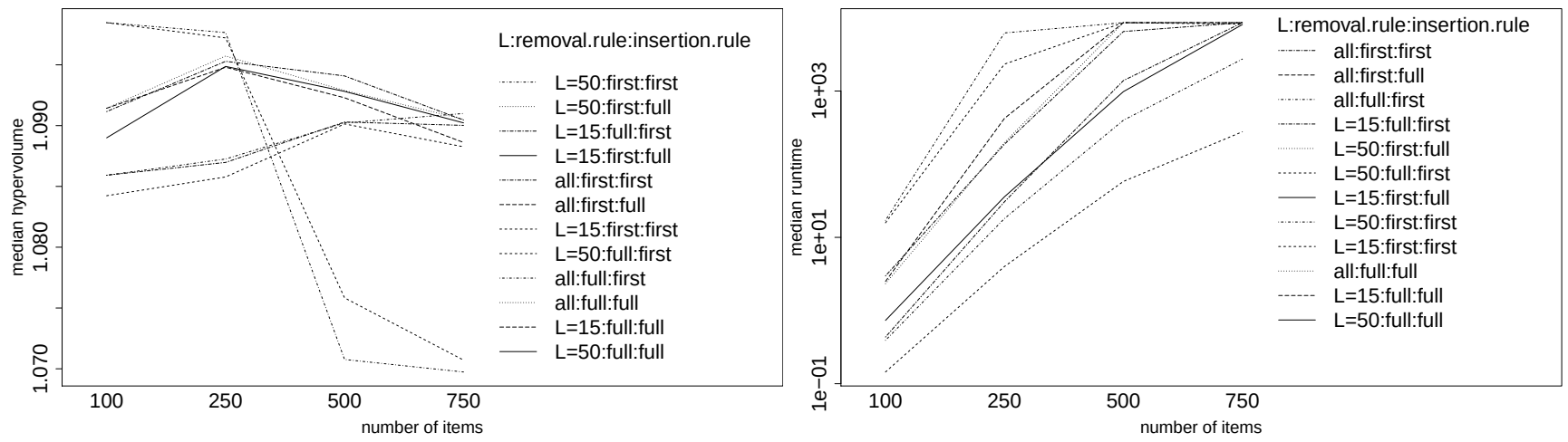


Figure E.3: Median hypervolume (left) and runtime (right) of different combinations of pivoting rule and candidate list sizes for PLS starting from greedy solutions and $r = 2$.

E.5 Experiments with PLS as a post-optimization method

To analyze PLS as a post-optimization method, we start by comparing PLS against the greedy procedure used for generating its input solutions. The motivation for this comparison is to understand whether PLS is actually significantly improving the input set or simply adding more nondominated solutions. The greedy procedure is run with η_1 and $2n$ weights, which is roughly the same number of solutions expected to be found by stand-alone PLS. The parameters used by PLS are: η_1, *highest-profit*, $L = 50$, *removal-full*, *insertion-first*, and $r = 1$. Fig. E.4 shows that the difference between the greedy procedure and PLS (using $2n$ weights for greedy solutions) is quite strong. The approximation set identified by PLS dominates the output of the greedy procedure across the entire range of the front, which means PLS is able to substantially improve all initial solutions. In addition, PLS finds a much larger approximation set.

We also add PLS as a post-optimization procedure to AutoMOACO, which is run using the same parameter setting and time limit described in Appendix D. The resulting approximation set is given as input to PLS, which is then run until completion using the same parameters as above. Fig. E.5 shows the EAF difference for ZTZ 750. Again, PLS is able to improve the approximation fronts over the entire objective space, while the runtimes of PLS remain low (see Table E.3). Thus, PLS significantly improves the approximation obtained by AutoMOACO, incurring only a reasonable computational overhead. To conclude this analysis, we compare all four algorithms considered in this section. Figure E.6 shows the boxplot of the hypervolume for all four ZTZ instances. As expected, the greedy procedure presents the worst hypervolume values. Among the remaining algorithms, AutoMOACO and stand-alone PLS perform similarly on most instances. Interestingly, although the number of solutions is greatly increased

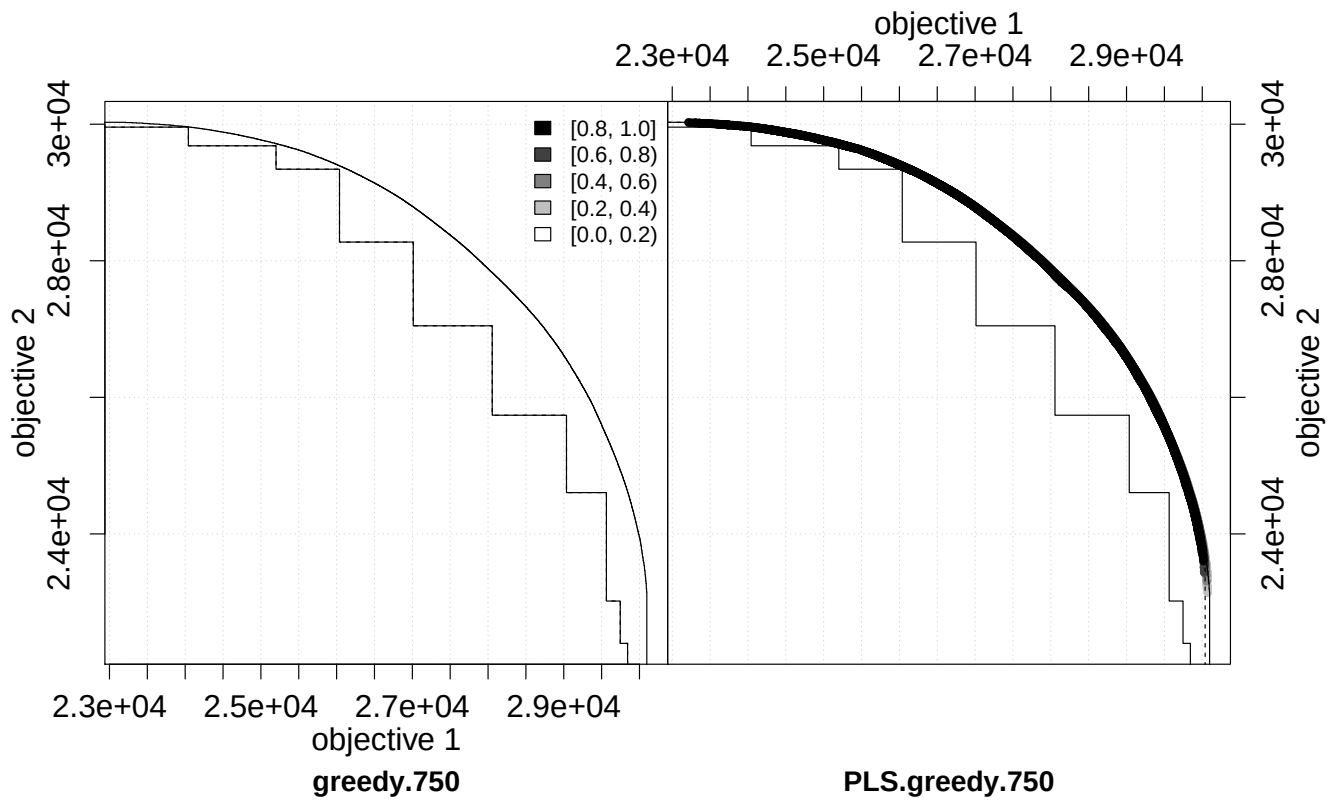


Figure E.4: EAF difference plot. Greedy solutions using $2n$ weights (Greedy) vs. PLS initialized with greedy solutions using $2n$ weights ($\text{PLS}_{\text{greedy}}$). Instance ZTZ 750.

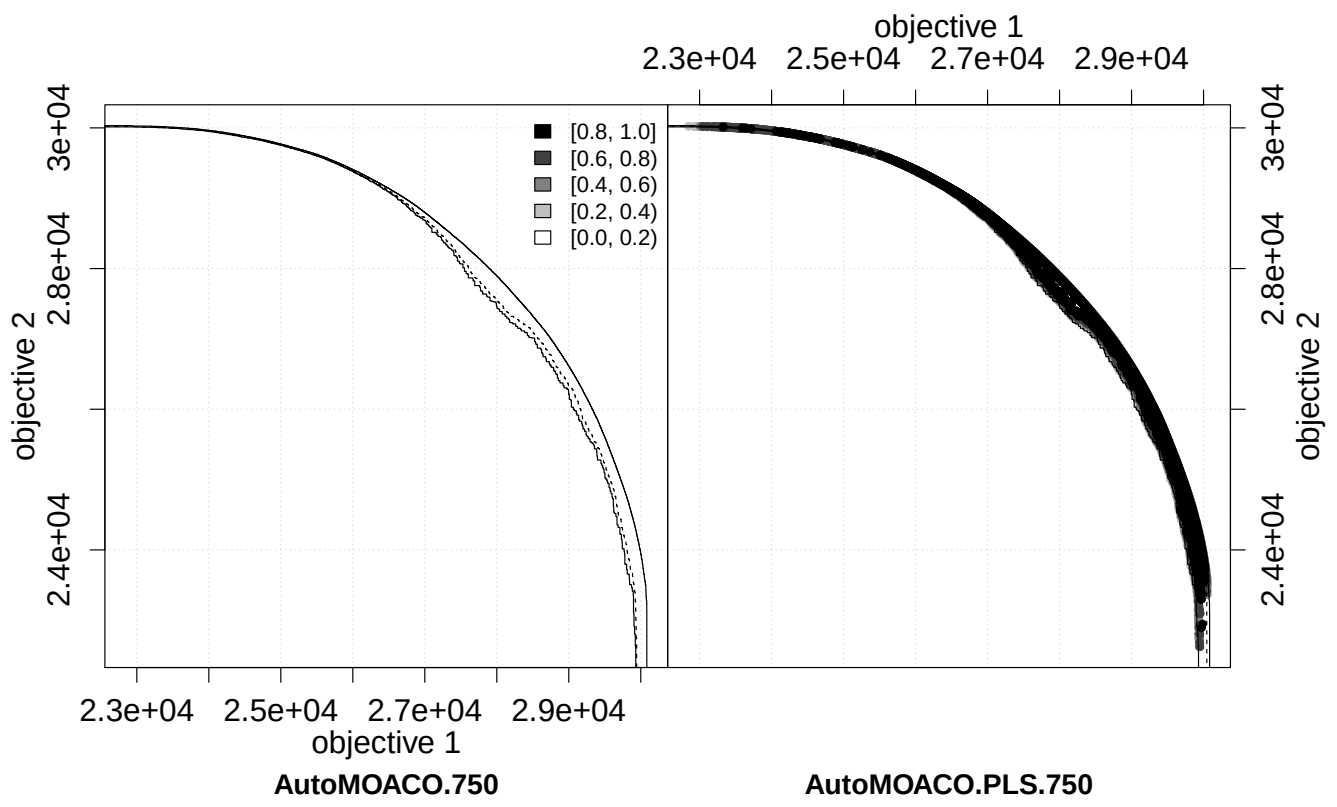


Figure E.5: EAF difference plot. AutoMOACO vs. AutoMOACO+PLS. Instance ZTZ 750.

when PLS is used as post-optimization for AutoMOACO, the differences in the hypervolume are relatively small. For the largest instance, the performance of AutoMOACO decreases considerably, but PLS is able to compensate such loss.

E.6 Further SLS approaches

Besides the algorithmic components identified in the PLS literature, the AutoPLS framework we propose in the next section contains components from other SLS approaches, namely *variable neighborhood search* (VNS) and iterated greedy (IG). For a better understanding of these components, we briefly

Table E.3: Average number of solutions and runtime: ZTZ instances, 25 runs.

	ZTZ 100	**ZTZ 250**	**ZTZ 500**	**ZTZ 750**
Greedy	13 (0.00s)	43 (0.01s)	69 (0.06s)	114 (0.12s)
PLS$_{greedy}$	98.81 (0.04s)	356.46 (1.08s)	742.38 (7.64s)	1502.2 (40.35s)
AutoMOACO	81.84 (1s)	287.84 (6.25s)	376 (26s)	273.08 (56.25s)
AutoMOACO+PLS	110.48 (1.03s)	382.76 (7.27s)	807.28 (33.48s)	1542.48 (114.6s)

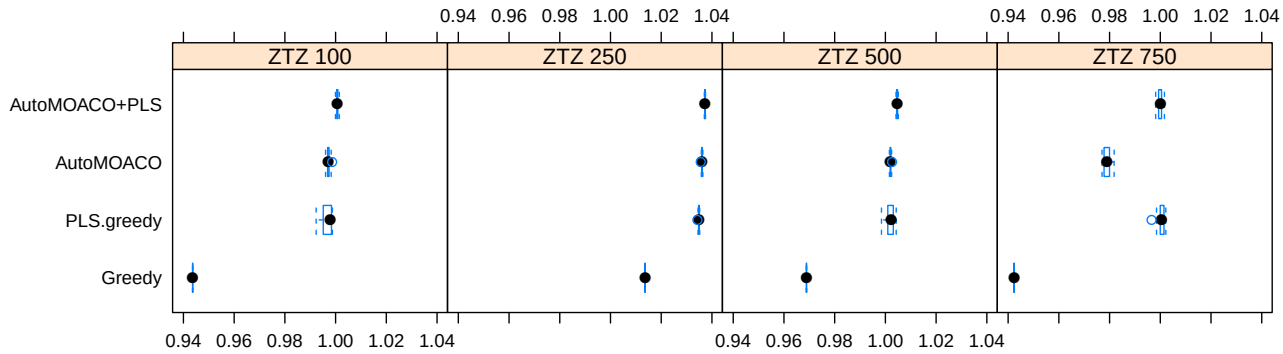


Figure E.6: I_H boxplot for ZTZ instances.

review these two approaches.

Variable neighborhood search comprises a set of standard algorithms proposed by Hansen and Mladenović [103] which try to escape local optima by using multiple neighborhood operators. The underlying concept behind VNS algorithms is that the local optimum for a given neighborhood operator σ_1 is usually not a local optimum for another operator σ_2. By systematically switching between different neighborhood operators, VNS algorithms are able to combine their search power. The simplest version of the VNS metaheuristic is the *variable neighborhood descent* (VND) procedure, which is often used in place of traditional local search procedures. VND procedures make use of multiple neighborhood operators σ_i, $i = \{1, ..., k\}$, usually ordered according to their computational cost. When an operator σ_i reaches a local optimum, the following operator, σ_{i+1}, is used. If σ_{i+1} is able to escape the local optimum, i.e., find an improving neighbor solution, the algorithm switches back to σ_1. Else, σ_{i+2} is used. Typically, VND procedures use at most two or three distinct operators, but are to able find results significantly better than traditional LS procedures [104].

Iterated greedy is a simple but efficient algorithmic strategy based on the idea of very large neighborhood search (VLNS). The concept of VLNS consists on exploring large neighborhoods, since the larger the neighborhood, the greater the chance of it having a near-optimal local optimum. However, since the search space defined by these large neighborhoods generally forbids systematic explorations, heuristic exploration methods need to be employed. VLNS algorithms can be implemented in several different ways as described by Pisinger and Ropke [191]. The iterated greedy (IG) metaheuristic [127], also known as *large neighborhood search* [210] or *ruin and recreate* [205], implements the VLNS concept by means of two algorithmic steps: (i) *destruction*, i.e., destroying part of the incumbent solution, and; (ii) *construction*, i.e, reconstructing the solution through constructive procedures. The definition of each of these algorithmic steps is of major importance to IG algorithms. First, the *degree of destruction* employed by the destruction phase must be carefully set. This parameter is typically set as a ratio of the solution, but this can lead to huge search spaces if large instances are considered. On the other hand, if fixed values are set, the absolute number of solution components removed may be too small, causing the algorithm to stagnate by always reconstructing the same solution(s). Regarding the construction phase, algorithm engineers may

choose between optimal or heuristic procedures. Using optimal procedures, high-quality solutions can be retrieved, but if the neighborhood is too large this can significantly reduce the performance of the algorithm. When heuristic procedures are used, solutions will be reconstructed fast, but the procedure may always generate the same solution if the partial solutions are too similar. For this reason, the destruction phase should preferably remove solution components randomly [191].

E.7 Automatic design using the PLS framework

After collecting insights through the large experimental campaigns described in the previous sections, we proceed to automatically generate PLS algorithms using the PLS framework. To do so, we allow irace to choose all components described in Sections E.1 and E.2. We then use the automatic algorithm design methodology as previously described. In particular, the remaining following algorithmic components have been applied to the bBKP. The full parameter space given to irace can be seen on Table E.4.

Dynamic acceptance criterion: by default, PLS uses *dominance* as acceptance criterion. However, when PLS is unable to find an acceptable neighbor for a solution s, the acceptance criterion is automatically switched to *nondominance*.

Dynamic pivoting rules: applicable to both removal and insertion rules. By default, a *first* rule is adopted. The algorithm only switches to a *full* rule when the archive stagnates. If both removal and insertion rules are set to be dynamic, the algorithm adopts the following sequence of rule combinations: (i) *removal-first, insertion-first*; (ii) *removal-first, insertion-full*, and; (iii) *removal-full, insertion-first*.

Variable neighborhood descent: PLS is allowed to increase the neighborhood size in case the archive has stagnated and previous approaches have already been tried. If dynamic pivoting rules are being used, they are reseted to their initial states.

Iterated-greedy initialization: an iterated-greedy approach is used to generate initial solutions. The size of the part of the solution to be destroyed is set according to a specific parameter, which can be defined as absolute or relative to the instance size.

Insertion candidate list: when exploring the neighborhood of a solution, the local search procedure may use a candidate list also for the items to be inserted in the knapsack.

Randomized/inverted local search: when choosing which items to remove from a solution, the local search procedure may start from the beginning of the permutation rather than from its end (inverted search direction). Moreover, it may also start from a randomly chosen starting point.

The configuration selected by irace for AutoPLS can be seen on Table E.5. Several important observations can be made from the analysis of the selected parameters. First, contrarily to what we expected, the iterated-greedy component was not selected. This is due most likely to the fact that our current implementation of this component is still incipient, lacking more powerful selectable subcomponents for irace. Moreover, the number of input weights is set to an absurdly high value (1000). This remains yet to be investigated, but one possible explanation is that the largest instance sizes demand more initial solutions. The second important group of observations is that the components which had been manually pointed out as best-performing through an extensive full factorial analysis are selected by irace. This means that even among so many design possibilities the automatic design allows the generation (and selection) of a complex layout such as (i) using nondominance since initial solutions are already high-quality solutions; (ii) using a faster removal pivoting rule to make the VND component feasible; and (iii) using roughly the same candidate list size for insertion and removal, which is a critical component of the current state-of-the-art.

Table E.4: Parameter space used by irace for the generation of AutoPLS.

Parameter	**Values**
input set	random, greedy, iterated-greedy
$\|\Lambda\|$	$\{2, 5, 10, 50, 100, 250, 500, 1000, 1500, 2000, 2500\}$
iterated-greedy destruction size	$\{10, \ldots, 50\}$
η	η_1, η_2
λ	*equal, random-discrete, random-continuous, largest-gap, smallest-gap, highest-profit, lowest-profit, proportional-same, proportion-opposite*
acceptance	*dominance, nondominance, dynamic*
removal rule	*removal-first, removal-full, dynamic*
removal candidate list (RMC)	*all,* $L = 15$, $L = 30$, $L = 50$
r_0	1, 2
r_{max}	2, 3
insertion rule	*insertion-first, insertion-full, dynamic*
insertion candidate list (ICL)	*proportional, fixed-size, none*
ICL ratio	$\{5\%, \ldots, 30\%\}$ (when a proportional candidate list is used)
ICL size	$\{5, \ldots, 50\}$ (when a fixed size candidate list is used)

Table E.5: Parameter settings chosen by irace for AutoPLS: TIME_4.

Parameter	input set	$\|\Lambda\|$	η	λ	*acceptance*	*removal rule*
Value	*greedy*	1000	η_1	*highest-profit*	*nondominance*	*first*

Parameter	*RMC*	r_0	r_{max}	*insertion rule*	*ICL*	*ICL size*
Value	$L = 30$	1	2	*dynamic*	*fixed-size*	33

We also attempted to automatically generate a hybrid algorithm combining MOACO and PLS (the latter as post-optimization). The parameter space provided to irace was the union of the parameter space of each individual framework, plus an extra parameter used to define the ratio of time PLS would be allowed to run. Under the given experimental setup, though, irace chooses not to use PLS. This behavior is most likely explained by the fact that AutoMOACO needs most of the runtime for generating high quality solutions, and hence the hybridization is not beneficial to the output quality. Although the experimental evaluation of the algorithms devised in this section in still ongoing work, the EAF difference plot depicted on Figure E.7 shows results are promising as expected.

E.8 Conclusions

In this chapter, we have proposed a Pareto local search (PLS) framework and applied it to the biobjective bidimensional knapsack problem. We have analyzed the impact of common local search components found in the literature, and empirically investigated by how much it can improve over existing algorithms. Our results show that the performance of stand-alone PLS strongly depends on high-quality input solutions. However, such solutions can be generated without significant computational overhead using greedy (meta)heuristics. Large neighborhood sizes proved prohibitive w.r.t. computation time

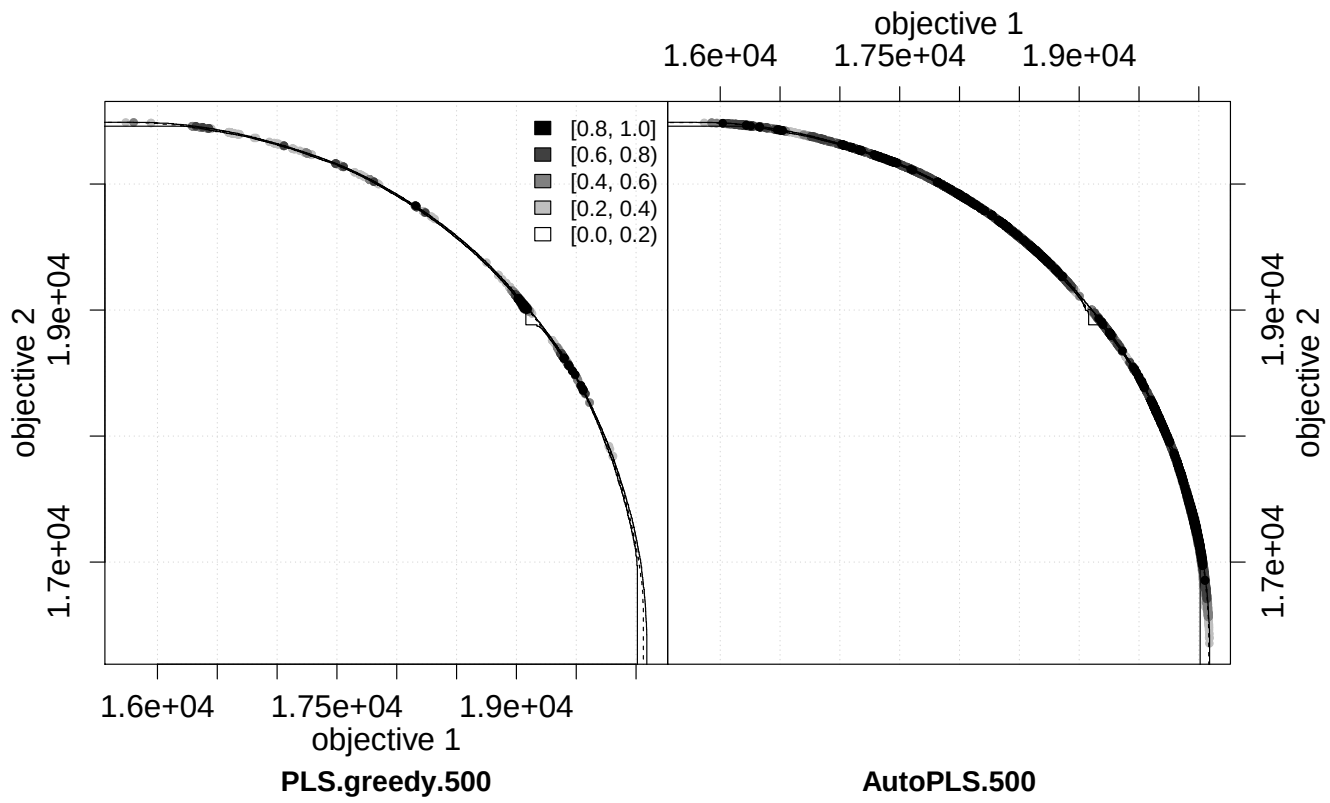


Figure E.7: EAF difference plot. PLS started from greedy solutions (PLS_{greedy}) vs. AutoPLS. Instance ZTZ 500.

even when combined with a candidate list. Nevertheless, archiving mechanisms that constraint the size of the approximation set remain to be tested.

Moreover, once again the automatic design methodology has proved effective, showing us that (i) automatic generated PLS algorithms are a promising research path, and (ii) for the experimental setup used here MOACO algorithms require a lot of computational runtime to generate high-quality solutions and hence hybridizing it with PLS is not advisable. The research on automatic design using the PLS framework is still ongoing work. The most important next steps are: (i) investigate efficient methodologies for generating high-quality initial solutions at a small computational cost, and (ii) perform experimental evaluations on more problems, and also with more objectives.

APPENDIX F

Bibliography

[1] Hussein A. Abbass. The self-adaptive Pareto differential evolution algorithm. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*, pages 831–836, Piscataway, NJ, 2002. IEEE Press.

[2] Hussein A. Abbass, Ruhul Sarker, and Charles Newton. PDE: a Pareto-frontier differential evolution approach for multi-objective optimization problems. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC'01)*, pages 971–978, Piscataway, NJ, 2001. IEEE Press.

[3] Hernán E. Aguirre. Advances on many-objective evolutionary optimization. In Christian Blum and Enrique Alba, editors, *GECCO (Companion)*, pages 641–666, New York, NY, 2013. ACM Press.

[4] Hernán E. Aguirre and Kiyoshi Tanaka. Many-objective optimization by space partitioning and adaptive ϵ-ranking on MNK-landscapes. In Matthias Ehrgott, Carlos M. Fonseca, Xavier Gandibleux, Jin-Kao Hao, and Marc Sevaux, editors, *Evolutionary Multi-criterion Optimization, EMO 2009*, volume 5467 of *Lecture Notes in Computer Science*, pages 407–422. Springer, Heidelberg, Germany, 2009.

[5] I. Alaya, C. Solnon, and Khaled Ghédira. Ant colony optimization for multi-objective optimization problems. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, volume 1, pages 450–457. IEEE Computer Society Press, Los Alamitos, CA, 2007.

[6] A. Alsheddy and E. Tsang. Guided Pareto local search and its application to the 0/1 multi-objective knapsack problems. In M. Caserta and Stefan Voß, editors, *Proceedings of MIC 2009, the 8th Metaheuristics International Conference*, Hamburg, Germany, 2010. University of Hamburg.

[7] Y. P. Aneja and K. P. K. Nair. Bicriteria transportation problem. *Management Science*, 25(1): 73–78, 1979.

[8] Carlos Ansótegui, Meinolf Sellmann, and Kevin Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In Ian P. Gent, editor, *Principles and Practice of Constraint Programming, CP 2009*, volume 5732 of *Lecture Notes in Computer Science*, pages 142–157. Springer, Heidelberg, Germany, 2009.

[9] Carlos Ansótegui, Yuri Malitsky, and Meinolf Sellmann. MaxSAT by improved instance-specific algorithm configuration. In David Stracuzzi et al., editors, *AAAI*, pages 2594–2600. AAAI Press, 2014.

[10] Carlos Ansótegui, Yuri Malitsky, Horst Samulowitz, Meinolf Sellmann, and Kevin Tierney. Model-based genetic algorithms for algorithm configuration. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI-15)*, pages 733–739. IJCAI/AAAI Press, Menlo Park, CA, 2015.

[11] Anne Auger and Nikolaus Hansen. A restart CMA evolution strategy with increasing population size. In *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, pages 1769–1776. IEEE Press, Piscataway, NJ, September 2005.

[12] Anne Auger, Dimo Brockhoff, Nikolaus Hansen, Dejan Tusar, Tea Tušar, and Tobias Wagner. GECCO workshop on real-parameter black-box optimization benchmarking (BBOB 2016): Focus on multi-objective problems. `https://numbbo.github.io/workshops/BBOB-2016/`, 2016.

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

[14] Prasanna Balaprakash, Mauro Birattari, and Thomas Stützle. Improvement strategies for the F-race algorithm: Sampling design and iterative refinement. In Thomas Bartz-Beielstein, María J. Blesa, Christian Blum, Boris Naujoks, Andrea Roli, Günther Rudolph, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 4771 of *Lecture Notes in Computer Science*, pages 108–122. Springer, Heidelberg, Germany, 2007.

[15] Benjamín Barán and Matilde Schaerer. A multiobjective ant colony system for vehicle routing problem with time windows. In *Proceedings of the Twenty-first IASTED International Conference on Applied Informatics*, pages 97–102, Insbruck, Austria, 2003.

[16] Thomas Bartz-Beielstein. *Experimental Research in Evolutionary Computation: The New Experimentalism*. Springer, Berlin, Germany, 2006.

[17] Thomas Bartz-Beielstein, C. Lasarczyk, and Mike Preuss. The sequential parameter optimization toolbox. In Thomas Bartz-Beielstein, Marco Chiarandini, Luís Paquete, and Mike Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 337–360. Springer, Berlin, Germany, 2010.

[18] William J. Baumol. Management models and industrial applications of linear programming. *Naval Research Logistics Quarterly*, 9(1):63–64, 1962.

[19] Nicola Beume, Boris Naujoks, and Michael Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[20] Nicola Beume, Carlos M. Fonseca, Manuel López-Ibáñez, Luís Paquete, and Jan Vahrenhold. On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, 13(5):1075–1082, 2009.

[21] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution stratagies: a comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

[22] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. Automatic generation of multi-objective ACO algorithms for the biobjective knapsack. In Marco Dorigo et al., editors, *Swarm Intelligence, 8th International Conference, ANTS 2012*, volume 7461 of *Lecture Notes in Computer Science*, pages 37–48. Springer, Heidelberg, Germany, 2012.

[23] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. Automatic generation of MOACO algorithms for the biobjective bidimensional knapsack problem: Supplementary material. `http://iridia.ulb.ac.be/supp/IridiaSupp2012-008/`, 2012.

[24] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. An analysis of local search for the bi-objective bidimensional knapsack: Supplementary material. `http://iridia.ulb.ac.be/supp/IridiaSupp2012-016/`, 2013.

[25] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. Automatic component-wise design of multi-objective evolutionary algorithms. Technical Report TR/IRIDIA/2014-012, IRIDIA, Université Libre de Bruxelles, Belgium, Brussels, Belgium, August 2014.

[26] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. Deconstructing multi-objective evolutionary algorithms: An iterative analysis on the permutation flowshop. In Panos M. Pardalos, Mauricio G. C. Resende, Chrysafis Vogiatzis, and Jose L. Walteros, editors, *Learning and Intelligent Optimization, 8th International Conference, LION 8*, volume 8426 of *Lecture Notes in Computer Science*, pages 57–172. Springer, Heidelberg, Germany, 2014.

[27] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. Automatic design of evolutionary algorithms for multi-objective combinatorial optimization. In Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipič, and Jim Smith, editors, *PPSN 2014*, volume 8672 of *Lecture Notes in Computer Science*, pages 508–517. Springer, Heidelberg, Germany, 2014.

[28] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. Automatic component-wise design of multi-objective evolutionary algorithms. `http://iridia.ulb.ac.be/supp/IridiaSupp2014-010/`, 2015.

[29] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. To DE or not to DE? multi-objective differential evolution revisited from a component-wise perspective: Supplementary material. `http://iridia.ulb.ac.be/supp/IridiaSupp2015-001/`, 2015.

[30] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. To DE or not to DE? multi-objective differential evolution revisited from a component-wise perspective. In António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos A. Coello Coello, editors, *Evolutionary Multi-criterion Optimization, EMO 2015 Part I*, volume 9018 of *Lecture Notes in Computer Science*, pages 48–63. Springer, Heidelberg, Germany, 2015.

[31] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. Comparing decomposition-based and automatically component-wise designed multi-objective evolutionary algorithms. In António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos A. Coello Coello, editors, *Evolutionary Multi-criterion Optimization, EMO 2015 Part I*, volume 9018 of *Lecture Notes in Computer Science*, pages 396–410. Springer, Heidelberg, Germany, 2015.

[32] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. Automatic component-wise design of multi-objective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20(3), 2016.

[33] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. A performance assessment of tuned multi- and many-objective evolutionary algorithms. `http://iridia.ulb.ac.be/supp/IridiaSupp2015-007/`, 2016.

[34] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. Automatically designing and understanding evolutionary algorithms for multi- and many-objective optimization. `http://iridia.ulb.ac.be/supp/IridiaSupp2016-004/`, 2016. To be submitted.

[35] Mauro Birattari. *The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective*. PhD thesis, IRIDIA, École polytechnique, Université Libre de Bruxelles, Belgium, 2004.

[36] Mauro Birattari. *Tuning Metaheuristics: A Machine Learning Perspective*, volume 197 of *Studies in Computational Intelligence*. Springer, Berlin/Heidelberg, Germany, 2009.

[37] Mauro Birattari, Zhi Yuan, Prasanna Balaprakash, and Thomas Stützle. F-race and iterated F-race: An overview. In Thomas Bartz-Beielstein, Marco Chiarandini, Luís Paquete, and Mike Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 311–336. Springer, Berlin, Germany, 2010.

[38] Francesco Biscani, Dario Izzo, and Chit Hong Yam. A global optimisation toolbox for massively parallel engineering optimisation. In *Astrodynamics Tools and Techniques (ICATT 2010), 4th International Conference on*, 2010. URL `http://arxiv.org/abs/1004.3824`.

[39] S. Bleuler, Marco Laumanns, Lothar Thiele, and Eckart Zitzler. PISA – a platform and programming language independent interface for search algorithms. In Carlos M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-criterion Optimization, EMO 2003*, volume 2632 of *Lecture Notes in Computer Science*, pages 494–508. Springer, Heidelberg, Germany, 2003.

[40] Christian Blum and Andrea Roli. Hybrid metaheuristics: an introduction. In *Hybrid Metaheuristics*, pages 1–30. Springer, 2008.

[41] George E. P. Box and Norman R. Draper. *Response surfaces, mixtures, and ridge analyses*, volume 649. John Wiley & Sons, 2007.

[42] L. Bradstreet, L. Barone, L. While, S. Huband, and P. Hingston. Use of the WFG toolkit and PISA for comparison of MOEAs. In *IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making, IEEE MCDM*, pages 382–389, 2007.

[43] Karl Bringmann, Tobias Friedrich, Frank Neumann, and Markus Wagner. Approximation-guided evolutionary multi-objective optimization. In Toby Walsh, editor, *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-11)*, pages 1198–1203. IJCAI/AAAI Press, Menlo Park, CA, 2011.

[44] B. Bullnheimer, Richard F. Hartl, and Christine Strauss. A new rank-based version of the Ant System: A computational study. *Central European Journal for Operations Research and Economics*, 7(1):25–38, 1999.

[45] Edmund K. Burke, Michel Gendreau, Matthew R. Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.

[46] Shelvin Chand and Markus Wagner. Evolutionary many-objective optimization: A quick-start guide. *Surveys in Operations Research and Management Science*, 20(2):35–42, 2015.

[47] Hsinchun Chen, Roger H. L. Chiang, and Veda C. Storey. Business intelligence and analytics: From big data to big impact. *MIS quarterly*, 36(4):1165–1188, 2012.

[48] Carlos A. Coello Coello. Handling preferences in evolutionary multiobjective optimization: A survey. In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC'00)*, pages 30–37. IEEE Press, Piscataway, NJ, July 2000.

[49] Carlos A. Coello Coello and Margarita Reyes-Sierra. A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In Raúl Monroy, Gustavo Arroyo-Figueroa, Luis Enrique Sucar, and Humberto Sossa, editors, *Proceedings of MICAI*. Springer, 2004.

[50] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, NY, 2007.

[51] W. J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, New York, NY, third edition, 1999.

[52] David Corne, Joshua D. Knowles, and M. J. Oates. The Pareto envelope-based selection algorithm for multiobjective optimization. In Marc Schoenauer et al., editors, *Proceedings of PPSN-VI, Sixth International Conference on Parallel Problem Solving from Nature*, volume 1917 of *Lecture Notes in Computer Science*, pages 839–848. Springer, Heidelberg, Germany, 2000.

[53] Indraneel Das and John E. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14(1):63–69, 1997.

[54] Swagatam Das and Ponnuthurai N. Suganthan. Differential evolution: a survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1), February 2011.

[55] Kenneth A. De Jong. *Evolutionary computation: a unified approach*. MIT press, Cambridge, 2006.

[56] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, UK, 2001.

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

[58] Kalyanmoy Deb and Debayan Deb. Analysing mutation schemes for real-parameter genetic algorithms. *International Journal of Artificial Intelligence and Soft Computing*, 4(1):1–28, 2014.

[59] Kalyanmoy Deb and Sachin Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

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

[61] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable test problems for evolutionary multiobjective optimization. In Ajith Abraham, Lakhmi Jain, and Robert Goldberg, editors, *Evolutionary Multiobjective Optimization*, Advanced Information and Knowledge Processing, pages 105–145. Springer, London, UK, January 2005.

[62] Roman Denysiuk, Lino Costa, and Isabel Espírito Santo. Many-objective optimization using differential evolution with variable-wise mutation restriction. In Christian Blum and Enrique Alba, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2013*, pages 591–598. ACM Press, New York, NY, 2013.

[63] Karl F. Doerner, Richard F. Hartl, and Marc Reimann. Are COMPETants more competent for problem solving? The case of a multiple objective transportation problem. *Central European Journal for Operations Research and Economics*, 11(2):115–141, 2003.

[64] Marco Dorigo and L. M. Gambardella. Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[65] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

[66] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–41, 1996.

[67] M̲ad̲alina M. Drugan and Dirk Thierens. Path-guided mutation for stochastic Pareto local search algorithms. In Robert Schaefer, Carlos Cotta, Joanna Kolodziej, and Günther Rudolph, editors, *Parallel Problem Solving from Nature, PPSN XI*, volume 6238 of *Lecture Notes in Computer Science*, pages 485–495. Springer, Heidelberg, Germany, 2010.

[68] Jérémie Dubois-Lacoste. *Anytime Local Search for Multi-Objective Combinatorial Optimization: Design, Analysis and Automatic Configuration*. PhD thesis, IRIDIA, École polytechnique, Université Libre de Bruxelles, Belgium, 2014.

[69] Jérémie Dubois-Lacoste, Manuel López-Ibáñez, and Thomas Stützle. A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. *Computers & Operations Research*, 38(8):1219–1236, 2011.

[70] Jérémie Dubois-Lacoste, Manuel López-Ibáñez, and Thomas Stützle. Automatic configuration of state-of-the-art multi-objective optimizers using the TP+PLS framework. In Natalio Krasnogor and Pier Luca Lanzi, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011*, pages 2019–2026. ACM Press, New York, NY, 2011. ISBN 978-1-4503-0557-0.

[71] Jérémie Dubois-Lacoste, Manuel López-Ibáñez, and Thomas Stützle. Pareto local search algorithms for anytime bi-objective optimization. In Jin-Kao Hao and M. Middendorf, editors, *Proceedings of EvoCOP 2012 – 12th European Conference on Evolutionary Computation in Combinatorial Optimization*, volume 7245 of *Lecture Notes in Computer Science*, pages 206–217. Springer, Heidelberg, Germany, 2012.

[72] Jérémie Dubois-Lacoste, Manuel López-Ibáñez, and Thomas Stützle. Combining two search paradigms for multi-objective optimization: Two-Phase and Pareto local search. In E-G. Talbi, editor, *Hybrid Metaheuristics*, volume 434 of *Studies in Computational Intelligence*, pages 97–117. Springer Verlag, 2013.

[73] Juan J. Durillo, Antonio J. Nebro, Francisco Luna, and Enrique Alba. On the effect of the steady-state selection scheme in multi-objective genetic algorithms. In Matthias Ehrgott, Carlos M. Fonseca, Xavier Gandibleux, Jin-Kao Hao, and Marc Sevaux, editors, *Evolutionary Multi-criterion Optimization, EMO 2009*, volume 5467 of *Lecture Notes in Computer Science*, pages 183–197. Springer, Heidelberg, Germany, 2009.

[74] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, 1995.

[75] Agoston E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[76] Chris Fawcett and Holger H. Hoos. Analysing differences between algorithm configurations through ablation. In *Proceedings of MIC 2013, the 10th Metaheuristics International Conference*, pages 123–132, 2013.

[77] Chris Fawcett and Holger H. Hoos. Analysing differences between algorithm configurations through ablation. *Journal of Heuristics*, pages 1–28, 2015.

[78] T. A. Feo and Mauricio G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–113, 1995.

[79] Peter J. Fleming, Robin C. Purshouse, and Robert J. Lygoe. Many-objective optimization: An engineering design perspective. In Carlos A. Coello Coello, A. H. Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-criterion Optimization, EMO 2005*, volume 3410 of *Lecture Notes in Computer Science*, pages 14–32. Springer, Heidelberg, Germany, 2005.

[80] Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In Stephanie Forrest, editor, *ICGA*, pages 416–423. Morgan Kaufmann Publishers, 1993. ISBN 1-55860-299-2.

[81] Carlos M. Fonseca, Luís Paquete, and Manuel López-Ibáñez. An improved dimension-sweep algorithm for the hypervolume indicator. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)*, pages 1157–1163. IEEE Press, Piscataway, NJ, July 2006.

[82] Carlos M. Fonseca, Andreia P. Guerreiro, Manuel López-Ibáñez, and Luís Paquete. On the computation of the empirical attainment function. In R. H. C. Takahashi et al., editors, *Evolutionary Multi-criterion Optimization, EMO 2011*, volume 6576 of *Lecture Notes in Computer Science*, pages 106–120. Springer, Heidelberg, Germany, 2011.

[83] Jose M. Framinan, Rainer Leisten, and Rubén Ruiz. *Manufacturing Scheduling Systems: An Integrated View on Models, Methods, and Tools*. Springer, New York, NY, 2014.

[84] D. Fudenberg and J Tirole. *Game Theory*. MIT Press, 1983.

[85] Carlos García-Martínez, Oscar Cordón, and Francisco Herrera. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. *European Journal of Operational Research*, 180(1):116–148, 2007.

[86] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman & Co, San Francisco, CA, 1979.

[87] M. R. Garey, David S. Johnson, and R. Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1:117–129, 1976.

[88] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Torsten Schaub, Marius Thomas Schneider, and Stefan Ziller. A portfolio solver for answer set programming: Preliminary report. In Pedro Calabar and Tran Cao Son, editors, *Logic Programming and Nonmonotonic Reasoning*, volume 8148 of *Lecture Notes in Artificial Intelligence*, pages 352–357. Springer, Heidelberg, Germany, 2013.

[89] Martin Josef Geiger. Decision support for multi-objective flow shop scheduling by the Pareto iterated local search methodology. *Computers and Industrial Engineering*, 61(3):805–812, 2011.

[90] Michel Gendreau and J.-Y. Potvin. Tabu search. In *Handbook of Metaheuristics* Gendreau and Potvin [91], pages 41–59.

[91] Michel Gendreau and J.-Y. Potvin, editors. *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*. Springer, New York, NY, 2 edition, 2010.

[92] Fred Glover. Tabu search – Part I. *INFORMS Journal on Computing*, 1(3):190–206, 1989.

[93] Fred Glover. Tabu search – Part II. *INFORMS Journal on Computing*, 2(1):4–32, 1990.

[94] Fred Glover. A template for scatter search and path relinking. In Jin-Kao Hao, Evelyne Lutton, Edmund M. A. Ronald, Marc Schoenauer, and Dominique Snyers, editors, *Artificial Evolution*, volume 1363 of *Lecture Notes in Computer Science*, pages 1–51. Springer, Heidelberg, Germany, 1998.

[95] Fred Glover, Manuel Laguna, and Rafa Martí. Scatter search and path relinking: Advances and applications. In Fred Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 1–35. Kluwer Academic Publishers, Norwell, MA, 2002.

[96] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Boston, MA, USA, 1989.

[97] Viviane Grunert da Fonseca, Carlos M. Fonseca, and Andreia O. Hall. Inferential performance assessment of stochastic optimisers and the attainment function. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *Evolutionary Multi-criterion Optimization, EMO 2001*, volume 1993 of *Lecture Notes in Computer Science*, pages 213–225. Springer, Heidelberg, Germany, 2001.

[98] David Hadka and Patrick Reed. Diagnostic assessment of search controls and failure modes in many-objective evolutionary optimization. *Evolutionary Computation*, 20(3):423–452, 2012.

[99] David Hadka, Patrick Reed, and T. W. Simpson. Diagnostic assessment of the Borg MOEA for many-objective product family design problems. In *Proceedings of the 2012 Congress on Evolutionary Computation (CEC'12)*, pages 1–10, Piscataway, NJ, 2012. IEEE Press.

[100] Prabhat Hajela and C-Y Lin. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 4(2):99–107, 1992.

[101] Nikolaus Hansen. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer, 2006.

[102] Nikolaus Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[103] Pierre Hansen and Nenad Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.

[104] Pierre Hansen, Nenad Mladenović, Jack Brimberg, and José A. Moreno Pérez. Variable Neighborhood Search. In Gendreau and Potvin [91], pages 61–86.

[105] Daniel P Heyman and Matthew J Sobel. *Stochastic models in operations research: stochastic optimization*, volume 2. Courier Corporation, 2003.

[106] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[107] Holger H. Hoos. Automated algorithm configuration and parameter tuning. In Y. Hamadi, E. Monfroy, and F. Saubion, editors, *Autonomous Search*, pages 37–71. Springer, Berlin, Germany, 2012.

[108] Holger H. Hoos. Programming by optimization. *Communications of the ACM*, 55(2):70–80, February 2012.

[109] Holger H. Hoos and Thomas Stützle. *Stochastic Local Search: Foundations and Applications*. Elsevier, Amsterdam, The Netherlands, 2004.

[110] Holger H. Hoos and Thomas Stützle. *Stochastic Local Search—Foundations and Applications*. Morgan Kaufmann Publishers, San Francisco, CA, 2005.

[111] J. Horn, N. Nafpliotis, and D.E. Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. In *Proceedings of the 1994 World Congress on Computational Intelligence (WCCI 1994)*, volume 1, pages 82–87, Piscataway, NJ, Jun 1994. IEEE Press.

[112] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.

[113] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, October 2009.

[114] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Automated configuration of mixed integer programming solvers. In A. Lodi, M. Milano, and P. Toth, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 7th International Conference, CPAIOR 2010*, volume 6140 of *Lecture Notes in Computer Science*, pages 186–202. Springer, Heidelberg, Germany, 2010.

[115] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization, 5th International Conference, LION 5*, volume 6683 of *Lecture Notes in Computer Science*, pages 507–523. Springer, Heidelberg, Germany, 2011.

[116] Frank Hutter, Manuel López-Ibáñez, Chris Fawcett, Marius Thomas Lindauer, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. AClib: a benchmark library for algorithm configuration. In Panos M. Pardalos, Mauricio G. C. Resende, Chrysafis Vogiatzis, and Jose L. Walteros, editors, *Learning and Intelligent Optimization, 8th International Conference, LION 8*, volume 8426 of *Lecture Notes in Computer Science*, pages 36–40. Springer, Heidelberg, Germany, 2014.

[117] Frank Hutter, Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, 206:79–111, 2014.

[118] IBM. ILOG CPLEX optimizer. `http://www.ibm.com/software/integration/optimization/cplex-optimizer/`.

[119] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, 2007.

[120] C. Igel, V. Heidrich-Meisner, and T. Glasmachers. Shark. *Journal of Machine Learning Research*, 9:993–996, June 2008. URL `http://www.jmlr.org/papers/volume9/igel08a/igel08a.pdf`.

[121] S. Iredi, D. Merkle, and M. Middendorf. Bi-criterion optimization with multi colony ant algorithms. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *Evolutionary Multi-criterion Optimization, EMO 2001*, volume 1993 of *Lecture Notes in Computer Science*, pages 359–372. Springer, Heidelberg, Germany, 2001.

[122] H. Ishibuchi and T. Murata. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics – Part C*, 28(3): 392–403, 1998.

[123] H. Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, and Yusuke Nojima. Modified distance calculation in generational distance and inverted generational distance. In António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos A. Coello Coello, editors, *Evolutionary Multi-criterion Optimization, EMO 2015 Part I*, volume 9018 of *Lecture Notes in Computer Science*, pages 110–125. Springer, Heidelberg, Germany, 2015.

[124] Hughes Evan J. Multiple single objective Pareto sampling. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, volume 4, pages 2678–2684, Piscataway, NJ, December 2003. IEEE Press.

[125] Hughes Evan J. MSOPS-II: A general-purpose many-objective optimiser. In *Proceedings of the 2007 Congress on Evolutionary Computation (CEC 2007)*, pages 3944–3951, Piscataway, NJ, 2007. IEEE Press.

[126] Hughes Evan J. Many-objective directed evolutionary line search. In Natalio Krasnogor and Pier Luca Lanzi, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011*, pages 761–768. ACM Press, New York, NY, 2011. ISBN 978-1-4503-0557-0.

[127] Larry W. Jacobs and Michael J. Brusco. A local search heuristic for large set-covering problems. *Naval Research Logistics*, 42(7):1129–1140, 1995.

[128] Andrzej Jaszkiewicz. Genetic local search for multi-objective combinatorial optimization. *European Journal of Operational Research*, 137(1):50–71, 2002.

[129] S. Jiang, Y. S. Ong, J. Zhang, and L. Feng. Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE Transactions on Cybernetics*, 44(12):2391–2404, 2014.

[130] Serdar Kadioglu, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. ISAC: Instance-specific algorithm configuration. In H. Coelho, R. Studer, and M. Wooldridge, editors, *Proceedings of the 19th European Conference on Artificial Intelligence*, pages 751–756. IOS Press, 2010.

[131] Dervis Karaboga and Bahriye Akay. A survey: Algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*, 31(1–4):61–85, 2009.

[132] V. Khare, X. Yao, and Kalyanmoy Deb. Performance scaling of multi-objective evolutionary algorithms. In Carlos M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-criterion Optimization, EMO 2003*, volume 2632 of *Lecture Notes in Computer Science*, pages 376–390. Springer, Heidelberg, Germany, 2003.

[133] A. R. KhudaBukhsh, Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown. SATenstein: Automatically building local search SAT solvers from components. In Craig Boutilier, editor, *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 517–524. AAAI Press, Menlo Park, CA, 2009.

[134] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[135] Joshua D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization.* PhD thesis, University of Reading, UK, 2002.

[136] Joshua D. Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

[137] Joshua D. Knowles and David Corne. Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[138] Joshua D. Knowles and David Corne. Bounded Pareto archiving: Theory and practice. In Xavier Gandibleux, Marc Sevaux, Kenneth Sörensen, and Vincent T'kindt, editors, *Metaheuristics for Multiobjective Optimisation*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*, pages 39–64. Springer, Berlin, Germany, 2004.

[139] Joshua D. Knowles, David Corne, and Mark Fleischer. Bounded archiving using the Lebesgue measure. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, volume 4, pages 2490–2497. IEEE Press, Piscataway, NJ, December 2003.

[140] Joshua D. Knowles, Lothar Thiele, and Eckart Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK-Report 214, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zürich, Switzerland, February 2006. Revised version.

[141] J. Koza. *Genetic Programming: On the Programming of Computers By the Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

[142] William H. Kruskal and Judith M. Tanur. *Linear Hypotheses*, volume 1. Free Press, 1978.

[143] S. Kukkonen and J. Lampinen. GDE3: the third evolution step of generalized differential evolution. In *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, pages 443–450. IEEE Press, Piscataway, NJ, September 2005.

[144] Pedro Larrañaga and Jose A. Lozano. *Estimation of distribution algorithms: A new tool for evolutionary computation*. Kluwer Academic Publishers, Boston, 2002.

[145] Marco Laumanns and R. Zenklusen. Stochastic convergence of random search methods to fixed size Pareto front approximations. *European Journal of Operational Research*, 213(2):414–421, 2011.

[146] Marco Laumanns, Eckart Zitzler, and Lothar Thiele. A unified model for multi-objective evolutionary algorithms with elitism. In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC'00)*, pages 46–53, Piscataway, NJ, July 2000. IEEE Press.

[147] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 10(3):263–282, 2002.

[148] Bingdong Li, Jinlong Li, Ke Tang, and Xin Yao. An improved two archive algorithm for many-objective optimization. In *Proceedings of the 2014 Congress on Evolutionary Computation (CEC 2014)*, pages 2869–2876, Piscataway, NJ, 2014. IEEE Press.

[149] Bingdong Li, Jinlong Li, Ke Tang, and Xin Yao. Many-objective evolutionary algorithms: A survey. *ACM Computing Surveys*, 48(1):1–35, 2015.

[150] Hui Li and Qingfu Zhang. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302, 2009.

[151] Miqing Li, Shengxiang Yang, Xiaohui Liu, and Ruimin Shen. A comparative study on evolutionary algorithms for many-objective optimization. In Robin C. Purshouse, Peter J. Fleming, Carlos M. Fonseca, Salvatore Greco, and Jane Shaw, editors, *Evolutionary Multi-criterion Optimization, EMO 2013*, volume 7811 of *Lecture Notes in Computer Science*, pages 261–275. Springer, Heidelberg, Germany, 2013. ISBN 978-3-642-37139-4.

[152] Arnaud Liefooghe, Jérémie Humeau, Salma Mesmoudi, Laetitia Jourdan, and E-G. Talbi. On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. *Journal of Heuristics*, 18(2):317–352, 2011.

[153] Arnaud Liefooghe, Laetitia Jourdan, and E-G. Talbi. A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. *European Journal of Operational Research*, 209(2):104–112, 2011.

[154] Marius Thomas Lindauer, Holger H. Hoos, Frank Hutter, and Torsten Schaub. AutoFolio: Algorithm configuration for algorithm selection. In Blai Bonet and Sven Koenig, editors, *AAAI*. AAAI Press, 2015.

[155] Marius Thomas Lindauer, Holger H. Hoos, Frank Hutter, and Torsten Schaub. AutoFolio: an automatically configured algorithm selector. *Journal of Artificial Intelligence Research*, pages 745–778, 2015.

[156] Manuel López-Ibáñez and Thomas Stützle. The impact of design choices of multi-objective ant colony optimization algorithms on performance: An experimental study on the biobjective TSP. In Martin Pelikan and Jürgen Branke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2010*, pages 71–78. ACM Press, New York, NY, 2010.

[157] Manuel López-Ibáñez and Thomas Stützle. The automatic design of multi-objective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 16(6):861–875, 2012.

[158] Manuel López-Ibáñez, Luís Paquete, and Thomas Stützle. Exploratory analysis of stochastic local search algorithms in biobjective optimization. In Thomas Bartz-Beielstein, Marco Chiarandini, Luís Paquete, and Mike Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 209–222. Springer, Berlin, Germany, 2010.

[159] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011. URL `http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2011-004.pdf`.

[160] Manuel López-Ibáñez, Joshua D. Knowles, and Marco Laumanns. On sequential online archiving of objective vectors. In R. H. C. Takahashi et al., editors, *Evolutionary Multi-criterion Optimization, EMO 2011*, volume 6576 of *Lecture Notes in Computer Science*, pages 46–60. Springer, Heidelberg, Germany, 2011.

[161] Antonio López Jaimes, Carlos A. Coello Coello, and Debrup Chakraborty. Objective reduction using a feature selection technique. In Conor Ryan, editor, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2008*, pages 673–680. ACM Press, New York, NY, 2008.

[162] T. Lust and Jacques Teghem. The multiobjective traveling salesman problem: A survey and a new approach. In Carlos A. Coello Coello, Clarisse Dhaenens, and Laetitia Jourdan, editors, *Advances in Multi-Objective Nature Inspired Computing*, volume 272 of *Studies in Computational Intelligence*, pages 119–141. Springer, 2010.

[163] T. Lust and Jacques Teghem. The multiobjective multidimensional knapsack problem: a survey and a new approach. *Arxiv preprint arXiv:1007.4063*, 2010.

[164] T. Lust and Jacques Teghem. The multiobjective multidimensional knapsack problem: a survey and a new approach. *International Transactions in Operational Research*, 19(4):495–520, 2012.

[165] W. J. Conover M. D. McKay, R. J. Beckman. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[166] Nateri K Madavan. Multiobjective optimization using a Pareto differential evolution approach. In *Proceedings of the 2002 World Congress on Computational Intelligence (WCCI 2002)*, pages 1145–1150, Piscataway, NJ, 2002. IEEE Press.

[167] Marie-Eléonore Marmion, Franco Mascia, Manuel López-Ibáñez, and Thomas Stützle. Automatic design of hybrid stochastic local search algorithms. In María J. Blesa, Christian Blum, Paola Festa, Andrea Roli, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 7919 of *Lecture Notes in Computer Science*, pages 144–158. Springer, Heidelberg, Germany, 2013. ISBN 978-3-642-38515-5.

[168] O. Maron and A. W. Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Research*, 11(1–5):193–225, 1997.

[169] E. Q. V. Martins. On a multicritera shortest path problem. *European Journal of Operational Research*, 16:236–245, 1984.

[170] Franco Mascia, Manuel López-Ibáñez, Jérémie Dubois-Lacoste, and Thomas Stützle. From grammars to parameters: Automatic iterated greedy design for the permutation flow-shop problem with weighted tardiness. In P. Pardalos and G. Nicosia, editors, *Learning and Intelligent Optimization, 7th International Conference, LION 7*, volume 7997 of *Lecture Notes in Computer Science*, pages 321–334. Springer, Heidelberg, Germany, 2013.

[171] Franco Mascia, Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Marie-Eléonore Marmion, and Thomas Stützle. Algorithm comparison by automatically configurable stochastic local search frameworks: A case study using flow-shop scheduling problems. In María J. Blesa, Christian Blum, and Stefan Voß, editors, *Hybrid Metaheuristics*, volume 8457 of *Lecture Notes in Computer Science*, pages 30–44. Springer, Heidelberg, Germany, 2014. ISBN 978-3-319-07643-0.

[172] Franco Mascia, Manuel López-Ibáñez, Jérémie Dubois-Lacoste, and Thomas Stützle. Grammar-based generation of stochastic local search heuristics through automatic algorithm configuration tools. *Computers & Operations Research*, 51:190–199, 2014.

[173] Robert I. Mckay, Nguyen Xuan Hoai, Peter Alexander Whigham, Yin Shan, and Michael O'Neill. Grammar-based genetic programming: A survey. *Genetic Programming and Evolvable Machines*, 11(3-4):365–396, September 2010.

[174] A. Menchaca-Mendez and Carlos A. Coello Coello. GDE-MOEA: A new MOEA based on the generational distance indicator and ϵ-dominance. In *Proceedings of the 2015 Congress on Evolutionary Computation (CEC 2015)*, pages 947–955, Piscataway, NJ, 2015. IEEE Press.

[175] Adriana Menchaca-Mendez and Carlos A. Coello Coello. GD-MOEA: A new multi-objective evolutionary algorithm based on the generational distance indicator. In António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos A. Coello Coello, editors, *Evolutionary Multi-criterion Optimization, EMO 2015 Part I*, volume 9018 of *Lecture Notes in Computer Science*, pages 156–170. Springer, Heidelberg, Germany, 2015.

[176] E. Mezura-Montes, M. Reyes-Sierra, and Carlos A. Coello Coello. Multi-objective optimization using differential evolution: a survey of the state-of-the-art. In Uday K. Chakraborty, editor, *Advances in differential evolution*, pages 173–196. Springer, Heidelberg, Germany, 2008.

[177] Kaisa Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

[178] Gerardo Minella, Rubén Ruiz, and M. Ciavotta. A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. *INFORMS Journal on Computing*, 20(3):451–471, 2008.

[179] Atefeh Moghaddam, Farouk Yalaoui, and Lionel Amodeo. Lorenz versus Pareto dominance in a single machine scheduling problem with rejection. In R. H. C. Takahashi et al., editors, *Evolutionary Multi-criterion Optimization, EMO 2011*, volume 6576 of *Lecture Notes in Computer Science*, pages 520–534. Springer, Heidelberg, Germany, 2011.

[180] Gilberto Montibeller and Hugo Yoshizaki. A framework for locating logistic facilities with multicriteria decision analysis. In R. H. C. Takahashi et al., editors, *Evolutionary Multi-criterion Optimization, EMO 2011*, volume 6576 of *Lecture Notes in Computer Science*, pages 505–519. Springer, Heidelberg, Germany, 2011.

[181] R. Nagy, M. Suciu, and D. Dumitrescu. Exploring Lorenz dominance. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2012 14th International Symposium on*, pages 254–259, 2012.

[182] V. Nannen and Agoston E. Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. In Manuela M. Veloso, editor, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 975–980. AAAI Press, Menlo Park, CA, 2007.

[183] Krzysztof Nowak, Marcus Märtens, and Dario Izzo. Empirical performance of the approximation of the least hypervolume contributor. In Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipič, and Jim Smith, editors, *PPSN 2014*, volume 8672 of *Lecture Notes in Computer Science*, pages 662–671. Springer, Heidelberg, Germany, 2014.

[184] Eoin O'Mahony, Emmanuel Hebrard, Alan Holland, Conor Nugent, and Barry O'Sullivan. Using case-based reasoning in an algorithm portfolio for constraint solving. In *Irish Conference on Artificial Intelligence and Cognitive Science*, pages 210–216, 2008.

[185] Michael O'Neill and Conor Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358, 2001.

[186] Luís Paquete and Thomas Stützle. A two-phase local search for the biobjective traveling salesman problem. In Carlos M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-criterion Optimization, EMO 2003*, volume 2632 of *Lecture Notes in Computer Science*, pages 479–493. Springer, Heidelberg, Germany, 2003.

[187] Luís Paquete and Thomas Stützle. Stochastic local search algorithms for multiobjective combinatorial optimization: A review. In T. F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*, pages 29–1—29–15. Chapman & Hall/CRC, Boca Raton, FL, 2007.

[188] Luís Paquete and Thomas Stützle. Design and analysis of stochastic local search for the multiobjective traveling salesman problem. *Computers & Operations Research*, 36(9):2619–2631, 2009.

[189] Luís Paquete, Marco Chiarandini, and Thomas Stützle. Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In Xavier Gandibleux, Marc Sevaux, Kenneth Sörensen, and Vincent T'kindt, editors, *Metaheuristics for Multiobjective Optimisation*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*, pages 177–200. Springer, Berlin, Germany, 2004.

[190] Leslie Pérez Cáceres, Manuel López-Ibáñez, and Thomas Stützle. An analysis of parameters of irace. In *Proceedings of EvoCOP 2014 – 14th European Conference on Evolutionary Computation in Combinatorial Optimization*, volume 8600 of *Lecture Notes in Computer Science*, pages 37–48. Springer, Heidelberg, Germany, 2014.

[191] David Pisinger and Stefan Ropke. Large neighborhood search. In Gendreau and Potvin [91], pages 399–419.

[192] Kata Praditwong and Xin Yao. A new multi-objective evolutionary optimisation algorithm: the two-archive algorithm. In *Computational intelligence and security, 2006 international conference on*, volume 1, pages 286–291. IEEE, 2006.

[193] Kenneth Price, Rainer M Storn, and Jouni A Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, New York, NY, 2005.

[194] Luca Pulina and Armando Tacchella. A self-adaptive multi-engine solver for quantified Boolean formulas. *Constraints*, 14(1):80–116, 2009.

[195] Robin C. Purshouse and Peter J. Fleming. On the evolutionary optimization of many conflicting objectives. *IEEE Transactions on Evolutionary Computation*, 11, 2007.

[196] Andreea Radulescu, Manuel López-Ibáñez, and Thomas Stützle. Automatically improving the anytime behaviour of multiobjective evolutionary algorithms. In Robin C. Purshouse, Peter J. Fleming, Carlos M. Fonseca, Salvatore Greco, and Jane Shaw, editors, *Evolutionary Multi-criterion Optimization, EMO 2013*, volume 7811 of *Lecture Notes in Computer Science*, pages 825–840. Springer, Heidelberg, Germany, 2013. ISBN 978-3-642-37139-4.

[197] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der bilogischen Evolution*. PhD thesis, Department of Process Engineering, Technical University of Berlin, 1971.

[198] Colin Reeves. Genetic algorithms. In Gendreau and Potvin [91], chapter 5, pages 109–140.

[199] M . Reyes-Sierra and Carlos A. Coello Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, 2(3):287–308, 2006.

[200] John R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

[201] Tea Robič and Bogdan Filipič. DEMO: Differential evolution for multiobjective optimization. In Carlos A. Coello Coello, A. H. Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-criterion Optimization, EMO 2005*, volume 3410 of *Lecture Notes in Computer Science*, pages 520–533. Springer, Heidelberg, Germany, 2005.

[202] Cynthia A Rodríguez Villalobos and Carlos A. Coello Coello. A new multi-objective evolutionary algorithm based on a performance assessment indicator. In Terence Soule and Jason H. Moore, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2012*, pages 505–512, New York, NY, 2012. ACM, ACM Press.

[203] Dhish Kumar Saxena, Joao A Duro, Anish Tiwari, Kaushik Deb, and Qingfu Zhang. Objective reduction in many-objective optimization: Linear and nonlinear algorithms. *IEEE Transactions on Evolutionary Computation*, 17(1):77–99, 2013.

[204] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In John J. Grefenstette, editor, *ICGA*, pages 93–100. Lawrence Erlbaum Associates, 1985. ISBN 0-8058-0426-9.

[205] Gerhard Schrimpf, Johannes Schneider, Hermann Stamm-Wilbrandt, and Gunter Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139 – 171, 2000.

[206] O. Schutze, X. Esquivel, A. Lara, and Carlos A. Coello Coello. Using the averaged Hausdorff distance as a performance measure in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 16(4):504–522, 2012.

[207] Haitham Seada and Kalyanmoy Deb. U-NSGA-III: A unified evolutionary optimization procedure for single, multiple, and many objectives: Proof-of-principle results. In António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos A. Coello Coello, editors, *Evolutionary Multi-criterion Optimization, EMO 2015 Part I*, volume 9018 of *Lecture Notes in Computer Science*, pages 34–49. Springer, Heidelberg, Germany, 2015.

[208] Jendrik Seipp, Silvan Sievers, Malte Helmert, and Frank Hutter. Automatic configuration of sequential planning portfolios. In Blai Bonet and Sven Koenig, editors, *AAAI*, pages 3364–3370. AAAI Press, 2015.

[209] P. Serafini. Some considerations about computational complexity for multiobjective combinatorial problems. In J. Jahn and W. Krabs, editors, *Recent Advances and Historical Development of Vector Optimization*, volume 294 of *Lecture Notes in Economics and Mathematical Systems*, pages 222–231. Springer, Berlin, Germany, 1986.

[210] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In Michael Maher and Jean-Francois Puget, editors, *Principles and Practice of Constraint Programming, CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer, Heidelberg, Germany, 1998.

[211] S. K. Smit and Agoston E. Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *Proceedings of the 2009 Congress on Evolutionary Computation (CEC 2009)*, pages 399–406. IEEE Press, Piscataway, NJ, 2009.

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

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

[214] Thomas Stützle. ACOTSP: A software package of various ant colony optimization algorithms applied to the symmetric traveling salesman problem, 2002. URL `http://www.aco-metaheuristic.org/aco-code/`.

[215] Thomas Stützle and Holger H. Hoos. $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[216] Kiyoharu Tagawa, Hidehito Shimizu, and Hiroyuki Nakamura. Indicator-based differential evolution using exclusive hypervolume approximation and parallelization for multi-core processors. In Natalio Krasnogor and Pier Luca Lanzi, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011*, pages 657–664. ACM Press, New York, NY, 2011. ISBN 978-1-4503-0557-0.

[217] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy, editors, *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013*, pages 847–855. ACM Press, New York, NY, 2013.

[218] Tea Tušar. Design of an algorithm for multiobjective optimization with differential evolution. M.sc. thesis, Faculty of Computer and Information Science, University of Ljubljana, 2007.

[219] Tea Tušar and Bogdan Filipič. Differential evolution versus genetic algorithms in multiobjective optimization. In S. Obayashi et al., editors, *Evolutionary Multi-criterion Optimization, EMO 2007*, volume 4403 of *Lecture Notes in Computer Science*, pages 257–271. Springer, Heidelberg, Germany, 2007.

[220] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, 8(2):125–147, 2000.

[221] Thomas Voß, Nikolaus Hansen, and Christian Igel. Improved step size adaptation for the MO-CMA-ES. In Martin Pelikan and Jürgen Branke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2010*, pages 487–494. ACM Press, New York, NY, 2010.

[222] Markus Wagner and Frank Neumann. A fast approximation-guided evolutionary multi-objective algorithm. In Sara Silva and Anna I. Esparcia-Alcázar, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015*, pages 687–694. ACM Press, New York, NY, 2015.

[223] L. While and L. Bradstreet. Applying the WFG algorithm to calculate incremental hypervolumes. In *Proceedings of the 2012 Congress on Evolutionary Computation (CEC'12)*, pages 1–8, Piscataway, NJ, 2012. IEEE Press.

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

[225] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla: portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606, June 2008.

[226] Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown. Hydra: Automatically configuring algorithms for portfolio-based selection. In Maria Fox and David Poole, editors, *AAAI*. AAAI Press, 2010.

[227] S. Yang, M. Li, X. Liu, and J. Zheng. A grid-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 17(5):721–736, 2013.

[228] Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[229] Qingfu Zhang and Ponnuthurai N. Suganthan. Special session on performance assessment of multiobjective optimization algorithms/CEC'09 MOEA competition. `http://dces.essex.ac.uk/staff/qzhang/moeacompetition09.htm`, 2009.

[230] Qingfu Zhang, Wudong Liu, and Hui Li. The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. In *Proceedings of the 2009 Congress on Evolutionary Computation (CEC 2009)*, pages 203–208, Piscataway, NJ, 2009. IEEE Press.

[231] Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, ETH Zürich, Switzerland, 1999.

[232] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In Xin Yao et al., editors, *Proceedings of PPSN-VIII, Eigth International Conference on Parallel Problem Solving from Nature*, volume 3242 of *Lecture Notes in Computer Science*, pages 832–842. Springer, Heidelberg, Germany, 2004.

[233] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto evolutionary algorithm. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[234] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In K. C. Giannakoglou, D. T. Tsahalis, J. Periaux, K. D. Papaliliou, and T. Fogarty, editors, *Evolutionary Methods for Design, Optimisation and Control*, pages 95–100. CIMNE, Barcelona, Spain, 2002.

[235] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

[236] Eckart Zitzler, Lothar Thiele, and Johannes Bader. On set-based multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 14(1):58–79, 2010.