Solving Mixed Pareto-Lexicographic Multi-Objective Optimization Problems: The Case of Priority Chains

Leonardo Lai, Lorenzo Fiaschi, Marco Cococcioni*

Department of Information Engineering, Largo Lucio Lazzarino, 1 – 56123 Pisa, ITALY

Abstract

This paper introduces a new class of optimization problems, called Mixed Pareto-Lexicographic Multi-objective Optimization Problems (MPL-MOPs), to provide a suitable model for scenarios where some objectives have priority over some others. Specifically, this work focuses on a relevant subclass of MPL-MOPs, namely problems involving Pareto optimization of two or more *priority chains*. A priority chain (PC) is a sequence of objectives lexicographically ordered by importance. After examining the main features of those problems, named PC-MPL-MOPs, we propose an innovative approach to deal with them, built upon the Grossone Methodology, a recent theory which enables handling the priority in an elegant and powerful way. The most interesting aspect of this technique is the possibility to seamlessly embed it in any existing evolutionary algorithm, without altering its logical structure. In order to provide concrete examples, we implemented it on top of the well-known NSGA-II and MOEA/D algorithms, calling these new generalized versions *PC-NSGA-II* and *PC-MOEA/D*, respectively. In the second part of this article, we test the strength of our strategy in solving multi- and even many-objective problems with priority chains, comparing it against the results achieved by standard priority-based and non-priority-based approaches. Experiments show that our algorithms are generally able to produce more solutions and of higher quality.

Keywords: Pareto Optimization, Lexicographic Optimization, Evolutionary Computation, Genetic Algorithms, Numerical Infinitesimals, Grossone Infinity Computing

1. Introduction

The optimization of two or more conflicting objectives is useful in many real-world scenarios; such problems, commonly referred to as *multi-objective*

*Corresponding author

Email addresses: leonardo.lai@live.com (Leonardo Lai), lorenzo.fiaschi@phd.unipi.it (Lorenzo Fiaschi), marco.cococcioni@unipi.it (Marco Cococcioni)

optimization problems (MOPs), are typically stated as follows:

$$\min \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{bmatrix} \quad s.t. \quad x \in \Omega$$

where Ω represents an arbitrary input space. The search for Pareto optimal solutions in MOPs is not a trivial task, and has been addressed over time using different strategies, from mathematical-programming methods to evolutionary multi-objective optimization *(EMO)* algorithms. Being population-based, the latter are able to produce a large number of solutions at the same time, whereas other approaches require the same procedure to be repeated over and over. In addition, evolutionary algorithms proved to be effective in solving hard problems too, for instance those with a large number of decision variables or several local optima. Thanks to these properties, EMO has become the standard in its field: noteworthy examples are *NSGA-II*, *SPEA2* and *MOEA/D* [1, 2, 3]. Recently, the research focus has shifted towards problems with a large number of objectives, known as *many-objective optimization problems (MaOPs)* [4]. This definition usually applies to any problem with more than three objectives, but it is common to have ten or more. It has been observed that traditional EMO algorithms designed for MOPs are often inadequate when facing MaOPs, for a variety of reasons. First, when the number of objectives grows, a larger fraction of the population becomes nondominated, slowing down the optimization process significantly. This is a very serious issue, which has to do with the ineffectiveness of the Pareto dominance definition; alternatives have been proposed, for instance in [5]. Second, the conflict between convergence and diversity assessment exacerbates in a large-dimensional space, making it harder to find a balance between these two pressures. Furthermore, the recombination operation may be inefficient, calculations and simulations become computationally expensive, results are harder to understand and visualize.

In recent years all these difficulties have led to the development of sophisticated methodologies specific for MaOPs, including some evolutionary algorithms that are often an overhaul of their corresponding one for MOPs. A remarkable example is *NSGA-III* [6], which enhances the fundamental ideas of NSGA-II to better deal with many-objective problems. Although the aforementioned issues can be mitigated by exploiting new techniques specifically designed for MaOPs, they cannot, by their very nature, be eliminated altogether. The goal of current research is, on the one hand, to improve and refine the existing algorithms, on the other to identify innovative ways to address these cases. Anyway, it is crucial not to forget that most of the optimization problems come from concrete real-world situations, therefore a feasible reverse strategy could be returning to the root of the problem, attempting to extract additional information from the context and, eventually, embedding this new knowledge into the mathematical model of the problem. For instance, in realistic MOPs and MaOPs, it is likely that some objectives are more important than others or, at least, have a stronger impact in the subsequent decision process; therefore, their optimization should

be prioritized. One way to accomplish this is by *weighted sum scalarization*, which reformulates the problem as a single-objective one:

$$\min \sum_{i=1}^{m} w_i f_i(x) \qquad w_i \in \mathbb{R}$$

Weights are proportional to the importance. Yet there is a drawback: it requires a priori knowledge or determination of such weights, which is not a trivial task. Moreover, small changes to the parameters lead to distinct problems, with possibly different solutions [7]. Finally, the use of very large (or very small) weights worsens the numerical stability of the algorithm.
In some scenarios, on the other hand, it is possible to rank the objectives *lexicographically*: the objective that is ranked first holds absolute priority over the second one, which in turn has precedence over the third, and so on. Lexicographic problems are not uncommon in real situations, concrete examples can be found in [8, 9]. They are typically addressed by solving an ordered sequence of single-objective problems: first the optimal values for the most important objective are found, then a constraint is added to force the solutions of the next subproblems to stay in the optimal set of the primary objective. The same procedure is then repeated for each objective, in order of importance, every time adding a further constraint. The weaknesses of this strategy, called *preemptive method*, are the growing size of the set of constraints, which makes it increasingly harder to locate feasible solutions, and the arbitrary structure of the constraints themselves, non-trivial to handle.

This article aims to describe a novel approach, which goes beyond the lexicographic method, to deal with multi- and many-objective optimization problems where lexicographic priorities exist between some objectives. It combines the effectiveness of evolutionary algorithms (specifically we chose NSGA-II and MOEA/D because of their popularity) with the flexibility of an emerging, yet powerful mathematical theory which makes it possible to work with infinite and infinitesimal quantities: the *Grossone Methodology* by Y. D. Sergeyev [10]. Our approach does not decompose the problem into smaller and prioritized subproblems, but deals with it in one fell swoop, hence bypassing the aforementioned difficulty of adding constraints. The same strategy has been already adopted by Cococcioni et al. to extend the well-known simplex algorithm [11, 12].

In Section 2 we first provide a mathematical description of the types of problems that we aim to solve, then point out the weaknesses of a standard approach in Section 3. A brief insight into the Grossone Methodology is given in Section 4 in order to introduce the key idea of our study, then we embed it in NSGA-II and MOEA/D in Section 5. Section 7 contains concrete examples of PC-MPL problems, used as benchmarks for a comparative quantitative evaluation of our approach and standard ones, adopting the metrics described in Section 6. These two sections show that our idea is not only a theoretical mathematical artifice, but also a practical and implementable tool. In Section 8 we discuss how the proposed approach differs from others existing in literature. Discussions and conclusions are provided in Sections 9 and 10, respectively.

2. Mixed Pareto Lexicographic Problems

Multi-objective and many-objective problems have been extensively studied in the past years, just as there is a vast literature about lexicographic optimization. Surprisingly, there does not seem to be any study concerning hybrid problems where only some of the objectives, or groups of them, can be lexicographically ranked, whereas the others are incommensurable and must be therefore treated in the Paretian way. A modest number of authors looked at specific instances in which priorities or precedences are somehow involved (see Section 8), but a general formulation is definitely lacking. This gap has gone unnoticed for years, however there are plenty of real world optimization problems that could benefit from research in this specific area. This is especially true for problems with a very large number of objectives: in most cases it is likely that not all of them hold the same exact importance, but one or more may have precedence over some others. Being able to exploit such information in the early stages of the optimization process is clearly better than postponing it to the final phase, that is decision making. Any approach of this kind, to be successful, should take advantage of the priority during the search itself, not only after. In other words, it would be ideal to use all the information available as soon (and as much) as possible in the process, and not merely at the end to filter the previously obtained solutions.
This specific paper aims to start filling this vacancy, describing the characteristics of those problems where some objectives do have priority over others, whereas some others do not; in addition, we suggest a strategy to address at least two subcategories of them, one in the present work and the other in a future work. First of all, since a general name for such problems does not exist in the literature yet, we choose to call them *Mixed Pareto-Lexicographic Multi-objective Optimization Problems (MPL-MOPs)*. This label refers to a broad class of problems, all sharing these features:

- At least two, but generally more objectives are involved (multi-objective)
- Some of the objectives have precedence over some others (lexicographic)
- Some of the objectives can not be compared to each other on the basis of importance (Pareto)

The word *mixed* reflects their non-homogeneous nature and suggests the use of a hybrid approach to deal with them. Furthermore, the relationship between Pareto optimality and priorities strongly depends on the structure of the specific problem: in particular, the manner in which the priorities are defined affects the mathematical description of the problem itself. In the next sections we identify and discuss two important subclasses of MPL-MOPs that appear to be significant for real applications. These two categories, respectively named *PC-MPL-MOPs* and *PL-MPL-MOPs*, share common characteristics, but are also very different in some respects.

2.1. MPL problems containing Priority Chains: PC-MPL

The first category is named *PC-MPL*, which is short for *Mixed Pareto-Lexicographic containing Priority Chains*. This class includes all the problems whose every low-priority objective is directly subordinated to exactly one other objective. The word *"direct"* is fundamental here: since some of the objectives can be arranged in a sort of "chain" built on the basis of precedence, *"direct"* means that, considering a pair of objectives, there is not another one between them, i.e. they are adjacent in that chain. Thinking of the objectives as nodes of a graph where the priorities are directed arcs, then each node has one or no direct predecessor, and one or no direct successor. Chains are not necessarily of the same length. Figure 1a can help to visualize the nature of these problems: it illustrates a possible precedence diagram of a PC-MPL-MOP. Formally, a problem of this kind is defined by these features:

- Any objective may have a direct precedence over another objective or none
- Any objective may be directly preceded by one other or none

The mathematical way to express their structure is the following:

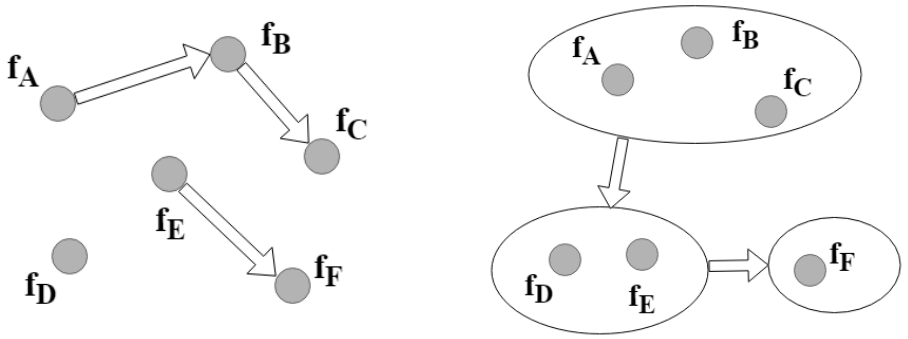
$$\min \begin{bmatrix} \text{lexmin} \left[f_1^{(1)}(x), f_1^{(2)}(x), \ldots, f_1^{(p_1)}(x) \right] \\ \text{lexmin} \left[f_2^{(1)}(x), f_2^{(2)}(x), \ldots, f_2^{(p_2)}(x) \right] \\ \vdots \\ \text{lexmin} \left[f_m^{(1)}(x), f_m^{(2)}(x), \ldots, f_m^{(p_m)}(x) \right] \end{bmatrix}.$$

Here the notation $\text{lexmin} \left[f_i^{(1)}, f_i^{(2)}, ..., f_i^{(pi)} \right]$ indicates the minimization of the objectives $\left\{ f_i^{(1)}, f_i^{(2)}, ..., f_i^{(pi)} \right\}$ according to the lexicographic method: $f_i^{(1)}$ is more important than $f_i^{(2)}$, which in turn has priority over $f_i^{(3)}$ and so on. Each lexmin block can be seen as a "container" of objectives ranked by importance: from now on we call this a *lexico-macro-objective* (or *macro-objective*, for brevity) and refer to it as $\underline{f_i}$. On the other hand, the outer min indicates the minimization (in the Pareto sense) of these macro-objectives. Hence we have:

$$\min \begin{bmatrix} \underline{f_1}(x) \\ \underline{f_2}(x) \\ \vdots \\ \underline{f_m}(x) \end{bmatrix}$$

where $\left\{ \underline{f_1}, \underline{f_2}, ..., \underline{f_m} \right\}$ are lexico-macro-objectives. Actually, from the optimization point of view, the transition from standard objectives to macro-objectives is not so disruptive. When comparing the macro-objective values of two different solutions, the solution with the lowest value of the primary objective (within that macro-objective) is preferred. But, if these two values happen to be equal, then they are assessed on the basis of the secondary objective, or the tertiary if they are identical again, and so forth. In other words, secondary objectives

are useful to compare solutions that have equal values of the more important objectives, within their relative macro-objective. It is important to stress that there is no mutual relationship between objectives belonging to different macro-objectives: they are incommensurable with each other, it is not possible to say which has priority over which. This property is non-trivial since it implies a certain degree of order and rigidity in the structure of PC-MPL-MOPs: for instance, two secondary objectives cannot be swapped in position in the model without also switching their primaries, otherwise it would become a completely different problem.



(a) PC-MPL-MOP (b) PL-MPL-MOP

Figure 1: Examples of two different classes of MPL-MOPs: a dot stands for an objective; an arrow indicates a precedence relationship; ovals represent groups of objectives.

Here is a minimal example of how comparisons between macro-objectives work. Consider the following three solutions of a minimization problem with two macro-objectives, each being a chain of two (shown horizontally):

$$A = \begin{bmatrix}[1 & 2]\\ [5 & 3]\end{bmatrix} \qquad B = \begin{bmatrix}[1 & 3]\\ [4 & 2]\end{bmatrix} \qquad C = \begin{bmatrix}[7 & 1]\\ [4 & 9]\end{bmatrix}$$

Solution A performs better than B in the first macro-objective (because $1 = 1$ and $2 < 3$), however B is better in the second macro-objective ($4 < 5$, the secondary objectives do not matter this time), therefore A and B are Pareto nondominated. A and C are nondominated too, since $1 < 7$ and $5 > 4$. On the other hand, B is better than C because it dominates on both the macro-objectives: in fact, $1 < 7$ (first macro-objective), $4 = 4$ and $2 < 9$ (second macro-objective).

The layout of PC-MPL-MOPs should be clear at this point. Our idea to solve them is described in detail from Section 5 onwards, but before that it is worth to outline another subclass of MPL-MOPs, which will be addressed in a future work.

2.2. MPL problems partitioned by Priority Levels: PL-MPL

The second category of problems we have identified is called *PL-MPL*, to indicate that the objectives have been partitioned according to a *priority level* assigned to each of them. Unlike PC-MPL problems, here the objectives are not arranged as chains, but instead they are classified into groups on the basis of their priority. Objectives belonging to the first priority level have precedence over all those in the second level, which in turn precede those in the third level, and so on. However, two objectives of the same level are considered equal in importance, so they must be optimized in the Pareto way. A secondary objective is not individually subordinated to a specific primary (in contrast to what happens in PC-MPL-MOPs), but is simply considered less important than all the primary objectives.
By definition, a PL-MPL problem has these characteristics:

- The objectives can be clustered by importance, creating a certain number of groups of objectives
- A group contains only objectives having the same priority
- Each objective belongs to exactly one priority level

Their mathematical description is:

$$\text{lexmin} \left[\min \begin{pmatrix} f_1^{(1)}(x) \\ f_2^{(1)}(x) \\ \vdots \\ f_{m1}^{(1)}(x) \end{pmatrix}, \min \begin{pmatrix} f_1^{(2)}(x) \\ f_2^{(2)}(x) \\ \vdots \\ f_{m2}^{(2)}(x) \end{pmatrix}, \ldots, \min \begin{pmatrix} f_1^{(p)}(x) \\ f_2^{(p)}(x) \\ \vdots \\ f_{mp}^{(p)}(x) \end{pmatrix} \right]$$

The best solutions for a PL-MPL-MOP are, first of all, Pareto optimal for the primary objectives. This Pareto optimal set becomes the domain of the search for solutions that also optimize (in the Pareto sense) the secondary objectives. The same is repeated for the third, fourth, ..., p-th level of priority. PL-MPL-MOPs are indeed very similar to pure lexicographic problems, with the only difference that they consist of a sequence of multi-objective problems instead of single-objective ones. An example of PL-MPL-MOP is depicted in Figure 1b. A deep analysis of PL-MPL-MOPs is left for future work.

3. Need of a new approach for PC-MPL-MOPs

In order to justify the necessity of a new algorithm, it is crucial to understand why the purely Paretian approach struggles with PC-MPL-MOPs. A standard optimizer like NSGA-II, or NSGA-III for many-objective problems, has fundamentally two options to face PC-MPL-MOPs: ignore priorities or ignore secondary objectives at all. Without any additional mechanism to handle priority, no other strategy is viable. As we are about to see, each of the above options has critical issues, which ultimately demands better ideas.

3.1. Ignoring secondary objectives

Ignoring all the objectives except the most important ones is certainly a way to simplify the problem, at the cost of sacrificing important information and, consequently, worsening the overall quality of the found solutions. As discussed previously, the purpose of secondary objectives is essentially to help discriminating between two or more solutions when the primaries are not enough to establish a clear winner. Without secondaries, the algorithm is bound to either preserve all the nondominated solutions, that is computationally expensive, or discard one or more of them, randomly or according to some policy, still with the risk of trashing potentially good ones thus slowing the optimization. When the number of nondominated solutions is sufficiently large, a common situation in many-objectives tasks, any of these strategy becomes inadequate and results in a significant performance degradation.

3.2. Ignoring the priorities, possibly filtering a posteriori

Another possibility is to treat all the objectives as if they had the same importance, no matter what their real priorities are, and run a conventional optimization algorithm on all these objectives. Then, *after* the end of the procedure, one may post-process the final population filtering out the solutions that would be dominated by other individuals in the same set if the priorities were eventually taken into account. Although slightly less naïve than the previous option, this one shows critical flaws too. First, a problem with m chains of p objectives each would become a purely Paretian problem with $m \times p$ objectives, likely a many-objective one, with all the negative consequences that this entails. Also, a secondary objective that is instead given the same importance of a primary one forces the algorithm to concurrently optimize both, or at least to try to; if those are somehow conflicting, spurious nondominated solutions or even clusters of them may emerge, in a way that is absolutely detrimental to the original optimization task. Moreover, the filtering procedure, despite being useful to pull out the very best solutions from the final set, may actually cause an overly severe skimming, sometimes preserving very few individuals out of many. Computationally speaking, ending up with two or three solutions after running the algorithm on a population of thousands is dramatically inefficient. The last two difficulties might also stack together (i.e., filtering an already mediocre set), exasperating the issue even further.

Now that it is clear why an innovative approach is needed, we are almost ready to present our strategy to address PC-MPL-MOPs. Before proceeding, a prerequisite to understand it is a little knowledge of the Grossone theory. Here follows a very short introduction to it, where we outline the key concepts. In-depth insights can be found in Sergeyev's papers, cited below.

4. The Grossone Methodology

A computational methodology able to deal with infinite, finite and infinitesimal numbers in the same single framework has been realized by Y. D. Sergeyev

[10, 13, 14, 15]. This has been possible thanks to the introduction of a new numeral system with infinite base. Such base is called *Grossone*, indicated by the numeral ① and defined as the number of elements in the set of the natural numbers $\mathbb{N}$. Historically, new numeral systems were often born as extensions of earlier and simpler ones, to overcome their limits and paradoxes (e.g. the invention of zero and negative numbers to switch from naturals to integers); Grossone pursues the same path, going beyond traditional finite-based numeral systems to get over the well-known difficulties in handling infinite and infinitesimal quantities. Three methodological postulates and The Infinite Unit Axiom have to be added to the axioms of real numbers to rigorously justify the advent of Grossone [10, 16]. In particular, this new axiom states that, for any given finite integer n, the infinite number $\frac{①}{n}$ is integer and larger than any finite number. The four basic operations, exponentiation and comparison operators are well-defined for ① and *Grossone*-based numerals; also, since the Infinite Unit Axiom is added to those of real numbers, all the standard properties (commutative, associative, existence of inverse, etc.) apply too. Instead of the usual symbol ∞, different infinite and/or infinitesimal numerals can be used thanks to ①. To set an example, the following relations hold for ①, $①^2$, $①^{-1}$, as for any other (finite, infinite, or infinitesimal) number expressible in the new numeral system:

$$0 \cdot ① = ① \cdot 0 = 0, \qquad ①^{-1} > ①^{-2} > 0$$
$$2① - ① = ①, \qquad ① \cdot ①^{-2} = ①^{-1}$$
$$\frac{-10.0①^3 + 16.0 + 42.0①^{-3}}{5.0①^3 + 7.0} = -2.0 + 6.0①^{-3}$$

The Grossone methodology has been already successfully applied to different areas of mathematics, such as optimization [17, 12, 18], ordinary differential equations [19, 20, 21] and game theory [22, 23, 24]. A complete list of papers using Grossone can be found at `http://www.theinfinitycomputer.com/~yaro/arithmetic/frv/`

A general way to express values formed by finite, infinite and infinitesimal quantities at the same time is provided in [10, 15, 16] by using a notation in the middle between the polynomial and the common positional numeral systems ones. A number $\underline{c}$ in this new numeral system ($\underline{c}$ is called *gross*-scalar) can be represented as:

$$\underline{c} = c_{p_m} ①^{p_m} + \ldots + c_{p_1} ①^{p_1} + c_{p_0} ①^{p_0} + c_{p_{-1}} ①^{p_{-1}} + \ldots + c_{p_{-k}} ①^{p_{-k}}$$

where $m, k \in \mathbb{N}$, exponents p_i are called *gross*-powers (they can be *gross*-scalars as well) with $p_0 = 0$. The digits $c_{p_i} \neq 0$, called *gross*-digits, are finite (positive or negative) numbers. In this system of numeration, finite numbers are represented by numerals with the highest *gross*-power equal to zero, e.g., $-6.2 = -6.2①^0$. Infinitesimals are represented by numerals having only negative (finite or infinite) *gross*-powers. The simplest infinitesimal is $①^{-1}$ for which $①^{-1} \cdot ① = 1$. Moreover it is worth to note that all infinitesimals are not

equal to zero, e.g. $①^{-1} > 0$. On the other hand, a number is infinite if it has at least one positive finite or infinite *gross*-power. For instance, the number $43.6①^{4.56} + 16.7①^{3.6} - 3.2①^{-2.1}$ is infinite, it consists of two infinite parts and one infinitesimal part.

4.1. Applications of Grossone for lexicographic problems

A very useful application of Grossone is a ploy to reformulate lexicographic problems in a way that allows performing actual numerical computations by means of lexicographic-macro-objectives, something that is not already possible in the previously described model. The idea is quite simple: given the lexicographic problem lexmin $\left[f^{(1)}(x), f^{(2)}(x), \ldots, f^{(p)}(x)\right]$, a *gross*-scalar is then built using each objective $f^{(j)}$ as a *gross*-digit relative to the *gross*-power $①^{1-j}$. Thus that problem can be rewritten as:

$$\min \left[f^{(1)}(x) + ①^{-1} f^{(2)}(x) + \ldots + ①^{1-p} f^{(p)}(x)\right]$$

or equivalently, as:

$$\min \left[\underline{f}(x)\right].$$

The higher the priority, the higher the *gross*-power: the most important objective, $f^{(1)}(x)$, is indeed associated the highest exponent (0), whereas $f^{(p)}(x)$ the lowest $(1-p)$. The advantage of this approach is clear: whereas before we had objectives represented by real numbers, now we have macro-objectives represented by *gross*-scalars. Since all the four basic operations are well-defined for *gross*-numbers, replacing objectives with macro-objectives in an algorithm is a completely transparent operation, which does not alter the inner logic of the code. Consequently, many existing non-lexicographic optimization algorithms can potentially be extended to solve certain types of lexicographic problems too. This same technique has been recently adopted to generalize the simplex algorithm [11, 12]. With regard to PC-MPL-MOPs, they can easily be formulated in terms of Grossone as:

$$\min \begin{bmatrix} f_1^{(1)}(x) + ①^{-1} f_1^{(2)}(x) + \ldots + ①^{1-p_1} f_1^{(p_1)}(x) \\ f_2^{(1)}(x) + ①^{-1} f_2^{(2)}(x) + \ldots + ①^{1-p_2} f_2^{(p_2)}(x) \\ \vdots \\ f_m^{(1)}(x) + ①^{-1} f_m^{(2)}(x) + \ldots + ①^{1-p_m} f_m^{(p_m)}(x) \end{bmatrix}$$

or, more concisely, as:

$$\min \begin{bmatrix} \underline{f_1}(x) \\ \underline{f_2}(x) \\ \vdots \\ \underline{f_m}(x) \end{bmatrix}.$$

5. PC-NSGA-II and PC-MOEA/D

Our proposal to solve PC-MPL problems consists of augmenting the original NSGA-II [1] and MOEA/D [3] algorithms, by extending them to handle *gross*-scalars quantities. We call them *PC-NSGA-II* and *PC-MOEA/D* respectively, to distinguish between these generalized versions (with *gross*-scalars) and the regular ones (with ordinary numbers). Of course these new versions must be equipped with the redefinition of the four elementary operations, in order to operate with *gross*-scalar quantities. We could build upon different algorithms as well, giving rise to PC-SPEA2 for instance, but we opted for the aforementioned two because they are very popular, simple and parameter-less, thus the ideal candidates to present our methodology. Clearly, they are considered parameter-less in the sense that they do not require other parameters in addition to those typically required by any population-based, single-objective genetic algorithm. Notice that such extended versions inherit the very same limits and benefit of the underlying algorithms when facing MOPs or MaOPs. As it will become clear in Section 7.3.1, inheriting limits and benefits means that if an algorithm, for instance, works particularly well with binary genotypes, then its PC version will manifest that property as well. On the contrary, if an algorithm becomes less effective as the number of objectives increases, its PC counterpart will suffer MaOPs with many chains. In order to further stress the enhancement in the supported data types, we also renamed the procedures to *PC_fast_nondominated_sort* and *PC_crowding_distance_assignment*. Again, these are aliases for the old names, as the logic behind the functions is still the same. The pseudocode is illustrated in Algorithm 1; at least three aspects deserve particular attention:

- *PC_fast_nondominated_sort*: the nondominance operator $\underline{\prec}$ (underlined) replaces the old $\prec$. This is very similar to the traditional definition of dominance, but the comparisons are now between macro-objectives (*gross*-scalars) instead of objectives (reals). The formal definition is:

  Definition 1. ***Pareto-Lexicographic dominance* (*PC_dominance*).** Given two solutions A and B, A dominates B (written $A \underline{\prec} B$) if the following relation holds:

  $$A \underline{\prec} B \iff \begin{cases} \underline{f_i}(x_A) \leq \underline{f_i}(x_B) & \forall i = 1 \dots m \\ \exists j : \underline{f_j}(x_A) < \underline{f_j}(x_B) \end{cases} \tag{1}$$

  The underlining notation has general validity: every time we use it, from now on, it indicates *gross*-scalar quantities or operations between them. This remains consistent with the use made in Section 2. Please notice that the lexicographic contribution is implicitly performed within the comparisons between two *gross*-scalars

- *PC_crowded_comparison_operator*: it works just like the old $\prec_n$ operator described in [1], but the last comparison is now performed between *gross*-

scalars because of the nature of PC_crowding_distance. We denote it by $\preceq_n$

- *PC_crowding_distance*: it is a *gross*-scalar, since it is computed by means of operations (subtractions, divisions, etc.) between *gross*-scalars. *gross*-operations, by design, preserve the relative importance between the objectives, and such property is reflected in the crowding distance computation too. The primary objectives, being the finite component of the macro-objectives values, play a major role in the density estimation, while the secondaries, mostly contributing to the infinitesimal part, are still very useful in those situations where it is hard to choose, e.g. when sorting two elements that have the same finite crowding distance but different infinitesimal values. The definition also makes sense from the perspective of the decision-maker, as it is reasonable to prefer having a larger variety of options in the domain of the most relevant aspects, rather than for less important objectives. Note that the crowding distance of the extreme points can be initialized to ①, similarly to NSGA-II where it is set to ∞. As long as we agree to represent the priorities with non-positive *gross*-powers only, which is a non-restrictive assumption, ① is always guaranteed to be greater than any other crowding distance value. An example of crowding distances computed among gross-scalars is provided in Table 2.

Algorithm 1 PC-NSGA-II algorithm

```
1:  procedure PC_NSGA_II
2:      R_t = P_t ∪ Q_t
3:      /* fast_nondominated_sort with gross-scalar objectives */
4:      F = PC_fast_nondominated_sort(R_t, 0)
5:      P_{t+1} = ∅
6:      i = 1
7:      while |P_{t+1}| + |F_i| ≤ N do
8:          /* Crowding distance with gross-scalars */
9:          PC_crowding_distance_assignment(F_i)
10:         P_{t+1} = P_{t+1} ∪ F_i
11:         i = i + 1
12:     /* Sort by PC_crowded_comparison_operator */
13:     sort(F_i, ≺_n)
14:     P_{t+1} = P_{t+1} ∪ F_i[1 : (N − [P_{t+1}])]
15:     Q_{t+1} = make_new_pop(P_{t+1})
16:     t = t + 1
```

Similarly, the MOEA/D algorithm can be extended by means of *gross*-scalars to obtain PC-MOEA/D (see Algorithm 2). For brevity reasons we do not go into the algorithm details, but please note that the PC extension of MOEA/D is even easier than the NSGA-II one. Indeed, the function to compute the dominance relation is the only one which needs to be extended to the *gross*-scalar case.

```
1: procedure PC_FAST_NONDOMINATED_SORT(P)
2:     for all p ∈ P do
3:         S_p = ∅
4:         n_p = 0
5:         for all q ∈ P do
6:             if p ⪯ q then                    ▷ PC_dominance being used here
7:                 S_p = S_p ∪ {q}
8:             else if q ⪯ p then               ▷ PC_dominance used also here
9:                 n_p = n_p + 1
10:        if n_p = 0 then
11:            p_rank = 1
12:            F_1 = F_1 ∪ {p}
13:    i = 1
14:    while F_i ≠ ∅ do
15:        Q = ∅
16:        for all p ∈ F_i do
17:            for all q ∈ S_p do
18:                n_q = n_q − 1
19:                if n_q = 0 then
20:                    q_rank = i + 1
21:                    Q = Q ∪ {q}
22:        i = i + 1
23:        F_i = Q
```

```
1: procedure PC_CROWDING_DISTANCE_ASSIGNMENT(F)
2:     n = |F|                          ▷ number of solution in the current front F
3:     for all i ∈ F do
4:         F[i]_dist = 0
5:     for j = 1 ... m do                ▷ m is the number of macro-objectives
6:         F = sort(F, f_j)              ▷ sort F according to j-th macro-objective
7:         F[1]_dist = F[n]_dist = ①
8:         for i = 2 ... (n − 1) do
9:             /* the PC_crowding_distance is a gross-scalar */
10:            F[i]_dist = F[i]_dist + (f_j(F[i+1]) − f_j(F[i−1])) / (f_j^max − f_j^min)
```

The new function works similarly to its old version except for the dominance operator, which becomes the $\underline{\prec}$ described in Equation 1 instead of the standard $\prec$.

Algorithm 2 PC-MOEA/D algorithm

procedure PC_MOEA/D
  $\underline{EP} = \emptyset$
  for $i = 1 \dots N$ **do**
    $\lambda^{i_1}, \dots, \lambda^{i_T} = find\_closest\_weights(\lambda_i, T)$
    $B_i = \{i_1, \dots, i_T\}$
  $x^1, \dots, x^N = initialize\_population(N)$
  for $i = 1 \dots N$ **do**
    $\underline{FV}^i = \underline{F}(x^i)$
  $\underline{z}_1, \dots, \underline{z}_m = initialize\_ref\_point(m)$
  while stop_criteria() = False **do**
    for $i = 1 \dots N$ **do**
      $k, l = rand(B_i, 2)$
      $y = make\_new\_sol(x^k, x^l)$
      $y' = mutate(y)$
      for $j = 1 \dots m$ **do**
        if $\underline{z}_j < \underline{f}_j(y')$ **then**
          $\underline{z}_j = \underline{f}_j(y')$
      for all $j \in B_i$ **do**
        if $\underline{g}^{te}(y'|\lambda^j, z) \leq \underline{g}^{te}(x^j|\lambda^j, z)$ **then**
          $x^j = y'$
          $\underline{FV}^j = \underline{F}(y')$
      $\underline{DD} = PC\_find\_dominated\_sol(\underline{EP}, \underline{F}(y'))$
      $\underline{DG} = PC\_find\_dominating\_sol(\underline{EP}, \underline{F}(y'))$
      $\underline{EP} = \underline{EP} \setminus \underline{DD}$
      if $\underline{DG} = \emptyset$ **then**
        $\underline{EP} = \underline{EP} \cup \underline{F}(y')$

6. Metrics

In the literature, several performance indicators have been proposed to evaluate the quality of a non-dominated solution set [25, 26, 27]. One of the most frequently used is the *hypervolume indicator* [28], mainly because it is the only Pareto compliant one. It has been shown that a Pareto non-compliant indicator may end up with misleading performance results [26, 27, 29]. However, the hypervolume indicator is often computationally intractable as soon as the objective space dimension grows up or more complex numerical structures are adopted, e.g. the gross-numbers, as in our work. Indeed, the less important objectives must be considered as if they were additional dimensions when assessing the computational effort needed for the hypervolume metrics. For example, the time required to compute the hypervolume in the benchmark MPL1 (see Section 7.2) is comparable to that of a problem with a six dimensional objective space ($\mathbb{R}^6$). Two other possible and well known metrics are the *generational distance* (GD) and the *inverted generational distance* (IGD), described in [30] and [31, 32] respectively. Both metrics evaluate the quality of a non-dominated objective vector set $A = \{a_1, ..., a_n\}$ with respect to a reference Pareto set $Z = \{z_1, ..., z_m\}$. The analytical definition of the former is:

$$GD(A) = \frac{1}{n}\left(\sum_{i=1}^{n} d_i^p\right)^{\frac{1}{p}}$$

where d_i is the Euclidean distance from a_i to its nearest reference point in Z, and $p \in \mathbb{N}$. The IGD metrics is an inverted form of the previous one, and is defined as:

$$IGD(A) = \frac{1}{m}\left(\sum_{i=1}^{m} d_i^p\right)^{\frac{1}{p}}$$

where d_i now represents the Euclidean distance from z_i to its nearest objective vector in A.

In this work we leveraged on the indicator $\Delta(A) = \max\{GD(A), IGD(A)\}$, firstly proposed in [29]. For our purposes, we set $p = 1$ throughout the whole paper. Such choice, in analogy with [33], is bivariate: i) it makes the meaning of GD and IGD clearly interpretable (the average of Euclidean distances); ii) it has often been used in the literature.

In this work we compared our approach against several well known algorithms found in the literature, as mentioned and described in Sections 7 and 8 respectively. In order to make the performances analysis quantifiable, we performed (benchmark-wise) a non-parametric statistical test. It consists of three steps: i) the Friedman test in order to compute a ranking among the algorithms performance distributions over the benchmark; ii) the Iman-Davenport test to evaluate whether there exists a statistical difference among the distributions; iii) if such statistical difference exists, i.e. if the Iman-Davenport p-value is lower than the significance level $\alpha = 0.05$, we can reject the null hypothesis and apply a post-hoc procedure, namely the Holm test. The latter checks the statistical

difference between the control approach (the one with the lowest Friedman rank) and the other ones. For more details on the tests see [34], while the description of the software tool (KEEL) used for the statistic can be found in [35].

7. Test cases for PC-MPL MOPs

There is now a substantial amount of scientific literature providing test problems to assess the performance of multi-objective optimization algorithms [36, 37, 38]. However, very few articles address problems where priorities exist among the objectives, and there does not seem to be any for the class of problems that we called *Mixed Pareto-Lexicographic*. Therefore, in order to verify our claims, we developed a couple of benchmark problems, specifically designed to have the following properties:

- Sufficiently complex to be significant, yet simple enough to be understandable
- Reasonably easy to explain
- Immediate to visualize

The purpose of the examples is to show that our approach to MLP-MOPs is not only viable but also more effective than using priorityless algorithms designed for standard MOPs and MaOPs. It is worth pointing out that our focus is on the proposed methodology rather than the algorithm itself. Comparisons are meaningful when done between our approach and its non-priority-based counterpart or a priority-based algorithm. Conversely, it is out of the scope of this article to analyze the raw performance of the PC algorithm, which ultimately depends on the chosen underlying MOP algorithm (NSGA-II or MOEA/D in this case), inheriting its pros and cons. We did our best to build examples that are both nontrivial and comprehensible, despite the lack of literature on this specific subject. A more detailed analysis concerning the identification and construction of meaningful or challenging benchmarks is left for future research.

In all our experiments, for real-encoded genotypes we adopt SBX operator and polynomial mutation. For binary-encoded ones we use two-point crossover and flip-bit mutation with probability $\frac{1}{l}$ (l is the length of the bitstring). For each benchmark we have repeated the experiments 50 times, computing 500 epochs per algorithm's run before stopping the experiment. The initial population is always generated randomly and includes 100 individuals.

In order to evaluate the quality of the Grossone-based approach, we compared its performances with six more algorithms: NSGA-II and MOEA/D ignoring the secondary objectives (Section 3.1), NSGA-II and MOEA/D ignoring the priorities and a posteriori filtering (Section 3.2), Tan et al. [39], Chang et al. [40] (Section 8).

7.1. Preliminary example (MPL0)

We chose to begin the experimental part of this article with knapsack problems (KP), because their popularity and plain structure make them ideal to be understood. But knapsack problems are as easy to understand as tough to solve: they are NP-hard, meaning that no algorithm exists which is able to identify the optimal solutions in polynomial time. There are countless problems in computer science, economics and math which involve resource allocation and can be modeled as KP or similar. The most common variant is probably the 0-1 knapsack problem, where each variable is binary and can only take the values 0 or 1. Another variation are multi-objective knapsack problems, where the objectives to maximize are two or more. In the scientific literature, the term "multi-objective" often hides the assumption that the objectives are handled in the Pareto sense. However, this is not always the case, because sometimes they should be treated lexicographically, or even *mixed Pareto-lexicographically*, like in this context. Here we focus on an instance of a PC-MPL 0-1 knapsack problem (abbreviated as PC-MPL-01-KP). As the name suggests, a PC-MPL-01-KP is nothing but a constrained PC-MPL problem with knapsack-like objective functions. A simple minimal example can explain the nature of such problems better than any words. Thus, after gaining confidence with a dummy problem, we move to another one which is similar in structure, but closer to real problems in size and complexity. The latter serves as the first benchmark for our study, and two more examples follow then. The preliminary problem, which we refer to as MPL0, is defined as:

$$
\begin{aligned}
&\max \begin{bmatrix} \underline{f}_1(x) \\ \underline{f}_2(x) \end{bmatrix} \\
&\text{s.t.} \quad Wx \leq C \\
&\underline{f}_1(x) = \underline{V}_1 x \\
&\underline{f}_2(x) = \underline{V}_2 x \\
&\underline{V}_1 = V_1^{(1)} + ①^{-1} V_1^{(2)} \\
&\underline{V}_2 = V_2^{(1)} + ①^{-1} V_2^{(2)} \\
&V_1^{(1)} = [4,5,6,7,11]^T \\
&V_1^{(2)} = [1,2,3,4,6]^T \\
&V_2^{(1)} = [7,8,2,1,3]^T \\
&V_2^{(2)} = [4,3,2,1,5]^T \\
&W = [1,2,3,4,5]^T \\
&C = 10 \\
&x_i \in \{0,1\} \qquad\qquad i = 1,\ldots,5
\end{aligned}
\tag{2}
$$

It features 5 binary variables, 2 macro-objectives where each is a chain of length 2 (4 vectors of item values in total), and a weight constraint. Since the number of variables is relatively low and the weights follow an intelligible pattern, it is straightforward to visualize which solutions fit in the knapsack. The item

values were empirically determined in order to make the example as descriptive as possible; of course, other combinations may work as well. Out of the $2^5 = 32$ existing solutions, we identified four (listed in Table 1) that are worth of special consideration:

	x	$\underline{f}_1(x)$	$\underline{f}_2(x)$	Pareto optimal
a	10011	$22 + 11①^{-1}$	$11 + 10①^{-1}$	no
b	01101	$22 + 11①^{-1}$	$13 + 10①^{-1}$	yes
c	11110	$22 + 10①^{-1}$	$18 + 10①^{-1}$	yes
d	11001	$20 + 9①^{-1}$	$18 + 12①^{-1}$	yes

Table 1: Four interesting solutions (not all optimal) of MPL0. Variables are expressed as bit-strings, along with their objective values.

- Solutions (a) and (b) are equivalent for the first macro-objective $\underline{f}_1$, but (b) is clearly more valuable for $\underline{f}_2$, therefore (b) dominates (a)
- (b) is slightly better than (c) for $\underline{f}_1$ because its secondary objective is higher (the secondary is taken into account since they are equivalent for the primary). On the other hand, (c) is better than (b) on the second macro-objective. Thus (b) and (c) do not dominate each other
- Similarly, (c) and (d) happen to be nondominated. This time, the second secondary plays an active role
- (b) and (d) are not PC_dominated too, and secondaries do not even have to be considered (because primaries differ)

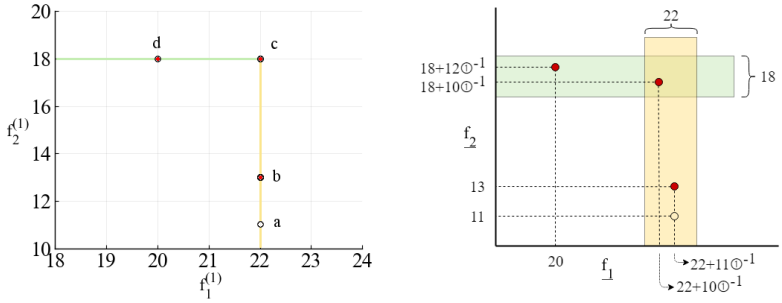
Except for (a), which is dominated, the other three solutions are nondominated. It can be verified that no other solution exists being better than these in any of the macro-objectives. Figure 2a shows the primary objectives of the four solutions. Note how (c) and (d) seem to be arranged horizontally, also (b) and (c) vertically; such a pattern would be impossible in a traditional maximization problem, since the "rightmost" solution would dominate the others on the same horizontal line, and analogously the "uppermost" would dominate those beneath in its same vertical line, since here we are considering a maximization problem. In a PC-MPL, on the other hand, this configuration is totally legitimate and common. Indeed, asserting that those solutions are aligned horizontally/vertically would be an inaccurate sentence: although they are in their finite components, this is no longer true if the infinitesimal parts are taken into account as well. Even though it is impossible to represent both finite and infinitesimal numbers on the same scale, one can imagine to "infinitely zoom" into one (finite) coordinate and visualize its infinitesimal neighborhood. Figure 2b does exactly this: it takes the previous figure and zooms into $f_1^{(1)} = 22$ and $f_2^{(1)} = 18$, revealing the infinitesimals corresponding to secondary objectives, thus disproving the apparent flaw in the model.

As an example, Table 2 contains the crowding distance computed for the four points; if an algorithm, for instance, had to select two out of the three

non-dominated solutions, this distance would be the determinant factor.

	x	$\underline{f}_1$	$\underline{f}_2$	PC_crowding_distance
a	10011	$22 + 11①^{-1}$	$11 + 10①^{-1}$	①
b	01101	$22 + 11①^{-1}$	$13 + 10①^{-1}$	$1 - 0.286①^{-1}$
c	11110	$22 + 10①^{-1}$	$18 + 10①^{-1}$	$1.714 + 0.082①^{-1}$
d	11001	$20 + 9①^{-1}$	$18 + 12①^{-1}$	①

Table 2: Crowding distance computed for the four solutions previously listed in Table 1.



(a) (b)

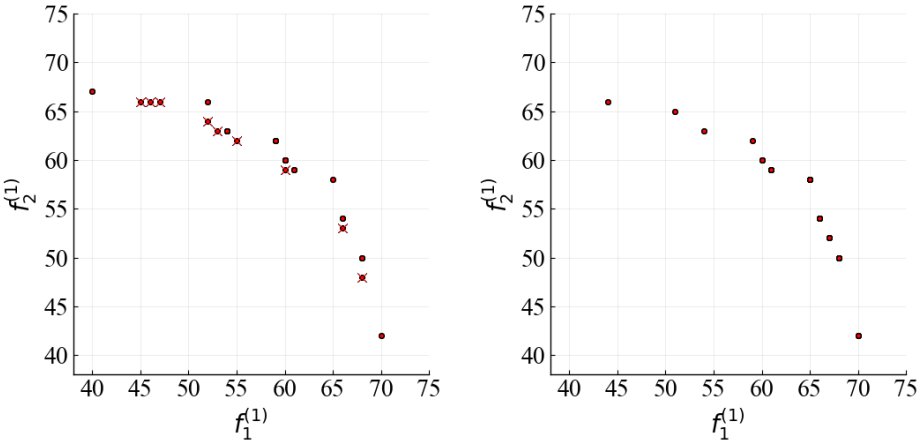
Figure 2: The four interesting solutions of MPL0. Pareto optimal ones are highlighted in red. Fig. 2a shows the primary objectives only; Fig. 2b, instead, plots both primaries and secondaries in a fictitious way: the infinitesimal components are "expanded" in $(22, 18)$, to better show that some solutions (but not all) are only apparently aligned horizontally/vertically, i.e. secondary objectives are to be considered in this case. All the points inside the yellow stripe share $f_1^{(1)} = 22$, similarly green ones have $f_2^{(1)} = 18$. Note that the solution *'a'*, according to Table 1, has the very same $\underline{f}_1$ value of *'b'*, hence these two are truly aligned; *'a'* is nevertheless dominated by *'b'*, since it does better on $\underline{f}_2$.

7.2. Test Problem 1: MPL1

The second example of a PC-MPL-01-KP, named MPL1, is larger and less obvious than the previous, but the structure remains fundamentally the same. MPL1 features 6 objectives, aggregated in 2 chains of 3 objectives each:

$$
\begin{aligned}
&\max \begin{bmatrix} \underline{f}_1(x) \\ \underline{f}_2(x) \end{bmatrix} \\
&\text{s.t.} \quad Wx \leq C \\
&\underline{f}_1(x) = \underline{V}_1 x \\
&\underline{f}_2(x) = \underline{V}_2 x \\
&\underline{V}_1 \;\; = V_1^{(1)} + ①^{-1} V_1^{(2)} + ①^{-2} V_1^{(3)} \\
&\underline{V}_2 \;\; = V_2^{(1)} + ①^{-1} V_2^{(2)} + ①^{-2} V_2^{(3)} \\
&V_1^{(1)} = [7,1,5,2,9,4,4,2,8,2,1,6,9,5,4,4,9,5,3,3]^T \\
&V_1^{(2)} = [3,1,6,6,7,2,5,2,1,9,1,8,9,6,7,4,4,5,9,4]^T \\
&V_1^{(3)} = [2,1,3,5,8,9,5,7,1,6,4,7,9,5,1,5,5,4,4,2]^T \\
&V_2^{(1)} = [1,7,3,6,2,3,7,7,9,9,5,3,5,3,1,8,6,1,9,3]^T \\
&V_2^{(2)} = [2,6,4,7,7,4,2,4,9,4,3,3,6,8,2,8,1,6,8,8]^T \\
&V_2^{(3)} = [8,9,5,9,8,9,4,9,7,5,4,5,3,4,4,7,3,8,2,4]^T \\
&W \;\; = [4,8,6,5,5,5,7,4,8,4,4,1,8,7,4,3,5,1,9,5]^T \\
&C \;\; = 50 \\
&x_i \in \{0,1\} \qquad\qquad i = 1,\ldots,20
\end{aligned}
\tag{3}
$$

Let us make a few qualitative considerations on the results achieved before moving to the statistical analysis of the outcomes. For the sake of simplicity, the focus will be on PC-NSGA-II and its standard counterpart. Running PC-NSGA-II, the experiment produces the result shown in Fig. 3a, which highlights the 19 distinct solutions found. We compared this result with the true Pareto front, computed by means of brute-force complete enumeration, which is made of the 22 distinct solutions displayed in Fig. 4. The listings show that, with respect to the 19 found solutions, 16 are true optima, 2 differ from their closest true optimum for a secondary, 1 for a primary. Regarding the 6 out of 22 truly optimal solutions that PC-NSGA-II did not manage to identify precisely, for 2 of them there exists at least one found solution with the same primary values; for the other 4, the error on primary objectives is never greater than 3. To give an idea of how important the secondaries are in the optimization process, consider also the standard NSGA-II working only on the primary objectives. The secondary objectives are computed only at the end of the optimization process, and do not affect the evolution at all. Not only the solutions happen to be fewer, but also of lower quality compared to PC-NSGA-II ones. Out of 11, 8 are real optima, 2 are sub-optimal with respect to primaries, 1 to secondaries. This shortage of solutions from NSGA-II is also the consequence of its inability to preserve those solutions that we previously denoted as "apparently horizontal/vertical", due to dominance reasons, as explained before. Such solutions, marked with a cross in Fig. 3a, are indeed absent in Fig. 3b.



(a) PC-NSGA-II, considering all the objectives during the optimization.

(b) NSGA-II, considering only the two primary objectives.

Figure 3: Primary objectives of MPL1 Pareto front, after 300 generations with 400 individuals. The solutions found by PC-NSGA-II that would be instead discarded by NSGA-II (i.e. missing on the right plot) are also marked with a cross.

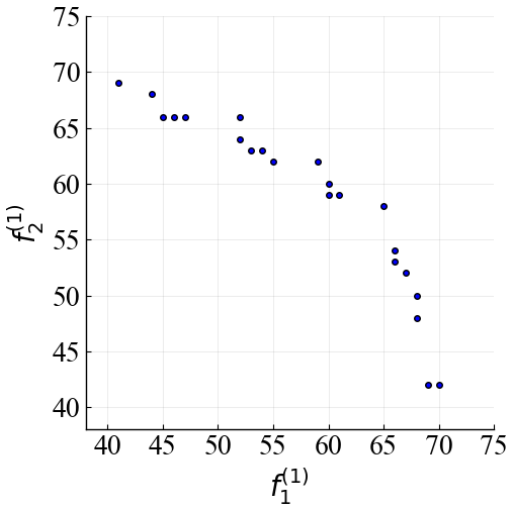


Figure 4: MPL1 true Pareto front, by complete enumeration.

7.3. Performance evaluation on MPL1

In this section we evaluate and compare the results of the algorithms on the MPL1 benchmark. The analysis follows three steps. Firstly, the algorithms performances have been measured by means of the metric $\Delta(\cdot)$ introduced in Section 6. In Table 3 the performances are reported as mean value and standard deviation of the measures collected over 50 runs. Then, the Friedman ranking

and the Iman-Davenport test have been applied to investigate the presence of a statistical separation. Finally, the Holm procedure between the best performing approach and all the others is computed, reporting the p-values in Table 5.

7.3.1. Mean and Standard deviation

Table 3 shows the mean and the standard deviation of the algorithms performances over the MPL1 benchmark. From that Table, we evince that the PC algorithms perform better than their standard counterparts and the other priority-based approaches. Moreover, PC-MOEA/D turns out to be the best performing algorithm, immediately followed by MOEA/D w/o the secondaries. This fact confirms what we asserted in Section 5, that is the PC algorithms get the same benefits and suffer the same limits of the underlying optimizer. Let us spend a few more words to clarify this concept. Given the results we have collected, it is reasonable to say that MOEA/D works consistently better than NSGA-II on this specific benchmark (MPL1). Indeed, not only PC-MOEA/D gets a lower score (i.e., mean) than PC-NSGA-II, but also MOEA/D w/o secondaries beats NSGA-II w/o secondaries, and the same holds even for the third version of MOEA/D versus NSGA-II.
Of course, the property of MOEA/D being better than NSGA-II is not a general one: as we will see, there exist other problems where the opposite is true, that is NSGA-II beats MOEA/D. Moreover, the algorithms w/o secondaries usually perform more or less like their PC counterpart, at least on the primary objectives, since both try to optimize them in a Pareto fashion: the first one giving them the highest priority and trying to optimize them before the others, the second working only on them. All in all, these considerations justify why PC-MOEA/D is at the top of the ranking (MOEA/D beats the others on MPL1, and the PC versions by design are better suited for this class of problems), and why MOEA/D w/o secondaries is second (the version w/o secondaries performs closely to its PC extension on the primaries). The previous observations will be valid also for the next two benchmarks, so we will avoid repeating them for brevity.

Algorithm	Mean	Std
PC-MOEA/D	$\mathbf{0.36 + 0.25①^{-1} + 0.99①^{-2}}$	$\mathbf{0.31 - 0.07①^{-1} + 1.11①^{-2}}$
MOEA/D pre	$0.54 + 0.43①^{-1} + 3.22①^{-2}$	$0.19 - 0.22①^{-1} + 1.37①^{-2}$
MOEA/D post	$1.44 - 0.22①^{-1} + 0.88①^{-2}$	$0.54 + 0.15①^{-1} + 0.59①^{-2}$
PC-NSGA-II	$\mathbf{1.6 + 0.17①^{-1} + 1.2①^{-2}}$	$\mathbf{0.74 + 0.19①^{-1} + 0.44①^{-2}}$
NSGA-II pre	$1.95 + 0.57①^{-1} + 2.37①^{-2}$	$0.91 + 0.21①^{-1} + 0.53①^{-2}$
NSGA-II post	$1.79 + 0.08①^{-1} - 0.77①^{-2}$	$0.33 - 0.02①^{-1} + 0.88①^{-2}$
Tan et al.	$12.83 + 2.71①^{-1} + 4.03①^{-2}$	$0.73 + 0.04①^{-1} + 1.2①^{-2}$
Chang et al.	$18.01 + 4.45①^{-1} - 4.42①^{-2}$	$2.53 + 2.31①^{-1} + 3.48①^{-2}$

Table 3: Mean and standard deviation of metric $\Delta(\cdot)$ on MPL1 after 50 repetitions.

7.3.2. Friedman and Iman-Davenport tests

In Table 4 the algorithms are ranked by means of the Friedman test, which also output a statistic (distributed according to a chi-square with 7 degrees of

freedom) of 300.761667 and a P-value of 0. This implies the presence of a statistical separation among some algorithms in the pool. In order to confirm such result, the Iman-Davenport test has been computed. Its statistic (distributed according to a F-distribution with 7 and 343 degrees of freedom) is 299.305859, while the P-value is again 0. Thus, the rejection of the null hypothesis is confirmed so we have to check how many algorithms are statistically separated from the best one.

Algorithm	Ranking
PCMOEA/D	1.28
MOEA/D pre	1.91
MOEA/D post	4.04
PC-NSGA-II	4.14
NSGA-II pre	4.73
NSGA-II post	4.9
Tan et al.	7.02
Chang et al.	7.98

Table 4: Algorithms ranked by the Friedman test (MPL1)

7.3.3. Post hoc comparison

By means of the Holm procedure we have investigated which performances are statistically separated from that of best algorithm, i.e., PC-MOEA/D. The test's output is reported in Table 5, where i indicates the Friedman rank, z is the normal distribution statistic for the p-value computation, p is precisely the p-value. In the Holm's procedure, the hypotheses with a p-value lesser or equal to $\alpha = 0.05$ are rejected, i.e., the corresponding algorithm can be considered statistically different when compared against PC-MOEA/D. As expected, the only non-separable algorithm is MOEA/D without secondaries. The reason is that, at the moment, a statistical test for comparing infinitesimal separations does not exist. Thus, since MOEA/D without secondaries has performance close to PC-MOEA/D when we look only at the primaries (i.e., finite values), and because the statistical test can work only on the finite components of the metrics, then it turns out that the two approaches cannot be statistically separated. However, another ingredient playing a role in the non-separability between PC-MOEA/D and MOEA/D without secondaries is the problem simplicity. Indeed, in MPL2 and MPL3, which are more complex, this fact does not happen.

7.4. Test Problem 2: MPL2

The previous examples shed some light on the properties of PC-MPL problems, showing how PC-NSGA-II and PC-MOEA/D are capable of exploiting the priority information better than both their standard counterparts and other priority-handling algorithms available in the literature. The following benchmarks are meant to be more general than the previous two in terms of shape of the objective functions.

i	Algorithm	$z = (R_0 - R_i)/SE$	p
7	Chang et al.	13.676318	0
6	Tan et al.	11.716726	0
5	NSGA-II post	7.389294	0
4	NSGA-II pre	7.042283	0
3	PC-NSGA-II	5.837951	0
2	MOEA/D post	5.633826	0
1	MOEA/D pre	1.285982	0.198449

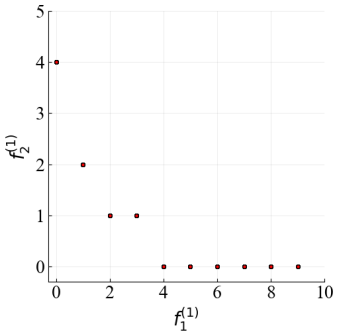
Table 5: Post hoc comparison with $\alpha = 0.05$ and $SE = 0.489$ (MPL1)

The next problem, MPL2, has two primary objectives and two secondaries:

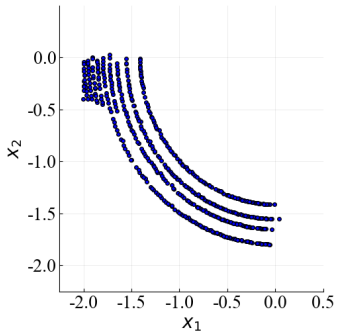
$$\begin{aligned}
&\min \begin{bmatrix} f_1^{(1)}(x) + ①^{-1} f_1^{(2)}(x) \\ f_2^{(1)}(x) + ①^{-1} f_2^{(2)}(x) \end{bmatrix} \\
&f_1^{(1)}(x) = \lfloor (x_1^2 + x_2^2 - 1)^2 \rfloor \\
&f_2^{(1)}(x) = \lfloor (x_1^2 + x_2^2 - 4)^2 \rfloor \\
&f_1^{(2)}(x) = (x_2 + 2)^2 \\
&f_2^{(2)}(x) = (x_1 + 2)^2 \\
&x_1, x_2 \in [-3, 3]
\end{aligned} \tag{4}$$

The first two objectives are kind of discrete distances between two circumferences, with radius 1 and 2 respectively. Without considering the *floor* function and also ignoring the secondary objectives, the Pareto optimal solutions would be formed by the annulus of all the points enclosed by these two circles. Adding the floor function makes the problem discrete, with multiple neighboring solutions "collapsing" into the same objective value: the result is a limited number of Pareto optimal solutions in the objective space, each of which can be generated by any point within a specific annulus in the decision space. Now examine the two secondary objectives: they are the euclidean distances from the lines $x_2 = -2$ and $x_1 = -2$ respectively; a greater distance from one of these lines results in a greater (i.e. worse) value of the corresponding objective for the given solution. To understand their effect within the overall problem, consider $f_1^{(2)}(x)$, which is the secondary of the first macro-objective. It plays a role only when comparing two solutions that share the same value of $f_1^{(1)}(x)$: between the two, the one with the lower value of $f_1^{(2)}(x)$ is then preferred. Practically, it means that if two solutions are equidistant from the inner circumference, then what matters to minimize the first macro-objective is the x_2 coordinate (closer to -2 is better). An analogous situation holds for the second macro-objective: first the value of $f_2^{(1)}(x)$ is taken into consideration in the comparison of two solutions and, if such values turn out to be equal (i.e. they are equidistant from the outer circle), then $f_2^{(2)}(x)$ is used to discriminate between the two.

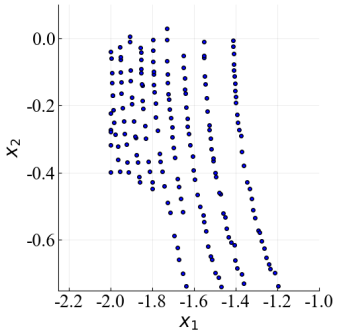
As before, let us give qualitative comments before moving to statistical analyses. For the sake of continuity, the focus is again on PC-NSGA-II and its standard counterpart NSGA-II. The results we obtained for the former are de-



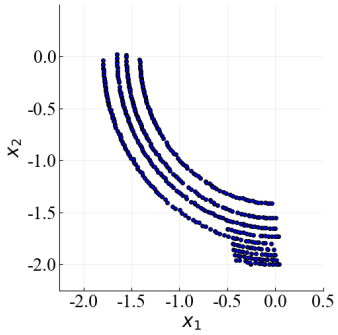
(a) Pareto front, primaries.



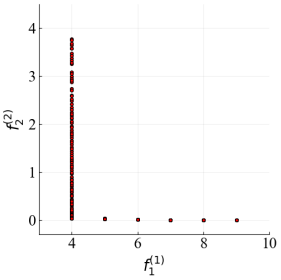
(b) Pareto set.



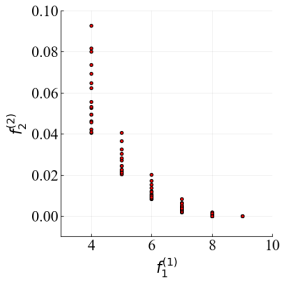
(c) Zoom of the "top" area of Figure 5b



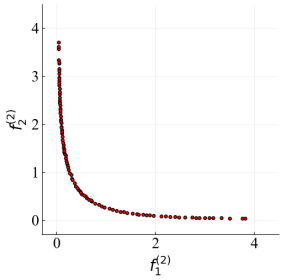
(d) Pareto set after swapping the position of secondaries.



(e) Subset of the Pareto front made by solutions with $f_2^{(1)} = 0$. $f_1^{(1)}$ $f_2^{(2)}$ are plotted here.



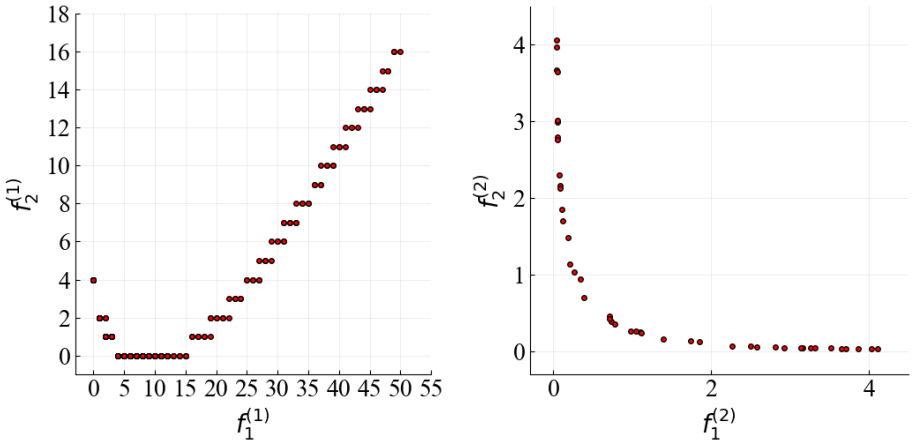
(f) Zooming the "bottom" area of Fig. 5e that one appearing as a series of horizontal points.



(g) Secondaries of the solutions having $f_1^{(1)} = 4$ and $f_2^{(1)} = 0$.

Figure 5: Obtained solutions by PC-NSGA-II for MPL2.

picted in Figure 5, with Fig. 5a showing explicitly the values of the primary objectives. Here, the Pareto front is made up of discrete points rather than



(a) Pareto front, first two objectives.

(b) Secondary objectives of the points with $f_1^{(1)} = 4$, $f_2^{(1)} = 0$.

Figure 6: Obtained solutions by NSGA-II for MPL2. Please notice that in Fig. 6a some solutions appear to be dominated, but they are not when considering all the four objective functions (only two objectives are shown here)

being a continuous curve. Actually, if we could zoom-in on any of these point, we would see not a single solution, but a cluster of infinitesimally close ones, whose distances are proportional to the values of the secondary objectives. Observe how a bunch of the solutions seem to be arranged horizontally, especially those having $f_1^{(1)} = 0$: it is the very same effect we have seen in the previous example, and can be examined in detail in Fig. 5e, that is an infinitesimal zoom on those solutions having $f_2^{(1)} = 0$, i.e. the "horizontal" ones: the x-axis represents $f_1^{(1)}$ as before, the y-axis instead shows the infinitesimal part of the second macro-objective, namely the second secondary objective $f_2^{(2)}$. Although some points still seem to be horizontal, now it is just a matter of scale and small numerical values; re-scaling the figure vertically is enough to prove that they are not (Figure 5f). Now that the horizontal "collapse" has been finally unraveled, we can observe it reappearing vertically; repeating the same procedure on the vertical axis, that is zooming the second macro-objective to expose its infinitesimal part, ends up in Figure 5g, which is indeed a zoom on those solutions with $f_1^{(1)} = 4$ and $f_2^{(1)} = 0$. Note the peculiarity of this: what at finite scale appears merely as a single solution, actually is an entire Pareto *subfront* at infinitesimal scale; in other words, we are observing a front inside a solution of a "parent" front. If the previous benchmark MPL1 proved that secondary objectives cannot be ignored without repercussions on the quality of solutions, MPL2 aims to show that bad consequences occur as well when the priorities are not taken into account. Here follows a concise analysis of the results achieved by NSGA-II running on the equivalent four-objective priority-less problem. The experiment result is shown in Fig. 6a, which displays the first two objectives

(primaries of the original problem, although priority is ignored here). One may argue that it does not look like a Pareto front, being a non-monotonic pseudo-curve: the reason is that only 2 out of the 4 objectives are plotted there; the Pareto jfront is actually in a 4-dimensional space, visualizing it is rather hard and not even relevant for the following considerations. What is evident, on the other hand, is that a large number of solutions are spurious and sub-optimal for the actual MPL2 problem (the one with priority, to be clear): the cause is the interference of low-priority obectives, which behave here as if they had the same importance of primaries; this effect was already mentioned in section 3. Furthermore, NSGA-II struggles not only with regard to quality, but also quantity of solutions: this can be observed comparing Figures 5g and 6b, both representing the third and fourth objectives for a certain subset of primary-optimal solutions; the latter clearly shows a minor density of points. An important thing to remark is that any traditional preemptive lexicographic approach would never preserve those solutions (potentially Pareto-optimal for PC-MPL-MOPs), that we grossly called "apparently horizontal/vertical", as they would be marked as dominated and discarded, fundamentally due to the inability of the algorithm to examine secondary functions. Finally, MPL2 makes it possible to verify an additional interesting property of PC-MPL-MOPs, that is the non-permutability of the secondary objectives within the problem formulation. In simple terms, exchanging the positions of any two objectives in the mathematical model may result in a radical alteration of the problem and its optima. With reference to MPL2, swapping the secondaries so that $f_1^{(2)}$ becomes the secondary of $f_2^{(1)}$ and $f_2^{(2)}$ the secondary of $f_1^{(1)} = 0$, causes the Pareto front to remain unchanged (coincidentally), however the Pareto set turns into Fig. 5d.

It should be noted that, in this example and the others, increasing the population size or the number of iterations would not result in a significant mitigation of the discussed problems, which is reasonable considering the intrinsic nature of such issues. We had confirmation of this by running the experiments several times with different parameters, which we do not report here merely due to space reasons.

7.5. Performance calculation on MPL2

In the following three subsections the results evaluation and comparison are illustrated. Maintaining the same order as before, firstly the means and the standard deviations of the metric $\Delta(\cdot)$ are discussed, then the investigation of the statistical separation is presented, finally the Holm test output is shown.

7.5.1. Mean and Standard deviation

In the next two tables, the mean and the standard deviation of the algorithms over the MPL2 benchmark are reported. From Table 6 we see that the PC algorithms perform better than their standard counterparts and than the priority-based approaches. Even if PC-NSGA-II turns out to be the best performing algorithm, PC-MOEA/D is the second one. This fact suggests that

NSGA-II and MOEA/D works more or less equivalently on the MPL2 benchmark. Such speculation is corroborated by the values of means of the standard versions of MOEA/D and NSGA-II, which are very close from each other (see Table 6).

Algorithm	Mean	Std
PC-NSGA-II	$\mathbf{0.07①^{-1}}$	**0.01**
PC-MOEA/D	$\mathbf{0.12 + 0.74①^{-1}}$	$\mathbf{0.06 - 0.06①^{-1}}$
NSGA-II post	$0.26 + 0.85①^{-1}$	$0.05 + 0.02①^{-1}$
MOEA/D post	$0.26 + 0.92①^{-1}$	$0.05 + 0.08①^{-1}$
NSGA-II pre	$0.26 + 5.69①^{-1}$	$0.05 - 0.03①^{-1}$
MOEA/D pre	$0.26 + 6.35①^{-1}$	$0.05 - 0.41①^{-1}$
Tan et al.	$0.29 + 0.69①^{-1}$	$0.24 - 0.34①^{-1}$
Chang et al.	$2.14 - 0.12①^{-1}$	$1.18 + 0.68①^{-1}$

Table 6: Mean and standard deviation of metric $\Delta(\cdot)$ on MPL2 after 50 repetitions.

7.5.2. Friedman and Iman-Davenport tests

The Friedman ranking of the algorithms is shown in Table 7. The test also outputs a statistic (distributed according to a chi-square with 7 degrees of freedom) of 253.073333 and a p-value of 0. This implies the presence of a statistical separation among some algorithms in the pool. In order to confirm such result, the Iman-Davenport test has been computed. Its statistic (distributed according to a F-distribution with 7 and 343 degrees of freedom) is 127.937891 and the p-value again 0. The rejection of the null hypothesis is confirmed, so we have to check how many algorithms are statistically separated from the best one.

Algorithm	Ranking
PC-NSGA-II	1
PC-MOEA/D	2.42
NSGA-II post	5.1
MOEA/D post	5.1
NSGA-II pre	5.1
MOEA/D pre	5.1
Tan et al.	5.18
Chang et al.	8

Table 7: Algorithms ranked by the Friedman test (MPL2)

7.5.3. Post hoc comparison

Table 8 reports the p-values obtained by applying the post hoc method, namely the Holm test. Analyzing them, we deduce that PC-NSGA-II is not only the best algorithm, but also statistically separable from all the other approaches. Indeed, all the p-values are smaller than the critical threshold $\alpha = 0.05$.

i	algorithm	$z = (R_0 - R_i)/SE$	p
7	Chang et al.	14.28869	0
6	Tan et al.	8.491148	0
5	MOEA/D post	8.36909	0
4	MOEA/D pre	8.36909	0
3	NSGA-II post	8.36909	0
2	NSGA-II pre	8.36909	0
1	PC-MOEA/D	2.898563	0.003749

Table 8: Post hoc comparison with $\alpha = 0.05$ and $SE = 0.489$ (MPL2)

7.6. Test Problem 3: MPL3

The test case we are about to describe is a modified version of POL, a well-known benchmark based on the Poloni's study [41] which is also included in Van Veldhuizen's suite [36]. The problem, labeled MPL3, is defined as:

$$
\begin{aligned}
&\min \begin{bmatrix} f_1^{(1)}(x) & + ①^{-1} f_1^{(2)}(x) \\ f_2^{(1)}(x) & \end{bmatrix} \\
&f_1^{(1)}(x) = \lfloor \alpha \left(1 + (A_1 - B_1)^2 + (A_2 - B_2)^2\right) \rfloor / \alpha \\
&f_2^{(1)}(x) = \lfloor \alpha((x_1 + 3)^2 + (x_2 + 1)^2) \rfloor / \alpha \\
&f_1^{(2)}(x) = x_1^2 \\
&A_1 = 0.5\sin(1) - 2\cos(1) + \sin(2) - 1.5\cos(2) \\
&A_2 = 1.5\sin(1) - \cos(1) + 2\sin(2) - 0.5\cos(2) \\
&B_1 = 0.5\sin(x_1) - 2\cos(x_1) + \sin(x_2) - 1.5\cos(x_2) \\
&B_2 = 1.5\sin(x_1) - \cos(x_1) + 2\sin(x_2) - 0.5\cos(x_2) \\
&x_1, x_2 \in [-\pi, \pi] \qquad \alpha = 3
\end{aligned}
\tag{5}
$$

The original structure of POL remains unchanged, except for the insertion of the $floor$ function (to discretize the problem) and the addition of a third objective, $f_1^{(2)}$, which is to be considered less important than $f_1^{(1)}$.

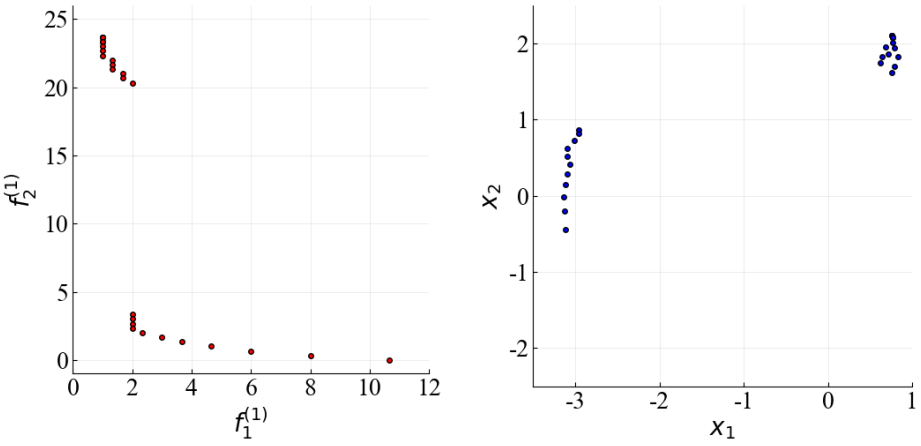
The parameter α can be tuned to adjust the granularity of the discretization. In the nondominance ranking procedure, the secondary objective $f_1^{(2)}$ intervenes only when two solutions have the same value of $f_1^{(1)}$, favoring the one whose coordinate x_1 is closer to 0. Therefore, it is reasonable to expect the new optimal solutions to be very similar to those of original POL, at most a little shifted towards the line $x_1 = 0$; of course, the magnitude of this effect depends on α. Once again, it may happen to find, in the Pareto frontier of primary objectives, multiple solutions with equal values of $f_1^{(1)}$ but different values of $f_2^{(1)}$, which graphically seem vertically-aligned points: this is no different from what we observed in MPL1 and MPL2.

Figure 7 shows the results of PC-NSGA-II for MPL3. As expected, the shape of the Pareto front looks very similar to that of POL: the biggest difference is that it appears as a set of disconnected points, due to the discretization (floor function). A certain number of solutions seem "vertical": although they share

the same value of $f_1^{(1)}$, they stand nondominated when compared using $f_1^{(2)}$ and $f_2^{(1)}$. As in MPL2, such points are vertical in their finite part, but not in their infinitesimal one.

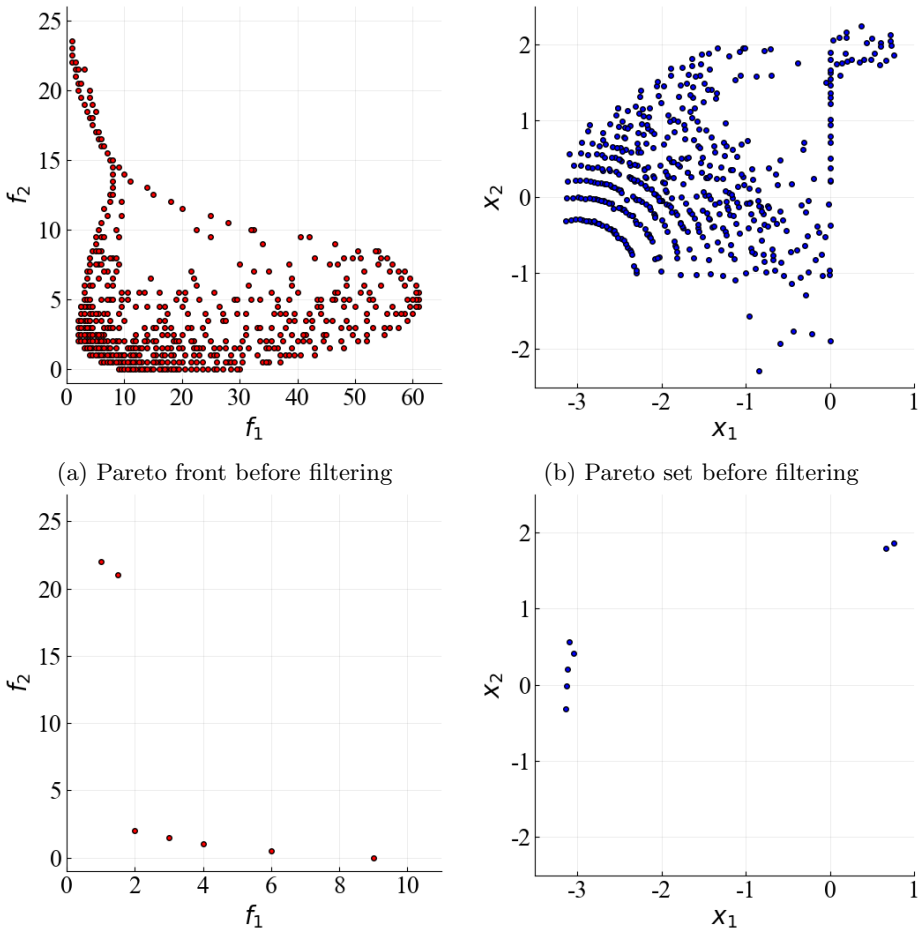
The fact that the picture apparently shows only a few dozens of solutions may be misleading: actually there are 700 fighting for optimality, but many of them happen to be very close to each other and therefore overlap in the figure. Using the standard approach to this problem, that is removing priorities, running NSGA-II on three objectives (with the same parameters as above) and eventually filtering by nondominance for the two primary objectives, leads to the result shown in Figure 8. What at first glance seems a richer front is actually noise. Treating the third objective as important as the other two causes a dramatic loss of accuracy in the search for optimal solutions, because part of the optimization focus is shifted towards the minimization of this objective and dragged away from the minimization of the other (most important) two. Indeed, after filtering the result set obtained by NSGA-II (Figure 8), only 7 out of 700 solutions are left, a very small fraction of the total population. Also, the solutions found by NSGA-II turn out to be further away from the real Pareto frontier (thus worse) than those found by PC-NSGA-II. Basically, this happens because NSGA-II completely ignores the priorities of the objectives, not taking advantage of that precious information. On the other hand, PC-NSGA-II is able to exploit this information, succeeding in finding more accurate solutions.

In conclusion, this and the other benchmarks show that NSGA-II, and in general any non-priority-based optimization algorithm, may not be the best available tool to address problems where priorities matter.



(a) Pareto front (b) Pareto set

Figure 7: Solutions obtained by PC-NSGA-II for MPL3



(a) Pareto front before filtering

(b) Pareto set before filtering

(c) Pareto front after filtering

(d) Pareto set after filtering

Figure 8: Obtained solutions by NSGA-II for MPL3

7.7. Performance evaluation on MPL3

In this section we focus on the performance evaluation inherently to the MPL3 benchmark. Results and considerations are presented following the very same guidelines adopted in the previous two sections.

7.7.1. Mean and Standard deviation

Table 9 reports the mean and the standard deviation of the algorithms performances on the MPL3 benchmark. We observe once again that the PC algorithms perform better than their standard counterparts and the other priority-based approaches. Moreover, PC-NSGA-II turns out to be the overall best performing algorithm this time; NSGA-II appears to work better than MOEA/D on the current benchmark. Indeed, the second best performing algorithm is

NSGA-II without secondaries. However, PC-MOEA/D still gets a good score, being in third position and very close to the second.

Algorithm	Mean	Std
PC-NSGA-II	$\mathbf{0.01}$	$\mathbf{0.02 + 0.01①^{-1}}$
NSGA-II pre	$0.34 + 0.09①^{-1}$	$0.09 + 0.02①^{-1}$
PC-MOEA/D	$\mathbf{0.59 - 0.02①^{-1}}$	$\mathbf{0.58 - 0.02①^{-1}}$
NSGA-II post	$0.65 - 0.08①^{-1}$	$0.16 - 0.07①^{-1}$
MOEA/D pre	$0.89 + 0.05①^{-1}$	$1.4 - 0.01①^{-1}$
MOEA/D post	$1.23 - 0.03①^{-1}$	1.48
Tan et al.	$1.34 - 0.82①^{-1}$	$0.28 - 0.16①^{-1}$
Chang et al.	$38.06 - 7.83①^{-1}$	$3.0 - 0.09①^{-1}$

Table 9: Mean and standard deviation of metric $\Delta(\cdot)$ on MPL3 after 50 repetitions.

7.7.2. Friedman and Iman-Davenport tests

Table 10 reports the algorithms ranks obtained by the Friedman test on MPL3. Moreover, the Friedman test gives a statistic (distributed according to a chi-square with 7 degrees of freedom) of 277.493333 and a P-value of 0. This implies the presence of a statistical separation among some algorithms in the pool. In order to confirm such result, the Iman-Davenport test has been computed. Its statistic (distributed according to a F-distribution with 7 and 343 degrees of freedom) is 187.529974 and the P-value again 0. The rejection of the null hypothesis is confirmed, so we have to determine which algorithms are statistically separated from the best one.

Algorithm	Ranking
PC-NSGA-II	1
NSGA-II pre	2.7
PC-MOEA/D	3.62
MOEA/D pre	4.26
NSGA-II post	4.48
MOEA/D post	5.52
Tan et al.	6.42
Chang et al.	8

Table 10: Algorithms ranked by the Friedman test (MPL3)

7.7.3. Post hoc comparison

The p-values obtained by applying the Holm test are reported in Table 11. Again, it turns out that PC-NSGA-II is statistically separable from all the other approaches.

8. Related Works

In the scientific literature, optimization based on priorities is not a new concept. For example, the work of Fonseca and Fleming is one of the first to raise

i	algorithm	$z = (R_0 - R_i)/SE$	p
7	Chang et al.	14.28869	0
6	Tan et al.	11.063529	0
5	MOEA/D post	9.226411	0
4	NSGA-II post	7.10352	0
3	MOEA/D pre	6.654447	0
2	PC-MOEA/D	5.348053	0
1	NSGA-II pre	3.47011	0.00052

Table 11: Post hoc comparison with $\alpha = 0.05$ and $SE = 0.489$ (MPL3)

awareness about the importance of priority [42] in multiobjective optimization problems. In that paper, they outline different kinds of preference articulation, and even suggest a special comparison operator which takes into account the priority information. Although the given formulation is a very important theoretical result, the article does not describe a full-fledged algorithm tailored for this particular class of problems nor provides any test problem. On the wake of this work, other authors developed priority-based algorithms suitable for different contexts. For instance, Basgalupp et al. [43] tackled the priority problem by means of a preemptive approach, i.e., representing each priority chain as a vector where each entry is associated to a specific objective function. In such work only one single priority chain is taken into account, and different individuals are compared considering, in principle, only the highest priority function. In case of a tie, the evaluation moves to the second most important function, and so on. Tan et al. proposed a new goal programming algorithm, featuring a two-stage domination scheme which ranks individuals on the basis of soft/hard goal and priority specifications [39]. Schmiedle et al. proposed a heuristic which builds a graph out of the found solutions on the basis of priority relations, then assigns fitness values accordingly [44]. Not always the priority is clearly specified by the decision maker: in the case examined by Allmendinger et al., for instance, precedences are due to the objectives having different latencies; decisions on which objective to optimize first depend on their evaluation time, making it similar to a scheduling problem [45]. Other authors, like Chen et al., studied problems where the number of objectives dynamically changes over time [46], but did not specifically deal with precedence relations. A soft prioritization is realized by Chang et al. [40] by means of an objectives dynamic weighting. Here the priority chains are represented by a weighted average of the involved objective functions. In particular, the weights change their values in time, initially favoring the most important objective and gradually shifting the relevance to the lesser ones. Dynamism is also present in [47], where the priority relations change among the objectives, that is the arrows directions within the same priority chain. Such changes are random, even if biased by the true priority relation existing among the objective functions. Another multi-objective problem which seems at first glance similar to the one presented here is bilevel-optimization [48]. In bilevel-optimization, the outer level approximated solution found so far is considered as a fixed parameter for the inner optimization problem. Once the latter is solved, another iteration of the outer level is performed.

Outer and inner problems have their own decision spaces, objective functions, and constraints. For such reasons, bilevel optimization is quite different from PL-MPL-MOP problems in terms of formulation and purpose.

A few more comments on the applicability of some of the previous approaches to our case of study, namely PC-MPL-MOPs. Despite the interesting novelties of Schmiedle et al., this method cannot be used as a yardstick in the current work because it is suited for a class of problems closer to a PL-MPL-MOP rather than a PC-MPL-MOP. A comparison with such priority-handling method is postponed to a future work about PL-MPL-MOPs. The Castro-Gutierrez proposal [47] has not been tested here, even if it is a feasible approach, for three main reasons: i) the core idea is very reminiscent of Chang et al., which is tested instead; ii) the lack of a true and deterministic respect of the objectives priorities; iii) it has been proposed just for one priority chain.

As a general consideration, please observe how the definition of a set of test problems to assess the effectiveness of algorithms on these problems, notably, was not addressed in any of the previous articles. Also, unlike some of those works, here we do not explicitly focus on goal programming. Furthermore, our approach enables handling precedence relations which exist between specific objectives only, and can not be expressed by just assigning them a generic priority value. Finally, in our proposed algorithms, all the objectives are always active and operate *simultaneously* during the optimization, just with a different impact in the overall process.

9. Discussions

In this work we were able to transform lexicographic sub-problems into scalar ones by using the Grossone Methodology. The powerfulness of this scalarization technique is that it does not require any big M constant, to weigh less and less the lower priority objectives, thus we do not need to choose such a critical parameter (an excessively low value could not be sufficient to model the priorities between objectives, while a too high one might lead to numerical instabilities). In addition, we want to clarify once again that our Grossone-based approach can work with other evolutionary optimizers too, like SPEA2, AMGA2, etc. Specifically, we have decided to build upon NSGA-II and MOEA/D because they have no additional parameters other than those of a standard genetic algorithm.

In order to assess the quality of the proposed approach, we compared their performances numerically, evaluating them by means of the indicator $\Delta(\cdot) = \max\{GD(\cdot), IGD(\cdot)\}$; the comparison has been performed by non-parametric statistical analysis.

Finally, note how the proposed approach can be easily parallelized on modern many-cores processors, following the same approach provided in [49], or can be tailored to deal with problems with non-convex Pareto fronts, as done in [50].

10. Conclusions and future works

We have shown in this article how, thanks to the Grossone Methodology, it is possible to solve a class optimization problems where some objectives have priority over some others, specifically PC-MPL-MOPs. In particular, we have seen how the additional information about priority relations among objectives can be exploited to significantly improve the search, guiding it towards higher quality solutions than those provided by conventional non-priority-based multi-objective algorithms. Such algorithms can be used to solve many-objective problems as well, provided that some of the objectives have priority over others and the number of macro-objectives (i.e., the number of chains) is relatively low. In a future work, we will combine NSGA-III and Grossone Methodology to solve massive-objective MPL problems using a many-objective algorithm, similarly to what has been done in this paper, where we have solved a many-objective MPL problem leveraging a multi-objective optimizer.

Acknowledgements

This work has been partially supported by the University of Pisa funded project PRA_2018_81 "Wearable sensor systems: personalized analysis and data security in healthcare". We also want to thank the three anonymous reviewers for their valuable feedbacks, and professor Pietro Ducange for his help in performing the statistical tests.

References

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

[2] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength pareto evolutionary algorithm, TIK-report 103, (2001),doi:10.3929/ethz-a-004284029.

[3] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation 11 (6) (2007) 712–731.

[4] H. Li, K. Deb, Q. Zhang, P. Suganthan, L. Chen, Comparison between MOEA/D and NSGA-III on a set of novel many and multi-objective benchmark problems with challenging difficulties, Swarm and Evolutionary Computation 46 (2019) 104 – 117. doi:https://doi.org/10.1016/j.swevo.2019.02.003.

[5] M. Garza-Fabre, G. T. Pulido, C. A. C. Coello, Ranking methods for many-objective optimization, in: Mexican International Conference on Artificial Intelligence, Springer, 2009, pp. 633–645.

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

[7] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, 2nd Edition, John Wiley & Sons Inc, 2001.

[8] J. Marques-Silva, J. Argelich, A. Graça, I. Lynce, Boolean lexicographic optimization: algorithms & applications, Annals of Mathematics and Artificial Intelligence 62 (3) (2011) 317–343.

[9] S. Khosravani, M. Jalali, A. Khajepour, A. Kasaiezadeh, S. K. Chen, B. Litkouhi, Application of lexicographic optimization method to integrated vehicle control systems, IEEE Transactions on Industrial Electronics 65 (12) (2018) 9677–9686.

[10] Y. D. Sergeyev, Arithmetic of Infinity, Edizioni Orizzonti Meridionali (2nd ed. 2013), Cosenza (Italy), 2003.

[11] M. Cococcioni, M. Pappalardo, Y. D. Sergeyev, Towards Lexicographic Multi-Objective Linear Programming using Grossone Methodology, in: Y. D. Sergeyev, D. Kvasov, F. Dell'Accio, M. Mukhametzhanov (Eds.), Proc. of the 2nd Intern. Conf. "Numerical Computations: Theory and Algorithms", Vol. 1776, AIP Publishing, New York, 2016, p. 090040.

[12] M. Cococcioni, M. Pappalardo, Y. D. Sergeyev, Lexicographic Multi-Objective Linear Programming using Grossone Methodology: Theory and algorithm, Applied Mathematics and Computation 318 (2018) 298–311.

[13] Y. D. Sergeyev, A new applied approach for executing computations with infinite and infinitesimal quantities, Informatica 19(4) (2008) 567–596.

[14] Y. D. Sergeyev, Computations with grossone-based infinities, in: C. Calude, M. Dinneen (Eds.), Unconventional Computation and Natural Computation: Proc. of the 14th International Conference UCNC 2015, Vol. LNCS 9252, Springer, New York, 2015, pp. 89–106.

[15] Y. D. Sergeyev, Numerical infinities and infinitesimals: Methodology, applications, and repercussions on two Hilbert problems, EMS Surveys Math. Sci 4 (2) (2017) 219–320.

[16] Y. D. Sergeyev, Counting systems and the First Hilbert problem, Nonlinear Analysis Series A: Theory, Methods & Applications 72(3-4) (2010) 1701–1708.

[17] Y. D. Sergeyev, D. E. Kvasov, M. S. Mukhametzhanov, On strong homogeneity of a class of global optimization algorithms working with infinite and infinitesimal scales, Communications in Nonlinear Science and Numerical Simulation 59 (2018) 319 – 330. doi:https://doi.org/10.1016/j.cnsns.2017.11.013.

[18] M. Cococcioni, A. Cudazzo, M. Pappalardo, Y. D. Sergeyev, Solving the Lexicographic Multi-Objective Mixed-Integer Linear Programming Problem using Branch-and-Bound and Grossone Methodology, Communications in Nonlinear Science and Numerical Simulation 84 (2020) 105177. doi:https://doi.org/10.1016/j.cnsns.2020.105177.

[19] Y. D. Sergeyev, Solving ordinary differential equations by working with infinitesimals numerically on the Infinity Computer, Applied Mathematics and Computation 219(22) (2013) 10668–10681.

[20] Y. D. Sergeyev, M. Mukhametzhanov, F. Mazzia, F. Iavernaro, P. Amodio, Numerical methods for solving initial value problems on the Infinity Computer, International Journal of Unconventional Computing 12(1) (2016) 3–23.

[21] P. Amodio, F. Iavernaro, F. Mazzia, M. Mukhametzhanov, Y. D. Sergeyev, A generalized Taylor method of order three for the solution of initial value problems in standard and infinity floating-point arithmetic, Mathematics and Computers in Simulation 141 (2017) 24–39.

[22] L. Fiaschi, M. Cococcioni, Numerical Asymptotic Results in Game Theory using Sergeyev's Arithmetic of Infinity, International Journal on Unconventional Computing 14 (2018) 1–25.

[23] L. Fiaschi, M. Cococcioni, Generalizing Pure and Impure Iterated Prisoner's Dilemmas to the Case of Infinite and Infinitesimal quantities, in: Y. D. Sergeyev, D. E. Kvasov (Eds.), Numerical Computations: Theory and Algorithms, Springer International Publishing, Cham, 2020, pp. 370–377.

[24] L. Fiaschi, M. Cococcioni, Non-Archimedean Game Theory: A Numerical Approach, Applied Mathematics and Computation, (2020), *to appear*.

[25] M. P. Hansen, A. Jaszkiewicz, Evaluating the quality of approximations to the non-dominated set, IMM, Department of Mathematical Modelling, Technical Universityof Denmark, 1994.

[26] J. Knowles, D. Corne, On metrics for comparing nondominated sets, in: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), Vol. 1, IEEE, 2002, pp. 711–716.

[27] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. G. Da Fonseca, Performance assessment of multiobjective optimizers: An analysis and review, IEEE Transactions on evolutionary computation 7 (2) (2003) 117–132.

[28] E. Zitzler, L. Thiele, Multiobjective optimization using evolutionary algorithmsa comparative case study, in: International conference on parallel problem solving from nature, Springer, 1998, pp. 292–301.

[29] O. Schutze, X. Esquivel, A. Lara, C. A. C. Coello, Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization, IEEE Transactions on Evolutionary Computation 16 (4) (2012) 504–522.

[30] D. A. Van Veldhuizen, Multiobjective evolutionary algorithms: classifications, analyses, and new innovations, Tech. rep., Air Force Institute of Technology Wright Patterson AFB, OH, USA (1999).

[31] C. A. C. Coello, M. R. Sierra, A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm, in: Mexican International Conference on Artificial Intelligence, Springer, 2004, pp. 688–697.

[32] M. R. Sierra, C. A. C. Coello, A new multi-objective particle swarm optimizer with improved selection and diversity mechanisms, Technical Report of CINVESTAV-IPN.

[33] H. Ishibuchi, H. Masuda, Y. Tanigaki, Y. Nojima, Modified distance calculation in generational distance and inverted generational distance, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2015, pp. 110–125.

[34] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: a case study on the cec2005 special session on real parameter optimization, Journal of Heuristics 15 (6) (2009) 617.

[35] I. Triguero, S. González, J. M. Moyano, S. García López, J. Alcalá Fernández, J. Luengo Martín, A. Fernández Hilario, J. Díaz, L. Sánchez, F. Herrera, et al., Keel 3.0: an open source software for multi-stage analysis in data mining.

[36] D. A. Van Veldhuizen, Multiobjective evolutionary algorithms: Classifications, analyzes, and new innovations, Air Force Inst. Technol., Dayton, OH, Tech. Rep. AFIT/DS/ENG/99-01.

[37] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable test problems for evolutionary multiobjective optimization, in: Evolutionary Multiobjective Optimization, Springer, 2005, pp. 105–145.

[38] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, S. Tiwari, Multiobjective optimization test instances for the cec 2009 special session and competition, University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report 264.

[39] K. C. Tan, E. F. Khor, T. H. Lee, R. Sathikannan, An evolutionary algorithm with advanced goal and priority specification for multi-objective optimization, Journal of Artificial Intelligence Research 18 (2003) 183–215.

[40] P.-C. Chang, J.-C. Hsieh, S.-G. Lin, The development of gradual-priority weighting approach for the multi-objective flowshop scheduling problem, International Journal of Production Economics 79 (3) (2002) 171–183.

[41] C. Poloni, Hybrid GA for multi objective aerodynamic shape optimisation, in: G. Winter, J. Periaux, M. Galan, P. Cuesta (Eds.), Genetic Algorithms in Engineering and Computer Science, John Wiley & Sons Ltd, 1995, pp. 397–415.

[42] C. M. Fonseca, P. J. Fleming, Multiobjective optimization and multiple constraint handling with evolutionary algorithms Part I: A unified formulation, IEEE Transactions on Systems, Man, and Cybernetics 28 (1) (1998) 26–37.

[43] M. Basgalupp, A. de Carvalho, R. Barros, D. Ruiz, A. Freitas, Lexicographic Multi-Objective Evolutionary Induction of DecisionTrees, International Journal of Bio-Inspired Computation 1 (1-2) (2009) 105–117.

[44] F. Schmiedle, N. Drechsler, D. Große, R. Drechsler, Priorities in multi-objective optimization for genetic programming, in: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, Morgan Kaufmann Publishers Inc., 2001, pp. 129–136.

[45] R. Allmendinger, J. Handl, J. Knowles, Multiobjective optimization: When objectives exhibit non-uniform latencies, European Journal of Operational Research 243 (2) (2015) 497–513.

[46] R. Chen, K. Li, X. Yao, Dynamic multiobjectives optimization with a changing number of objectives, IEEE Transactions on Evolutionary Computation 22 (1) (2018) 157–171.

[47] J. Castro-Gutiérrez, D. Landa-Silva, J. Moreno-Pérez, Dynamic lexicographic approach for heuristic multi-objective optimization, in: Proceedings of the Workshop on Intelligent Metaheuristics for Logistic Planning (CAEPIA-TTIA 2009)(Seville (Spain)), 2009, pp. 153–163.

[48] X. He, Y. Zhou, Z. Chen, Evolutionary bilevel optimization based on covariance matrix adaptation, IEEE Transactions on Evolutionary Computation 23 (2) (2019) 258–272.

[49] M. Cococcioni, mspMEA: the microcones separation parallel multiobjective evolutionary algorithm and its application to fuzzy rule-based ship classification, in: R. Abielmona, R. Falcon, N. Zincir-Heywood, H. Abbass (Eds.), Recent Advances in Computational Intelligence in Defense and Security, Vol. 621, Springer Series on Studies in Computational Intelligence, 2015, pp. 445–465.

[50] M. Cococcioni, P. Ducange, B. Lazzerini, F. Marcelloni, A new multi-objective evolutionary algorithm based on convex hull for binary classifier optimization, in: in Proc. 2007 IEEE Congress on Evolutionary Computation (IEEE-CEC'07), 2007, pp. 3150–3156.