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

Leonardo Lai, Lorenzo Fiaschi, Marco Cococcioni, *Senior Member, IEEE*, and Kalyanmoy Deb, *Fellow, IEEE*

***Abstract*—This paper concerns the study of Mixed Pareto-Lexicographic Multi-objective Optimization Problems where the objectives must be partitioned in multiple *priority levels*. A priority level (PL) is a group of objectives having the same importance in terms of optimization and subsequent decision-making, while between PLs a lexicographic ordering exists. A naive approach would be to define a multi-level dominance relationship and apply a standard EMO/EMaO algorithm, but the concept does not conform to a stable optimization process as the resulting dominance relationship violates the transitive property needed to achieve consistent comparisons. To overcome this, we present a novel approach which merges a custom non-dominance relation with the *Grossone* methodology, a mathematical framework to handle infinite and infinitesimal quantities. The proposed method is implemented on a popular multi-objective optimization algorithm (NSGA-II), deriving a generalization of it called by us PL-NSGA-II. We also demonstrate the usability of our strategy by quantitatively comparing the results obtained by PL-NSGA-II against other priority and non-priority-based approaches. Among the test cases, we include two real-world applications: one 10-objective aircraft design problem and one 3-objective crash safety vehicle design task. The obtained results show that PL-NSGA-II is more suited to solve lexicographical many-objective problems than the general purpose EMaO algorithms.**

***Index Terms*—Multi-Objective Optimization, Lexicographic Optimization, Evolutionary Computation, Genetic Algorithms, Numerical Infinitesimals, Grossone Methodology**

I. Introduction

Evolutionary algorithms [1] quickly emerged as one of the most effective strategies to face many real problems requiring the simultaneous optimization of two or more functions, mainly due to their ability to manage and generate a diverse group of trade-off solutions, instead of focusing on a single one. However, researchers have identified critical issues that arise when the number of objectives grows, generally over three or four [2, 3]. The most troublesome aspects of dealing with these so-called many-objective optimization problems include: i) ineffectiveness of the standard Pareto dominance; ii) need for a significantly larger population, due to the possibly increased dimensionality of the efficient set; iii) computational inefficiency of the diversity estimation functions; and iv) ineffectiveness of canonical recombination operators. To overcome these problems, researchers have proposed various approaches: decomposition [4], Pareto dominance alternatives [5], custom recombination operators [6], new diversity management mechanisms [7] or even full-fledged many-objective specific frameworks [8]–[10].

While the increase in dimension raises computational challenges, the multi- or many-objective optimization problems originate from practical considerations and often not all the objectives are of equal preference to decision makers. Indeed, the objectives can be likely grouped on the basis of their preference: a group of one or more objectives is expected to be more important than other objective groups, although at the formulation stage none of the objectives can be completely eliminated based on their preference level. A recent problem from an automobile industry having six objectives was clearly stated – the weight objective was most important to minimize, but designers were also interested in solutions having a trade-off in other five objectives [11]. In such situations, instead of treating all the given objectives as equally important, which is the case of the most common evolutionary multi-objective optimization (EMO) algorithms, the priority information shall be taken into account during the optimization phase for a number of reasons:

- in the presence of preference information, a more focused search can be performed to find the part of the entire Pareto-optimal set which corresponds to best solutions of the preferred objectives. This makes the whole optimization effort more computationally efficient too;
- by focusing on the trade-off among similar priority objectives, more insights among the important objectives can be obtained, rather than blurring the analysis with all objectives.

The domain of our interest are *Mixed Pareto-Lexicographic Multi-objective Optimization Problems* (abbreviated as MPL-MOPs), a broad class of problems that assume precedence between some objectives, differing in how this precedence is distributed. The subclass of *PC-MPL-MOPs (Priority Chains MPL-MOPs)* is characterized by a clear precedence structure among objectives belonging to a chain. A scalarization of objectives within a chain is obtained using the Grossone methodology. Thus each chain produces a single macro-objective; the macro-objectives are then optimized in the Pareto sense. More details are provided in [12]. On the contrary, in the Priority Level MPL-MOPs (PL-MPL-MOPs) there are no such chains, but the objectives are grouped into multiple priority levels. Objectives within a level are equally important and must be handled in the Pareto sense, while groups of objectives forming higher priority levels are lexicographically more important than any lower-level group. While handling PC-MPL-MOPs is quite straightforward, PL-

MPL-MOPs require special attention, which motivates the present study. As in the previous work, we resort on the *Grossone Methodology* to deal with infinite and infinitesimal weights by means of which assign priority to the levels (rather than within the chains).

In Section II, we outline the mathematical description of the problems of our interest, then we point out the weaknesses of the standard approaches. A brief insight into the Grossone Methodology is given in Section III, to provide the reader with the essential knowledge to understand our idea. Section IV presents a special dominance relation which fits very well with the nature of PL-MPL-MOPs. This result is exploited in Section V to build a full PL-MPL-specific optimization algorithm, inspired by NSGA-II but actually generalizing it. Section VI presents four PL-MPL test problems, through which we demonstrate the working of our proposed algorithm. Such problems also serve as benchmarks to assess the performance of PL-NSGA-II with respect to the standard priority and non-priority-based approaches. The study ends in Section VII with conclusions and future works.

II. Mixed Pareto-Lexicographic Problems

The generic formulation of a MPL-MOP is:

$$\begin{aligned} \text{Minimize} \quad & \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})\}, \\ \text{Subject to} \quad & \mathbf{x} \in \Omega \\ & \wp\left(f_1, f_2, \ldots, f_m\right) \end{aligned}$$

where m is the number of objective functions, Ω is the feasible variable space, also called *decision space* or *search space* or *design space*, and $\wp(\cdot)$ is a generic priority distribution among the objectives. In the absence of $\wp(\cdot)$, the optimization is implicitly assumed to be of Paretian type, a class of problems that are usually solved by means of EMO algorithms, which return multiple trade-off solutions, approximated or not, after a finite number of steps.

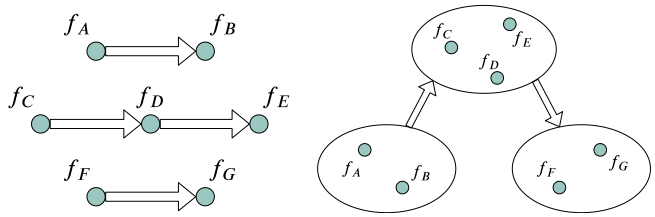
At the opposite of Pareto optimization, another well known corner case of MPL-MOPs is represented by lexicographic problems, where $\wp(\cdot)$ arranges the objectives according to a total order of importance. A common way to denote them is:

$$\begin{aligned} \text{lexmin} \ & \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})\} \\ \text{s.t} \ & \mathbf{x} \in \Omega \end{aligned}$$

This expression indicates the minimization of the objectives $\{f_1, f_2, \ldots, f_m\}$ ranked lexicographically: f_1 is infinitely more important than f_2, which in turn has priority over f_3, and so on.

The more general case where $\wp(\cdot)$ can be whatever function (i.e., a generic MPL-MOP) has not received enough research attention, despite being potentially applicable to many real-world scenarios, such as [11]. Indeed, algorithms which are able to take into account priority would definitely help in finding better quality solutions, as opposed to Pareto ones that struggle in both convergence and quality, and lexicographic ones which cannot be applied at all. There are several studies in the literature proposing ad-hoc approaches to deal with generic priority relations, such as linear scalarization [1] or ϵ-constrained methods [1], other scalarization strategies can be found in [13]. Although more effective than general purpose methods, these algorithms are not designed to cope with MPL-MOPs in a proper way, manifesting significant drawbacks indeed. Even if scalarized problem are easier to solve, the weights should be chosen carefully, otherwise the resulting solution may be wrong or very inaccurate. Moreover, such approaches typically find only one solution at a time, with dramatic consequences on the overall computation time.

So far, researchers have identified two relevant but poorly studied subclasses of MPL-MOPs: i) objective precedence arranged as chains [12], namely PC-MPL-MOP, represented in Fig. 1a; ii) objective precedence among groups of objectives, the topic of this work, shown in Fig. 1b. In a PC-MPL-MOP, multiple chains exist and each objective belongs to one of them. For example, the objectives f_A and f_B are part of the same chain. Within each chain, there exists a precedence structure, which is then handled by a single scalarization function using the Grossone methodology. The arrow indicates that f_A is lexicographically more important than f_B. For this chain, a single scalarized function is formed using both f_A and f_B. The scalarized objectives from different chains are then optimized in a Pareto sense. In the example case, the process will end up solving a three-objective optimization problem to find the final solutions. The formal definition of PC-MPL-MOP is given in [12]. On the contrary, our proposed



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

Fig. 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.

PL-MPL-MOPs allow one to consider groups of conflicting objectives ordered by priority. For instance, among the seven objectives of Fig 1b, a priority structure can be defined as illustrated: f_A and f_B are in the same priority level and f_C, f_D and f_E are also in the same priority level, but the former objectives are lexicographically more important than the latter. The objectives f_F and f_G are in the same priority level, but they have worse priority than the other ones. Clearly, the two problem descriptions are different from each other, so their final solutions cannot be compared in any sense.

A. Problems Partitioned by Priority Levels: PL-MPL-MOP

PL-MPL-MOPs, are problems in which objectives are grouped in levels and lexicographic priority exists among them, as reported in Figure 1b. More formally:

Definition 1 (PL-MPL-MOP). *An MPL-MOP satisfying the conditions below is called "Mixed Pareto-Lexicographic Multi-objective Optimization Problem partitioned by Priority Levels" (PL-MPL-MOP):*

- *Some groups of objectives have clear precedence over some others (in a lexicographic sense), and*
- *Within each group, objectives can not be compared to each other on the basis of importance (in the Pareto sense).*

The general mathematical structure is:

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

where p is the number of priority levels and $f_i^{(j)}$ is the i_{th} objective in the j_{th} level of importance.

In a PL-MPL-MOP, the Pareto-optimal solutions of the objectives in the first level form the decision space for the Pareto optimization of the objectives in the second level, and then again the latter solutions become the domain for the third one, and so on for every level. PL-MPL-MOPs are structurally quite similar to purely lexicographic problems, except that each element of the sequence is now a multi-objective problem on its own, instead of single-objective function in the original lexicographic sense [14]. Of course, our above formulation covers the original case as well, since a purely lexicographic problem is nothing but a particular case with one objective function in each class. Similarly, the classical formulation of Pareto MOPs occurs when there is only one priority level.

B. A first example of a real-world PL-MPL-MO Problem

The automotive crashworthiness problem [15] consists in finding the best car designs balancing performance and safety. The original benchmark aims at maximizing the vehicle acceleration while minimizing mass and toe-board intrusion. Even if the three objectives are all necessary for a consistent design, one may assume that they are not all equally important for a firm, and perhaps more interestingly, that their mutual relation can vary from one firm to another.

For instance, a company may be interested in designing high performing cars, in which case it would put more emphasis on the first two objectives (mass and acceleration) than on the safety one. Therefore, during the optimization the toe-board intrusion should play a role in discriminating among the high speed cars, rather than on the whole pool of possible vehicles. A possible way to cope with the industrial requests of this kind of firms could be the reformulation of the problem in a PL-fashion, as illustrated in Equation (1). The model splits the original 3-objective optimization task in two PLs: the first (and high-priority) one contains the mass and the acceleration, while the second (and low-priority) one introduces the safety information still keeping in mind the goal of maximizing the acceleration.

$$\begin{aligned} &\text{lexmin} \left[\min \begin{pmatrix} \texttt{mass}(\mathbf{x}) \\ -\texttt{accel}(\mathbf{x}) \end{pmatrix}, \min \begin{pmatrix} \texttt{toe}(\mathbf{x}) \\ -\texttt{accel}(\mathbf{x}) \end{pmatrix} \right] \\ &\text{s.t. } \mathbf{x} \in [1,\, 3]^5. \end{aligned} \tag{1}$$

On the contrary, other firms could prefer investing in the design of safe cars. Then, among those who are equally Pareto efficient, they shall seek for high performing ones. Without entering again in the details, this second model would be the following:

$$\begin{aligned} &\text{lexmin} \left[\min \begin{pmatrix} \texttt{mass}(\mathbf{x}) \\ \texttt{toe}(\mathbf{x}) \end{pmatrix}, \min \begin{pmatrix} \texttt{toe}(\mathbf{x}) \\ -\texttt{accel}(\mathbf{x}) \end{pmatrix} \right] \\ &\text{s.t. } \mathbf{x} \in [1,\, 3]^5. \end{aligned}$$

In the rest of the paper we will often make reference to the two sub-problems above, especially the first one. In the end, we will also solve the first one, using the new algorithms proposed in this paper.

C. Need of a New Approach for PL-MPL-MOPs

In this subsection, we aim to point out the limitations of strategies based on standard optimization techniques when solving PL-MPL-MOPs. In the absence of tools that are able to concurrently perform Pareto and lexicographic optimization, any viable approach necessarily requires these two to run separately in distinct stages. How these steps are interleaved may vary, leading to different options. We identify here two strategies that are worth of consideration.

The first way is to simply ignore the precedence relations during the multi-objective search, thus Pareto optimizing all the objectives together, indistinctly. Only afterwards the priorities are considered, by filtering the solutions that are group-lexicographically optimal. We refer to this approach as *postfiltering*. Clearly, such an approach is computationally inefficient, as many more solutions need to be found by the original EMO algorithm in order to have a sizable number of solutions at the end.

The second approach uses a multi-objective optimizer solving the problem considering only the first level, and it is motivated by the observation that primary objectives have the highest impact during the optimization process. At the end of it, the other levels information is also taken into account to rank the obtained solutions. The main weakness of such method, called here *prefiltering*, is the null impact of the less important objectives in the optimization process. However, as opposed to the postfiltering method, it has the benefit that the objectives of the first level play a dominant role in the decision making.

In Section V, we propose a new algorithm that is able to overcome these difficulties, in which the optimization is not split into sub-tasks, and all the objectives are used during the whole process, still privileging those belonging to higher levels over the less important ones. The approach uses a recent development in handling infinite and infinitesimal numbers, which we briefly discuss in the next section.

III. The Grossone Methodology

A numerical methodology for computations including infinite and infinitesimal quantities has been proposed and developed by Sergeyev (see [16] and references therein). In short, the key idea is the introduction of *Grossone*, denoted by ①, an infinite number which serves as base for a new numeral system which makes it possible to express infinite

and infinitesimal quantities in a way which lets one easily manipulate them numerically.

The Grossone Methodology (GM) has already been successfully applied to different optimization problems [12, 17]–[20], other than many other mathematical fields such as Ordinary Differential Equations [21]–[23], Game Theory [24, 25], and Decision Theory [26]. Within it, the four basic operations between two finite, infinite or infinitesimal numbers are fairly intuitive and easy to implement. The next few lines are examples of such operations, just to get a basic understanding of their arithmetical behavior:

$$① \cdot (① + 2) = ①^2 + 2①, \qquad 0 < \frac{1}{①} = ①^{-1} < ①^0 = 1 < ①^1 = ①,$$

$$\frac{-10.0①^3 + 16.0 + 42.0①^{-3}}{5.0①^3 + 7.0} = -2.0 + 6.0①^{-3}.$$

The operators also inherit all the standard properties of their counterparts in $\mathbb{R}$, like commutativity, associativity, and existence of inverse. Numerical values composed by finite, infinite and infinitesimal components together represent the general case; a numeral c in the GM, called *gross*-scalar, is indicated using the notation:

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

where $m, k \in \mathbb{N}$, the exponents p_i and the digits $c_{p_i} \neq 0$ are called *gross*-powers and *gross*-digits, respectively. For the remaining of the work, we assume both *gross*-powers and *gross*-digits are finite and real-valued. It can be observed that finite numbers are represented by numerals with highest *gross*-power equal to zero, e.g., $-6.2 = -6.2①^0$, while infinite and infinitesimal quantities correspond to positive and negative highest *gross*-powers, respectively. A *gross*-scalar may contain components of different magnitudes: just to make an example, the *gross*-scalar $12.5①^{3.2} - 8.7①^{0.4} - 4.4①^{-9}$ is infinite, is made of two infinite parts and an infinitesimal one.

It is possible to compare two *gross*-scalars by the operator $<$, which induces a total ordering. The comparison definition is very intuitive; *gross*-powers are scanned by descending order, and the relative *gross*-digits compared: if the *gross*-powers are different or they are equal but the *gross*-digits different, then the biggest corresponds to the largest *gross*-scalar, otherwise the procedures goes on with the next (smaller) couple of *gross*-powers. The total order turns out to be of crucial importance in the remain of the work.

A. Enhancing PL-MPL problems by means of Grossone

The idea of lexicographic relationships indicates that one class is infinitely more important than another one. Hence, ① stands as a viable way to represent such relationship in a single algorithm, instead of splitting the overall problem in multiple stages. A clear example is the case of MPL-MOPs with priority chains, as can be evinced in previous work [12].

On the other hand, an approach to solve problems involving PLs rather than chains, is still lacking. With the purpose of filling this gap, below we propose the following reformulation of PL-MPL-MOPs defined in Definition 1:

$$\min \left[\begin{pmatrix} f_1^{(1)}(\mathbf{x}) \\ f_2^{(1)}(\mathbf{x}) \\ \vdots \\ f_{m1}^{(1)}(\mathbf{x}) \end{pmatrix} + ①^{-1} \begin{pmatrix} f_1^{(2)}(\mathbf{x}) \\ f_2^{(2)}(\mathbf{x}) \\ \vdots \\ f_{m2}^{(2)}(\mathbf{x}) \end{pmatrix} + \ldots + ①^{1-p} \begin{pmatrix} f_1^{(p)}(\mathbf{x}) \\ f_2^{(p)}(\mathbf{x}) \\ \vdots \\ f_{mp}^{(p)}(\mathbf{x}) \end{pmatrix} \right].$$

In this representation, the priority information is added to the problem by means of the *gross*-powers, which sharply separate the PLs. Without going into the mathematical details, the optimization problem can be thought has a minimization problem (and not lexicographic minimization) of the PLs, whose contribution is weighted in accordance to their priority. The level with the highest priority is weighted by $①^0$ (=1, omitted), the second one by $①^{-1}$, and so on until no more PLs remain, thereby giving infinitely more importance to the higher levels. Therefore, the Grossone-oriented rewriting of the problem in (1), henceforth called PL-Crash, is

$$\begin{aligned} \min \left[\begin{pmatrix} \texttt{mass}(\mathbf{x}) \\ -\texttt{accel}(\mathbf{x}) \end{pmatrix} + ①^{-1} \begin{pmatrix} \texttt{toe}(\mathbf{x}) \\ -\texttt{accel}(\mathbf{x}) \end{pmatrix} \right] \\ \text{s.t. } \mathbf{x} \in [1,\, 3]^5. \end{aligned}$$

Given this Grossone-based reformulation of the problem, we need of one more ingredient to propose a PL extension of an EMO algorithm for solving PL-MPL-MOPs. Such ingredient is a novel definition of dominance; it also leverages the GM and it is presented extensively in the next section. After that, it will become clear how the GM can be exploited to improve some fundamental routines of the standard EMO algorithms to handle PL-MPL-MOPs in an effective way.

IV. Handling Precedence Relationships in PL-MPL-MOPs

In this section, we propose a mechanism to exploit precedence relationships to guide the search towards better optima. In essence, it consists in a new priority-based ranking system for the solutions.

A. A New Definition of Dominance: PL-dominance

When coping with PL-MPL-MOPs, a crucial step is to provide a suitable re-definition of Pareto dominance, since it is clear that the original Pareto-optimality is not sufficient as it completely ignores any priority information. Defining an appropriate *PL-dominance* is not trivial, as demonstrated by the following definition, which may seem reasonable and straightforward but actually does not work:

Definition 2 (PL-Dominance (wrong)). *Given two solutions A and B, A is said to "PL-dominate" B if it exists a priority level i such that A Pareto-dominates B on the objectives of priority i and A and B are non-dominated in each priority level higher than i.*

The issue in this definition is that, mathematically speaking, this binary relation lacks of the transitivity property. In other words, there may exists three elements A, B, C such that A dominates B and B dominates C, but C dominates A. It may

not be immediately evident, so the following counterexample is provided as a proof.

Consider these three solutions A, B, C for a generic PL-MPL-MOP with two primary and two secondary objectives, all to be minimized:

$$A = \binom{2}{5} + ①^{\cdot 1}\binom{2}{2},\ B = \binom{4}{3} + ①^{\cdot 1}\binom{7}{6},\ C = \binom{1}{4} + ①^{\cdot 1}\binom{9}{8}.$$

With reference to Def. 2, there is a dominance-loop since:

- A PL-dominates B because, although they are non-dominated for the primary objectives, A performs better than B on the secondaries.
- B PL-dominates C for the same reason.
- C PL-dominates A because it does better on the primary objectives.

Our research work eventually led to another definition of dominance, which works both in theory (it is loop-free) and also in practice, as shown by experiments. The idea is to extend the concept of *non-dominated fronts*. According to Pareto dominance, a front is defined as a set of solutions which do not dominate each other, but are dominated by those of previous fronts and, in turn, dominate those in the subsequent fronts. In our new definition, we introduce the concept of "subfronts" which generalize the idea of fronts and consist in nested fronts generated by the priority. The procedure to partition the population into subfronts is described hereafter: for some aspects, it resembles the preference-based ranking scheme proposed by Fonseca and Fleming [27]. Our approach is tailored to address the structure of PL-MPL-MOPs.

First, considering only the objectives with the highest priority, a set of non-dominated fronts is determined as usual, according to Pareto dominance. Then, these fronts are taken individually (one by one), and each is further split on the basis of the objectives of the second highest priority level, determining new groups of (sub)fronts. The same procedure is iterated recursively on the newly defined subfronts, until there are no more priority levels left. The whole population is eventually partitioned in a hierarchy of subfronts, where the term "subfronts" indicates the groups obtained after every splitting. GM comes in handy, letting one uniquely identify any subfront within the hierarchy by means of a *gross*-scalar index. For instance, F_i where $i = 2①^{0} + 7①^{\cdot 1} + 3①^{\cdot 2}$, denotes all the solutions in the front 2 for the primary objectives, front 7 for the secondary objectives (within the individuals in the front 2 for the primaries) and front 3 for the tertiaries (among those with primary front 2 and secondary front 7).

Given the points and their subfronts, we propose an improved definition of PL-dominance (indicated by $\prec^*$) :

Definition 3 (PL-Dominance (correct)). *Given two solutions A and B, belonging to the subfronts F_i and F_j, respectively, A "PL-dominates" B if and only if i is strictly smaller than j: $A \prec^* B \iff i < j$.*

It is worth noting the *a posteriori* nature of the definition: it is not always possible to tell whether two elements are dominated or not before assigning a fitness rank to the whole population. In other words, the non-dominance relation is not just a function of two arguments (solutions), but has a global dependency on all the other individuals.

Let us go back to the counterexample we used to invalidate the Definition 2. Assuming for simplicity that there are no other elements in the population, and adopting Definition 3, B and C turn out to be in the first primary front, while A is in the second one (C Pareto dominates A at first level). Within the first front, B lies in the first subfront, C in the second. Therefore, while $B \prec^* C$ and $C \prec^* A$, it is no longer true that A dominates B. In the next proposition we claim the consistency of *PL-Dominance*.

Proposition. *PL-dominance in Definition 3 is transitive.*

Proof. Let $\mathcal{S}$ be a set of distinct solutions already partitioned in subfronts. Following the notation above, we indicate the partitioning in subfronts as follows: $\forall\, X \in \mathcal{S},\ X \in F_x$. In order to demonstrate that $\prec^*$ is transitive, the following must hold: $\forall\, X, Y, Z \in \mathcal{S},\ X \prec^* Y \wedge Y \prec^* Z \Rightarrow X \prec^* Z$. One can show that it is always true by repeatedly applying Definition 3. Indeed, $X \prec^* Y \wedge Y \prec^* Z \Rightarrow x < y \wedge y < z \Rightarrow x < y < z$. □

B. PL-Dominance in PL-Crash Problem

For a better comprehension of how PL-Dominance works, the PL-Crash problem in (1) is considered here. Assume one seeks for the optimal Pareto front; in accordance with Definition 3, it shall consists of all the solution falling in the subfront indexed by the *gross*-scalar $1 + 1①^{\cdot 1}$. This means that among those solutions which are Pareto efficient considering the objective in the first PL (mass and acceleration), the ones that are truly optimal are also Pareto efficient considering acceleration and toe-board intrusion only.

This request affects the optimal three-dimensional front of the original problem (given in [8]) as described in the next lines. First, all the solutions are projected on the mass-acceleration plane and only the Pareto efficient survives. Figure 2 reports this process: the green dots indicate the individuals which are preserved and therefore form the parent front, the one indexed by the *gross*-scalar 1. Then, the survived solutions are projected on the toe-acceleration plane (second priority level, Figure 3), and only those which are non-dominated also there can be considered truly optimal. Such individuals, highlighted in red in Figure 4, are the only ones which belong to the subfront $1 + 1①^{\cdot 1}$, since they are the non-dominated solutions (considering the objectives in the second PL) of the non-dominated solutions (this time, considering the objectives in the first PL).

If one wants to retrieve the sub-optimal solutions belonging to the $1 + 2①^{\cdot 1}$ subfront (assuming it is not empty), s/he should discard the solutions just found and repeat the procedure above. On the other hand, if one wants the solutions in the $2 + ①^{\cdot 1}$ subfront, s/he should eliminate all the solutions in the front 1 (regardless the infinitesimals), and again repeat the procedure just illustrated. Figure 5 shows the solutions of the PL-MPL-MOP problem (red points) in the entire three-dimensional objective space (marked with blue points). A little investigation will reveal that no known domination check on the entire objective space will reveal the red points. Only with our proposed PL-dominance approach, the true priority-based optimal solutions can be obtained.

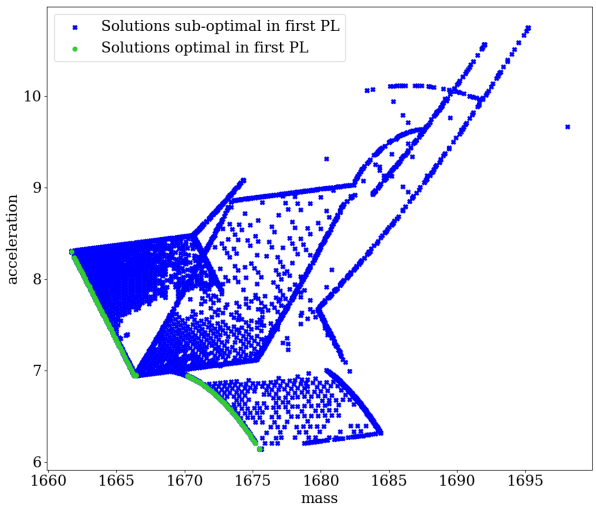


Fig. 2: Original optimal solutions on the mass-acceleration plan: in green the optimal solutions from the first PL.

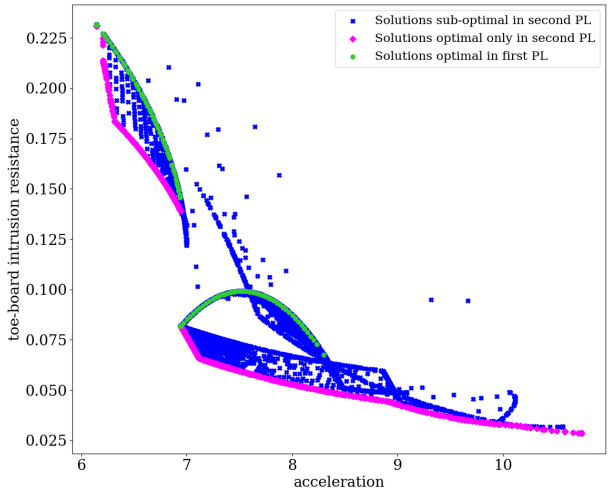


Fig. 3: Original optimal solutions on the toe-acceleration plan: in magenta the Pareto-optimal solutions of the second PL, in green the optimal ones from the first PL.

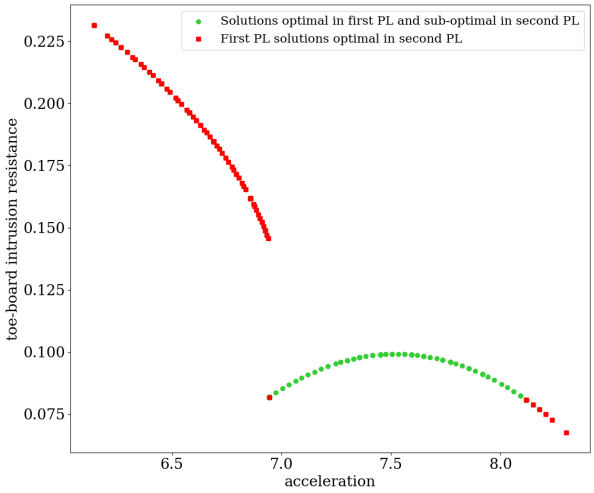


Fig. 4: Original optimal solutions on the toe-acceleration plan: in red the optimal solutions for both the PLs.

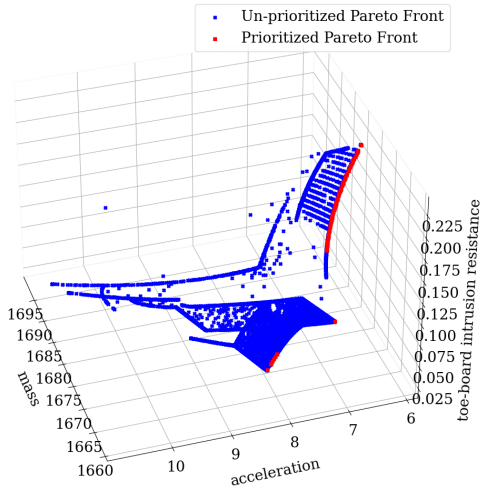


Fig. 5: Effect of the priority on the 3D Pareto front. Only the red solutions are truly optimal.

It is worth to stress that the postponed filtering just illustrated works only if the whole front of the non-prioritized problem is known. Such task rapidly becomes extremely difficult and very inefficient as the problem dimension grows, as already discussed in Section II-C. To overcome these issues, in the next section we introduce a generalization of NSGA-II, enhanced with GM and PL-Dominance in order to transparently and directly cope with any PL-MPL-MOP.

V. Generalized NSGA-II

The new definition of PL-dominance paves the way for a generalization of ranking-based EMO algorithms to solve PL-MPL problems. Here, we do so only for NSGA-II, yet nothing prevents the very same approach to be exploited in other frameworks as well. NSGA-II core operations are: (i) assignment of a non-dominance rank to each solution; (ii) sorting the population by rank; (iii) best-to-worst accommodation of the fronts in the next population, until a too large to fit one is found; (iv) filling the next population with the solutions of the last front according to their crowding distance.

The first two steps, together, are efficiently carried out by the NSGA-II subprocedure called *fast_nondominated_sort*, while the fourth mainly leverages on the routine *crowding_distance_assignment*. Since they are not designed to do anything special when some objectives have priority over others, we improved them to increase their effectiveness when solving PL-MPL-MOPs, obtaining their PL extension. The pieces are eventually put together, creating *PL-NSGA-II*, our variant of NSGA-II specifically adapted for PL-MPL-MOPs.

A. PL fast non-dominated sort and PL crowding distance

In PL-NSGA-II, the solutions are split into subfronts on the basis of the PL-Dominance, which by definition makes two solutions less likely to be non-dominated each other than NSGA-II does. As a consequence, *PL_fast_nondominated_sort* is

expected to return a reasonably larger number of smaller-sized fronts, that is a positive thing considering that the main weakness of many-objective algorithms is indeed having large sets of non-dominated solutions, as mentioned earlier. This difficulty is alleviated by taking into account the priority levels of the objectives. A visual example of the rank assignment procedure is outlined in Figure 6. Refer to Algorithm 1 for a more precise description, in the form of a pseudocode.

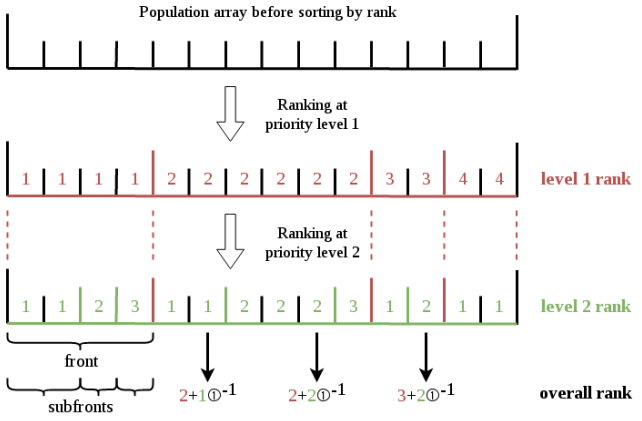


Fig. 6: An example of ranking procedure based on Definition 3.

The proposed *PL_crowding_distance_assignment* is described in Algorithm 2, and it basically consists in a *gross*-weighted sum of the standard crowding distances computed within every priority level. In practice, it follows these steps: first, the standard crowding distance is computed for the primary (read most important) objectives and weighted by $①^0$; then, the same happens considering the second level objectives only, with weight $①^{-1}$; this is repeated for every level, decreasing the *gross*-weight after each step; finally, all the partial results are added together to form a *gross*-scalar, representing the PL crowding distance.

Since *PL_crowding_distance_assignment* is computed within each subfront, and subfronts tend to be generally smaller with respect to *NSGA-II* fronts because of the new ranking system, we can observe that in *PL-NSGA-II* the non-dominated sorting algorithm carries more weight than the crowding distance one. A shift in the delicate balance between convergence and diversity preservation may look questionable, but it can be justified by the following consideration. Diversity preservation mechanisms exist to keep a certain degree of spread among the best solutions. With the addition of extra information to the problem, namely the priorities, we have more means to tell which solutions are better than which, therefore it is easier to circumscribe a smaller set of best solutions. Such an increase in the ability to estimate the quality of solutions partially alleviates the need to maintain a large diversity between them because, from the perspective of a decision maker, it may be preferable to have a narrow set of truly optimal solutions rather than a larger group of diversified but potentially sub-optimal ones. That being said, the PL crowding distance still plays a crucial role after all, and does a decent job at spreading the solutions across the efficient set, especially nearer to the convergence.

Algorithm 1 Priority Levels fast non-dominated sort.

```
/* This function is recursive */
/* P is the population, lvl is the current level to consider */
procedure PL_FAST_NONDOMINATED_SORT(P, lvl)
    /* Base case of recursion */
    if lvl < min_lvl then return
    /* The first iteration also initializes all ranks to 0 */
    if lvl == 0 then
        for all p ∈ P do
            p_rank = 0
    /* Ranking within the priority level to determine subfronts */
    F^(lvl) = fast_nondom_sort_in_level(P, lvl)
    /* Repeat for every subfront found */
    for all F_i ∈ F^(lvl) do
        /* In the next priority level */
        F_i = PL_fast_nondominated_sort(F_i, lvl-1)
```

```
procedure FAST_NONDOM_SORT_IN_LEVEL(P, lvl)
    for all p ∈ P do
        S_p = ∅
        n_p = 0
        for all q ∈ P do
            /* non-dominance at specified priority level */
            if p ≺^lvl q then
                S_p = S_p ∪ {q}
            else if q ≺^lvl p then
                n_p = n_p + 1
        if n_p == 0 then
            /* First subfront within P */
            p_rank = p_rank + ①^lvl
            F_1 = F_1 ∪ {p}
    /* Start from the first subfront */
    i = ①^lvl
    while F_i ≠ ∅ do
        Q = ∅
        for all p ∈ F_i do
            for all q ∈ S_p do
                n_q = n_q − 1
                if n_q == 0 then
                    /* q belongs to the i–th subfront */
                    q_rank = q_rank + i + ①^lvl
                    Q = Q ∪ {q}
        /* Index of the next subfront */
        i = i + ①^lvl
        F_i = Q
```

B. PL-NSGA-II

By incorporating both *PL_fast_nondominated_sort* and *PL_crowding_distance_assignment* into NSGA-II, we obtain an enhanced version that is also capable of handling PL-MPL-MOPs in a proper way: we named this algorithm *PL-NSGA-II*. Now let us look how the two new functions affect the *make_new_pop* procedure, that is drafting the parent population for the next generation. Similarly to what happens in NSGA-II, the subfronts are sequentially included in the next

Algorithm 2 Priority Levels crowding distance assignment.

```
/* F is a leaf-front in the hierarchy of population fronts partitioning */
procedure PL_CROWDING_DIST_ASSIGNMENT(F)
    n = |F|
    for all i ∈ F do
        F[i]_dist = 0
    /* For each level of priority q; p is the index of the last level */
    for q = 1 ... p do
        for j = 1 ... m_q do
            F = sort(F, f_j^(q))
            /* +Inf means "full-scale" (IEEE 754 standard) */
            F[1]_dist += +Inf ①^(1−q)
            F[n]_dist += +Inf ①^(1−q)
            for i = 2 ... (n − 1) do
                /* PL_crowd_dist_ass. has infinitesimal parts */
                F[i]_dist += ①^(1−q) (f_j^(q)(F[i+1]) − f_j^(q)(F[i−1])) / (f_j^(q)max − f_j^(q)min)
```

population in order of rank; when there remains no enough room to insert the next subfront in its entirety, the latter is sorted by PL crowding distance and the fittest individuals eventually survive. PL-NSGA-II could in theory support custom recombination and mutation operators too, but we chose to defer their development to a future, dedicated research work. The pseudocode of PL-NSGA-II is specified in Algorithm 3.

Algorithm 3 PL-NSGA-II algorithm.

```
/* P_0 is the starting population, T is the total number of iterations */
/* The function returns the approximated Pareto front P_T */
procedure PL_NSGA_II (P_0, T)
    for t = 0 ... T − 1 do
        Q_t = make_new_pop(P_t)
        R_t = P_t ∪ Q_t
        /* F is the set of all the non-dominated subfronts */
        F = PL_fast_nondom_sort(R_t, 0)
        P_{t+1} = ∅
        /* i is a gross-index, a.k.a. gross-scalar */
        i = 1①^0
        while |P_{t+1}| + |F_i| ≤ N do
            /* Crowding distance is computed within subfront F_i */
            PL_crowding_dist_assignment(F_i)
            P_{t+1} = P_{t+1} ∪ F_i
            /* Move to the next subfront */
            i = next_index(F, i)
        Sort(F_i, ≺*_n)
        P_{t+1} = P_{t+1} ∪ F_i[1 : (N − [P_{t+1}])]
    return P_T
```

C. Lexicographic NSGA-II

Before concluding the section, we propose a Grossone-less extension of the original NSGA-II (Lex-NSGA-II) for solving PL-MPL-MOPs. This choice is twofold. First, it provides an additional reference algorithm to compare with our proposed PL-NSGA-II. Second, it lets us emphasize the main advantages in representing the priority information by means of a Grossone-based approach, rather than a standard one. Indeed, the former gives birth to an elegant and parameter-less algorithm whose generalization to many hierarchical classes of objectives is straightforward. The same cannot be said for standard approaches like Lex-NSGA-II, whose generalization to many classes may be tedious.

As before, in order to embed the priority information in Lex-NSGA-II we slightly change the domination and the crowding distance (CD) computations of the original NSGA-II algorithm. The former is modified so that the domination check is restricted to one PL at a time in a hierarchical manner. Notice that it considers CD too when two solution are PL-wise Paretian mutually non-dominated. However, the CD concerning next PLs values only when the previous CDs of the two solutions fall within a threshold τ (a new parameter). An implementation is presented in Algorithm 4 and addressed to as Lex-dominance check. On the other hand, the lexicographic crowding distance (Lex-CD) operator computes the crowding distance PL-wise in a hierarchical manner. For brevity, an algorithmic description is omitted here.

Algorithm 4 Lex-Dominance check used in Lex-NSGA-II.

```
/* This function is recursive */
/* a and b are two individuals to compare at the lvl-th level */
/* The function returns the non-dominated individual */
procedure LEX_DOMINANCE(a, b, lvl)
    /* Base case of recursion */
    if lvl < min_lvl then return (rand() > 0.5)? a : b
    /* Dominance of a over b on the level lvl */
    if dominates(a, b, lvl) then
        return a
    else if dominates(b, a, lvl) then
        return b
        /* CD(a, lvl) = Crowding distance of a on the level lvl */
        /* τ = user defined threshold */
    else if CD(a, lvl) − CD(b, lvl) ≥ τ then
        return a
    else if CD(b, lvl) − CD(a, lvl) ≥ τ then
        return b
    else
        return LEX_DOMINANCE(a, b, lvl − 1)
```

Of course, there may be other ways to implement these functionalities, and the same ideas can also be ported to other EMO algorithms as well . We have chosen this algorithm as it looks very similar to our Grossone-based generalization, which is both short to present and easy to understand.

VI. Test cases for PL-MPL-MOPs

Despite a large number of scalable test problems for multi- and many-objective optimization problems in the EMO literature, there does not seem to exist any test problem involving differing priorities. Since the existing benchmark problems are stated in a way that implicitly assumes all the objectives to be equally important, they can not be an adequate test cases to evaluate the proposed algorithms. Here, we design two custom test problems (called PL-A and PL-B) and we

adapt a real-world 10-objective GAA problem [28], where the introduction of levels of priority is quite natural (with the help of a domain expert). We called this modified problem PL-GAA. Finally, also the PL-Crash problem introduced in Section II-B is considered.

In order to assess the effectiveness of our PL-extension, we compare quantitatively the performances of PL-NSGA-II with those of seven classic EMO algorithms, namely: i) postfiltered NSGA-II (NSGA-II-post), ii) prefiltered NSGA-II (NSGA-II-pre), iii) postfiltered NSGA-III (NSGA-III-post), iv) postfiltered MOEA/D (MOEA/D-post), v) prefiltered MOEA/D (MOEA/D-pre), vi) the favoring graph-based approach proposed in [29] (GRAPH), and vii) the dynamic prioritized goal-based approach proposed in [30] (GOAL); vii) the Grossone-less, parametric generalization proposed in Section V-C, namely Lex-NSGA-II, with a threshold τ set to 0.1 based on empirical observation. Of course, we didn't consider NSGA-III prefiltered since the number of objectives at the first level is too low to justify the use of a many-objective optimizer. The authors are aware that the performances of direction-based algorithm such as NSGA-III and MOEA/D may improve if the priority information is somehow injected in the model. This shall be a key point in the future work where the possibility to implement the PL extension for such algorithms will be investigated. Instead, the main idea of this work is to stress how a novel and more general definition of dominance (PL-dominance) comes naturally when considering prioritized problems from a non-Archimedean perspective, e.g., through Grossone Methodology lenses. We aim also to numerically verify how a naive implementation of PL-dominance well performs against nitty-gritty implementations of widely recognized competitive algorithms.

We adopt SBX crossover and polynomial mutation operators in all problems. Also, we run each algorithm 50 times per problem, performing a non-parametric statistical study by means of the KEEL tool [31]. The study consists of four steps: i) collecting the performances by means of the metric $\Delta(\cdot) = \max\{GD(\cdot), IGD(\cdot)\}$, where GD indicates the *generational distance* [32], and IGD the *inverted generational distance* [33, 34]; ii) ranking the algorithms performances exploiting the Friedman test; iii) evaluating the presence of a statistical separation among the outcomes with the Iman-Davenport procedure [35]; iv) if such a separation exists, checking the statistical difference between the best algorithm (according to the Friedman ranking) and all the others, by means of the post-hoc comparison known as Holm test [36]. The population size and the number of epochs per run are specified in the description of each benchmark. Notice that the metric $\Delta(\cdot)$, originally proposed in [37], has already been applied in combination with *gross*-scalars to assess algorithm's performances in PC-MPL-MOPs [12]. As any other metric for EMO algorithms evaluation [38], $\Delta(\cdot)$ is not free from inadequacies, but covers convergence and diversity of obtained solutions. In future works concerning PL-MPL-MOPs, we plan to use some of the additional metrics reviewed in [38] for the evaluation of the performance of the compared algorithms.

For the sake of completeness, we also measure the algorithms execution time on the real-world PL-GAA benchmark. Since PL-A, PL-B, and PL-Crash need a short amount of time to run, we do not compare the various algorithms on them.

A. PL-A problem

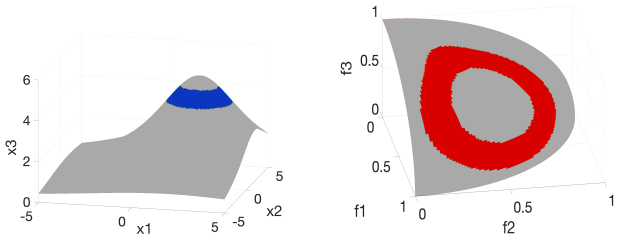
The first PL-MPL-MOP benchmark we propose has five objectives, two of which are less important than the others.

$$\min \left[\begin{pmatrix} f_1^{(1)}(\mathbf{x}) \\ f_2^{(1)}(\mathbf{x}) \\ f_3^{(1)}(\mathbf{x}) \end{pmatrix} + ①^{\cdot 1} \begin{pmatrix} f_1^{(2)}(\mathbf{x}) \\ f_2^{(2)}(\mathbf{x}) \end{pmatrix} \right],$$

where

$$f_1^{(1)}(\mathbf{x}) = \cos\left(\frac{\pi}{10}x_1\right)\cos\left(\frac{\pi}{10}x_2\right),$$
$$f_2^{(1)}(\mathbf{x}) = \cos\left(\frac{\pi}{10}x_1\right)\sin\left(\frac{\pi}{10}x_2\right),$$
$$f_3^{(1)}(\mathbf{x}) = \sin\left(\frac{\pi}{10}x_1\right) + g(\mathbf{x}),$$
$$f_1^{(2)}(\mathbf{x}) = ((x_1-2)^2 + (x_2-2)^2 - 1)^2,$$
$$f_2^{(2)}(\mathbf{x}) = ((x_1-2)^2 + (x_2-2)^2 - 3)^2,$$
$$g(\mathbf{x}) = \left(x_3 - \frac{45}{9 + (x_1-2)^2 + (x_2-2)^2}\right)^2,$$
$$x_1, x_2, x_3 \in [0, 5].$$

The easiest way to grasp this problem is to decompose it into smaller parts. Considering only the first three objectives, namely those whose priority is maximum, it can be recognized that the Pareto surface (on which $g(\mathbf{x}) = 0$) is an octant of a unit sphere. In the variable space, they come from $x_1 \in [0, 5]$ and $x_2 \in [0, 5]$ and $x_3 = 45/\left(9 + (x_1-2)^2 + (x_2-2)^2\right)$. This relationship produces a bell-shaped function of x_3 as a function of x_1 and x_2, as shown in Figure 7a. Among all Pareto-optimal solutions for the primary objectives, secondary objectives $f_1^{(2)}$ and $f_2^{(2)}$, make non-dominated only the points within the blue annular region. Moving to the objective space, the entire positive octant represents the efficient set in the primary objective space, while only the red region the one of the whole problem, i.e., considering also the secondaries (Figure 7b).



(a) Bell-shaped surface on $\mathbf{x}$-space, satisfying $g(\mathbf{x}) = 0$.

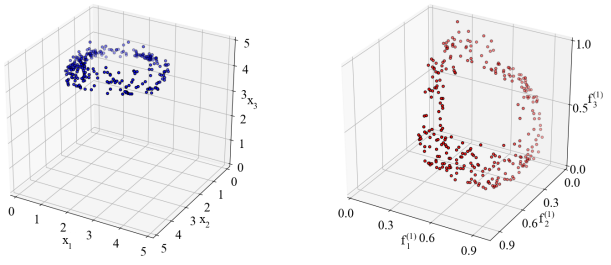
(b) Resulting efficient set in objective space.

Fig. 7: Theoretical Pareto set (blue) and theoretical efficient set (red), for PL-A.

B. Results on PL-A

A typical result using PL-NSGA-II is shown in Figure 8. On the face of it, the quality of the solutions found by PL-NSGA-II is evident. Indeed, the set of optimal points both in the $\mathbf{x}$-space (Figure 8a) and in the objective space (Figure 8b) seems to closely fit the theoretical expectations of Figure 7.

About the technical aspects, we use a population of 100 individuals and run for 500 generations. The mean and the standard deviation of their performances are reported in Table I.



(a) Pareto set in **x**-space. (b) Efficient set in obj. space.

Fig. 8: Obtained solutions by PL-NSGA-II for PL-A.

PL-NSGA-II appears to be the best algorithm, outperforming the second one both in terms of mean and standard deviation. In other words, not only PL-NSGA-II achieves better results, but it is also more stable.

TABLE I: PL-A: Mean and Std. Dev. for metric $\Delta(\cdot)$.

Algorithm	Mean	Std
PL-NSGA-II	$\mathbf{0.002 + 0.43①^{-1}}$	$\mathbf{0.002 - 0.09①^{-1}}$
MOEA/D-post	$0.01 + 2.18①^{-1}$	$0.01 + 0.33①^{-1}$
MOEA/D-pre	$0.01 + 2.94①^{-1}$	$0.01 + 0.90①^{-1}$
NSGA-II-pre	$0.01 + 3.53①^{-1}$	$0.01 + 0.53①^{-1}$
Lex-NSGA-II	$0.06 + 3.99e4①^{-1}$	$0.04 + 74.29①^{-1}$
NSGA-II-post	$0.15 + 2.42e4①^{-1}$	$0.03 - 6.94e3①^{-1}$
NSGA-III-post	$0.49 + 1.47e3①^{-1}$	$0.23 + 8.11e3①^{-1}$
GOAL	$6.63 + 6.62e4①^{-1}$	$7.16 - 3.42e4①^{-1}$
GRAPH	$8.92 + 4.10e4①^{-1}$	$6.8 - 7.12e3①^{-1}$

The higher quality of PL-NSGA-II is confirmed in Table II, where the algorithms are ranked accordingly to the Friedman test. The statistic is 380.155 (distributed according to chi-square with 8 degrees of freedom), and the p-value is zero. This implies the presence of a statistical separation among some algorithms in the pool, as confirmed by the Iman and Davenport test, whose statistic (distributed according to F-distribution with 8 and 392 degrees of freedom) is 938.638 and the associated p-value is zero.

TABLE II: PL-A, Ranking produced by the Friedman test.

Algorithm	Ranking
PL-NSGA-II	**1.06 (best)**
MOEA/D-post	2.58
MOEA/D-pre	3.08
NSGA-II-pre	3.4
Lex-NSGA-II	4.94
NSGA-II-post	5.96
NSGA-III-post	6.98
GRAPH	8.34
GOAL	8.06 (worst)

In order to determine whether there exists a statistical separation between the performances of PL-NSGA-II and the other algorithms, we use Holm's procedure. The results come out from the test are reported in Table III, where i indicates the Friedman rank and z is the normal distribution statistic for the p-value computation. The latter is shown in the last column. Whenever the p-value is less or equal to $\alpha = 0.05$, the null-hypothesis must be rejected, i.e., the corresponding algorithm must be considered statistically different from PL-NSGA-II. For completeness, we mention that the statistics z for the i-th best algorithm is computed as the difference between its average rank (R_i) and the average rank of the best algorithm on the current benchmark (R_0), both according to the Friedman test. Then, such difference is adjusted dividing it by a value SE, in order to compensate for multiple comparison and to control the family-wise error rate. The factor SE is a function of both the number of algorithms (k) and the number of runs (n). Its definition follows:

$$SE = \sqrt{\frac{k\,(k+1)}{6n}}.$$

These results suggest that the Grossone-based PL-extension proposed here may be an effective way to handle hierarchical preference information. Such trend is also confirmed for the next test case.

C. PL-B problem

The second PL benchmark problem has nine objectives, qualifying it as a *many-objective* problem. The objectives are ranked by priority in groups of three, i.e., there are 3 primary objectives, 3 secondaries and 3 tertiaries. We refer to this problem as PL-B.

$$\min \left[\begin{pmatrix} f_1^{(1)}(\mathbf{x}) \\ f_2^{(1)}(\mathbf{x}) \\ f_3^{(1)}(\mathbf{x}) \end{pmatrix} + ①^{-1} \begin{pmatrix} f_1^{(2)}(\mathbf{x}) \\ f_2^{(2)}(\mathbf{x}) \\ f_3^{(2)}(\mathbf{x}) \end{pmatrix} + ①^{-2} \begin{pmatrix} f_1^{(3)}(\mathbf{x}) \\ f_2^{(3)}(\mathbf{x}) \\ f_3^{(3)}(\mathbf{x}) \end{pmatrix} \right],$$

where

$$f_1^{(1)}(\mathbf{x}) = (1 - x_1 x_2)(1 + g(\mathbf{x})),$$

$$f_2^{(1)}(\mathbf{x}) = (1 - x_1\,(1 - x_2))\,(1 + g(\mathbf{x})),$$

$$f_3^{(1)}(\mathbf{x}) = x_1(1 + g(\mathbf{x})),$$

$$f_1^{(2)}(\mathbf{x}) = \cos\left(\frac{\pi}{2}x_1\right)\cos\left(\frac{\pi}{2}x_2\right) + \beta\left(x_1^{\alpha x_3} - x_2\right)^2,$$

$$f_2^{(2)}(\mathbf{x}) = \cos\left(\frac{\pi}{2}x_1\right)\sin\left(\frac{\pi}{2}x_2\right) + \beta\left(x_1^{\alpha x_3} - x_2\right)^2,$$

$$f_3^{(2)}(\mathbf{x}) = \sin\left(\frac{\pi}{2}x_1\right) + \beta\left(x_1^{\alpha x_3} - x_2\right)^2,$$

$$f_1^{(3)}(\mathbf{x}) = \cos\left(\frac{\pi}{2}\left(x_{1,3} - 1\right)\right),$$

$$f_2^{(3)}(\mathbf{x}) = \cos\left(2\pi x_{1,3}\right),$$

$$f_3^{(3)}(\mathbf{x}) = \frac{1}{1 + 0.3(x_{1,3} - 1)^2} - \frac{2}{1 + 1750\,(x_{1,3} - 1)^2},$$

$$g(\mathbf{x}) = \beta(x_3 - x_1)^2,$$

$$x_{1,3} = x_1 + x_3,$$

$$x_1, x_2, x_3 \in [0, 1],$$

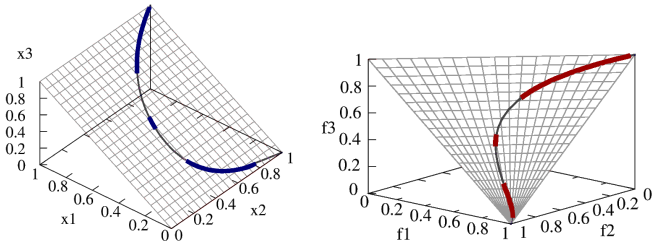
$$\alpha = 2.5, \quad \beta = 10.$$

The primary objectives follow DTLZ1 formulation concept and constitutes the entire triangular planar surface (Figure 9b), which strictly depends on two variables x_1 and x_2 only. The

TABLE III: PL-A, Holm test ($\alpha = 0.05$ and $SE = 0.548$) used to assess whether PL-NSGA-II is statistically better or not.

i	Algorithm	$z = (R_0 - R_i)/SE$	P-value	PL-NSGA-II stat. better?
1	MOEA/D-post	2.775128	0.005518	Yes
2	MOEA/D-pre	3.687999	0.000226	Yes
3	NSGA-I pre	4.272236	0.000019	Yes
4	Lex-NSGA-II	7.083878	0	Yes
5	NSGA-II-post	8.946135	0	Yes
6	NSGA-III-post	10.808392	0	Yes
7	GRAPH	13.291401	0	Yes
8	GOAL	13.875638	0	Yes

respective $\mathbf{x}$-vectors lie on the plane shown in Figure 9a. The function $g(\mathbf{x})$ forces variable x_3 to match x_1 (parameter β adjusts this closeness). The Pareto-optimal solutions of the primary objectives are $x_3 = x_1$, and $x_1,\ x_2 \in [0, 1]$. The three blue parts of the gray line in Figure 9a show this set. The secondary objectives, if considered separately from the others, have their optimal solutions on a curve lying on a unit-spherical surface: $x_2 = x_1^{\alpha x_3}$ and $x_1 \in [0, 1]$. However, in a PL-MPL-MOP the optimal solutions of the secondary objectives must come from the optimal set of the primaries, so the actual efficient set of the 6-objectives problem (3 primaries, 3 secondaries) is a different yet similar curve (blue line in Figure 9b), part of the triangular surface (primaries' efficient set) rather than the spherical one, given as: $x_3 = x_1$, $x_2 = x_1^{\alpha x_1}$, and $x_1 \in [0, 1]$. The tertiary objectives are defined through a meta-variable $x_{1,3} = x_1 + x_3$, which generates Pareto-optimal solutions in three disconnected regions of its domain. They are plotted in blue and red color in Figures 9a and 9b, respectively.

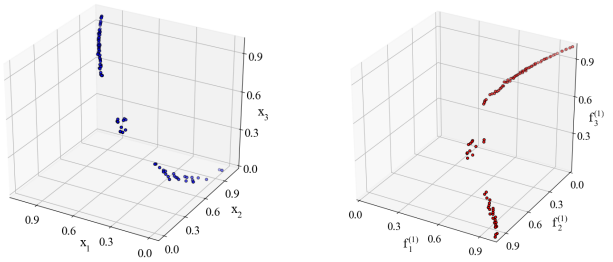


(a) Pareto set in $\mathbf{x}$-space. (b) Efficient set in objective space.

Fig. 9: Theoretical Pareto-optimal solutions (marked in blue and red color) for PL-B.

D. Results on PL-B

Figure 10 shows a typical outcome of PL-NSGA-II algorithm. Since a qualitative perspective, the result is notably close to the real optimum reported in Figure 9 both in the $\mathbf{x}$-space and in the objective space.



(a) Pareto set in $\mathbf{x}$-space. (b) Efficient set in obj. space.

Fig. 10: Solutions found by PL-NSGA-II for PL-B.

Similar to PL-A, we perform runs with 100 individuals for 500 generations. In Table IV, we show the performance of each algorithm, while Table V ranks them according to the Friedman test. The statistic is 298.075 with a p-value of zero, suggesting the presence of a statistical separation, which is confirmed by the Iman-Davenport test (statistic: 143.298, p-value: zero). Once again, the best performing algorithm is PL-NSGA-II. This time, however, the separation between the first and the second most valuable algorithms is thinner, suggesting that the current benchmark is still a simple one for evaluating an algorithm comprehensively. A the same time, it paves the way for further studies to develop challenging test suite for lexicographic multi-objective optimization studies.

TABLE IV: PL-B, Mean and Std. Dev. for metric $\Delta(\cdot)$

Algorithm	Mean	Std
PL-NSGA-II	$\mathbf{0.22 + 0.02①^{-1} + 0.12①^{-2}}$	$\mathbf{0.83 + 0.01①^{-1} + 0.08①^{-2}}$
MOEA/D-post	$0.24 + 0.22①^{-1} + 0.46①^{-2}$	$0.84 - 0.04①^{-1} + 0.06①^{-2}$
Lex-NSGA-II	$0.31 + 7.05①^{-1} + 1.06①^{-2}$	$0.86 - 1.29①^{-1} + 22.17①^{-2}$
NSGA-II-pre	$0.35 + 23.94①^{-1} + 0.90①^{-2}$	$0.81 - 5.13①^{-1} + 7.65①^{-2}$
MOEA/D-pre	$0.36 + 15.29①^{-1} + 0.77①^{-2}$	$0.83 - 3.51①^{-1} + 4.62①^{-2}$
GRAPH	$0.39 + 1.67①^{-1} + 1.09①^{-2}$	$0.9 - 0.25①^{-1} + 2.88①^{-2}$
NSGA-III-post	$0.59 + 2.37①^{-1} + 0.63①^{-2}$	$1.32 + 1.75①^{-1} - 5.24①^{-2}$
NSGA-II-post	$1.17 + 71.66①^{-1} + 1.36①^{-2}$	$4.17 - 2.75①^{-1} + 59.40①^{-2}$
GOAL	$87.87 + 82.66①^{-1} + 1.77①^{-2}$	$73.25 + 19.10①^{-1} + 27.30①^{-2}$

TABLE V: PL-B, Ranking produced by the Friedman test.

Algorithm	Ranking
PL-NSGA-II	**1.16 (best)**
MOEA/D-post	2.3
Lex-NSGA-II	4.2
NSGA-III-post	4.4
NSGA-II-pre	5.22
MOEA/D-pre	5.48
GRAPH	6.04
NSGA-II-post	7.36
GOAL	8.84 (worst)

Finally, the results of the Holm's procedure on PL-B are listed in Table VI. It highlights the statistical separation between PL-NSGA-II and the other approaches.

TABLE VI: PL-B, Holm's test ($\alpha = 0.05$ and $SE = 0.548$), used to assess whether PL-NSGA-II is statistically better or not.

i	Algorithm	$z = (R_0 - R_i)/SE$	P-value	Is PS-NSGA-II stat. better?
1	MOEA/D-post	2.081346	0.037402	Yes
2	Lex-NSGA-II	5.550255	0	Yes
3	NSGA-III-post	5.915404	0	Yes
4	NSGA-II-pre	7.412512	0	Yes
5	MOEA/D-pre	7.887205	0	Yes
6	GRAPH	8.90962	0	Yes
7	NSGA-II-post	11.3196	0	Yes
8	GOAL	14.021697	0	Yes

E. The PL-GAA problem: A real-world problem

The GAA benchmark [28] is a difficult many-objective problem from a real-world scenario. It challenges the decision maker to tune 27 constrained variables in order to design a family of three slightly different aircrafts for general aviation. The crucial point is that not only the optimization procedure must maximize the aircraft performances over conflicting parameters, but also the resulting airplanes must be similar enough not to force industrial facilities diversification. The objective space has 10 dimensions, 9 of which describe the aircraft performances, while the last measures their diversity (henceforth called PFPF). The 9 performance metrics are: the takeoff noise (NOISE), empty weight (WEMP), direct operating costs (DOC), ride roughness (ROUGH), fuel weight (WFUEL), flight range (RANGE), purchase price (PURCH), maximum lift/drag ratio (LDMAX), maximum cruise speed

(VCMAX). Also, PFPF and the first 6 performance metrics are constrained objectives.

With the help of a domain expert, we reformulate the problem dividing the 10 objectives in 4 PLs, obtaining the new problem PL-GAA:

$$\min \left[\left(\lfloor \text{PFPF} \rfloor \right) + \textcircled{1}^{\cdot 1} \begin{pmatrix} \text{DOC} \\ \text{–LDMAX} \\ \text{–RANGE} \end{pmatrix} + \textcircled{1}^{\cdot 2} \begin{pmatrix} \text{NOISE} \\ \text{PURCH} \\ \text{WEMP} \end{pmatrix} + \textcircled{1}^{\cdot 3} \begin{pmatrix} \text{ROUGH} \\ \text{WFUEL} \\ \text{–VCMAX} \end{pmatrix} \right].$$

The PL with the highest importance consists of only one objective: PFPF. Since the aircraft similarity is the decision-maker's main concern, this choice comes as a natural option. As stated in [28], the decision-maker can tolerate a diversity level not greater than 0.05. Thus, we discretize the PFPF output space in intervals of length 0.05, by means of the floor operator: $\lfloor \cdot \rfloor$. In this way, all the aircraft configurations within the tolerance constraint (PFPF $\in [0, 0.05]$) are considered maximally optimized for the first PL. This will produce a number of feasible designs which will be considered equally good for PFPF objective. Subsequent levels will then determine a single or more preferred solutions. With regard to the second PL, the choice fell on the DOC objective because its importance is already pointed out in [28], while the other two (LDMAX and RANGE) are two critical factors in aircraft performance and usability, respectively. A detailed discussion about the remaining two PLs is omitted for brevity; nevertheless, it should be noted that the third PL still contains some highly impacting aircraft features, like NOISE (which can limit the possibility to land close to an urban center) or PURCH (of course, the price plays a relevant role in the decision process). Conversely, the least important level contains not so crucial properties (e.g., the maximum speed VCMAX), or aspects for which relatively cheap and simple customer solutions already exist (e.g., the roughness ROUGH).

F. Results on PL-GAA

Due to the complexity of the problem, each run with 200 individuals is made for 2,000 generations. Moreover, due to the presence of constraints also on the objectives, we exclude the pre-filtered approaches from the performance comparison, since they perform poorly. In Table VII, we report the algorithms' average performances. As before, PL-NSGA-II finds noticeably better solutions than the other algorithms, even for this challenging problem. It is worth noting that the finite component of the $\Delta(\cdot)$ metric has a special meaning: it is zero only when PFPF falls within $[0, 0.05]$, which indicates that all the algorithms with a non-infinitesimal mean were not always able to fully satisfy the PFPF constraint. Also, the higher the mean value, the higher the average decision maker dissatisfaction; the higher the variance, the higher the randomness of the performance quality.

The numerical evidence tells us that only three algorithms are able to respect such a constraint, namely PL-NSGA-II (always), GRAPH (most of the cases), and Lex-NSGA-II (most of the cases). This fact is quite interesting, indeed, because while PL-NSGA-II is designed to properly master the Pareto dominance also in the presence of priority levels (PL-dominance), GRAPH does not use the Pareto dominance at all. Instead, it leverages on favoring the binary relation (see [29]), suggesting to address to this aspect the reasons of its high performances and capability to satisfy the PFPF's constraint. As for Lex-NSGA-II, its incapability to always satisfy the feasibility request on the PFPF parameter should be considered as the effect of a non-optimal tuning of the threshold τ (along with the use of algorithm-oriented genetic operators). Still, it achieves very good performances, reflecting the fact that it is meant to operate with priority levels since the early design. All in all, the only algorithm with an infinitesimal mean, i.e., able to always guarantee the PFPF constraint satisfaction is PL-NSGA-II, proving its strength.

Moving to the non-parametric statistical analysis, the Friedman test (6 degrees of freedom) outputs a statistic of 132.051 with a p-value equal to zero, suggesting the presence of statistical separation. This is confirmed by the Iman and Davenport test, whose statistic (6 and 294 degrees of freedom) is 38.527 and the corresponding p-value is zero. The aforementioned tests are designed to deal with finite quantities only, so we expect them to be unable to markedly distinguish the three best performing algorithms (PL-NSGA-II, GRAPH, and Lex-NSGA-II). Indeed, in the majority of the runs, all the three have the finite component of the mean very close or equal to zero. This could also be the reason behind the low values obtained by the two statistical procedures; the hypothesis is supported by the Friedman test ranking, reported in Table VIII.

We provide the results of the Holm test in Table IX. According to it, the statistical separation between the first three algorithms and the others is very clear, which is reasonable since their performances (see Table VII) are superior on the most important priority level. To determine whether a separation exists between PL-NSGA-II and GRAPH (that would also imply a separation between PL-NSGA-II and Lex-NSGA-II), we performed an ad-hoc Student's t-test, which can also handle infinitesimal quantities: the resulting value of t is $4.76 + 49.38\textcircled{1}^{\cdot 1} - 49.08\textcircled{1}^{\cdot 2}$, which denotes a strong statistical separation between the two algorithms.

G. Time complexity and empirical findings

An analytical assessment of the PL-NSGA-II time complexity is rather difficult and still under study. It is expected to be reasonably slower than NSGA-II-post, but not significantly slower than Lex-NSGA-II. Nevertheless, monitoring their execution time may give a first rough estimation of the algorithms performances from a temporal point of view. We compared them on PL-GAA because it represents the most meaningful benchmark being high-dimensional, complex, and based on a real-world problem. The measurements reported in Table X are the average time needed by each algorithm to execute once. Since each run is mainly characterized by a loop repeated many times, the values are intrinsically very stable: thus, in the table we omitted the standard deviation, which adds little to no valuable information in this context.

The simulations are performed on a machine featuring one 64-bit Intel Core i7-9700 CPU 3GHz and 32GiB RAM DIMM DDR4 Synchronous 2666 MHz. Table X shows that PL-NSGA-II higher performances come at the cost of a relatively

TABLE VII: PL-GAA: Mean and Std. Dev. for metric $\Delta(\cdot)$.

Algorithm	Mean	Std. Dev.
PL-NSGA-II	$\mathbf{4.29e3①^{-1} + 1.91e4①^{-2} + 21.82①^{-3}}$	$\mathbf{1.65e4①^{-1} + 3.23e4①^{-2} + 4.04e4①^{-3}}$
GRAPH	$0.06 + 1.00e4①^{-1} + 7.74e4①^{-2} + 19.7①^{-3}$	$0.24 - 1.30e3①^{-1} + 5.63e8①^{-2} + 3.08e12①^{-3}$
Lex-NSGA-II	$0.16 + 1.21e5①^{-1} + 5.51e5①^{-2} + 311.36①^{-3}$	$0.32 - 1.73e3①^{-1} + 5.11e9①^{-2} + 2.72e13①^{-3}$
NSGA-II-post	$1.24 + 1.12e5①^{-1} + 6.66e5①^{-2} + 482.83①^{-3}$	$1.36 - 1.12e4①^{-1} + 2.55e9①^{-2} + 2.08e13①^{-3}$
NSGA-III-post	$3.9 + 3.24e4①^{-1} + 4.13e5①^{-2} + 214.36①^{-3}$	$4.99 + 1.13e4①^{-1} + 8.48e7①^{-2} - 1.91e11①^{-3}$
GOAL	$102.55 + 8.90e4①^{-1} + 5.48e5①^{-2} + 421.65①^{-3}$	$1.22e2 + 1.23e4①^{-1} + 1.92e7①^{-2} - 1.99e9①^{-3}$
MOEA/D-post	$113.08 + 8.23e4①^{-1} + 8.66e5①^{-2} + 1.47e3①^{-3}$	$26.11 - 5.64e3①^{-1} + 9.56e7①^{-2} + 2.01e10①^{-3}$

TABLE VIII: PL-GAA, Ranking produced by the Friedman test.

Algorithm	Ranking
PL-NSGA-II	**2.57 (best)**
GRAPH	2.69
Lex-NSGA-II	2.83
NSGA-II-post	4.28
NSGA-III-post	4.37
GOAL	4.60
MOEA/D-post	6.56 (worst)

TABLE IX: PL-GAA, Holm test ($\alpha = 0.05$ and $SE = 0.432$) used to assess whether PL-NSGA-II is statistically better or not.

Algorithm	$z = (R_0 - R_i)/SE$	p-value	PL-NSGA-II stat. better?
GRAPH	0.278	0.781	Equiv.
Lex-NSGA-II	0.602	0.547	Equiv.
NSGA-II-post	4.166	$2.8(10^{-5})$	Yes
NSGA-III-post	4.189	$3.1(10^{-5})$	Yes
GOAL	4.699	0	Yes
MOEA/D-post	9.235	0	Yes

TABLE X: PL-GAA, Average execution time for single run.

Algorithm	Avg. time (s)
PL-NSGA-II	597.135
GRAPH	176.683
Lex-NSGA-II	122.047
NSGA-II-post	6.448
NSGA-III-post	10 102.342
GOAL	1 519.953
MOEA/D-post	2 270.763

longer computing time, essentially for two reasons. Firstly, our extension deals with more information (i.e., priority) than most of the other algorithms, so processing it demands more resources with respect to those cases where the priority is not embedded in the algorithm at all. Other algorithms that handle priority in a similar fashion, namely Lex-NSGA-II, GRAPH, and GOAL, show similar behaviours. Secondly, the current implementation of PL-NSGA-II is quite naive, with plenty of room for optimizations that would reduce the execution time without affecting the quality. Conversely, for other algorithms, like NSGA-III and GOAL, it may be difficult to develop an equivalent high-performance implementation. All in all, the actual time consumption of PL-NSGA-II compares quite well with those of the fastest competitors, especially in consideration of the superior quality of its solutions, which are more numerous and better distributed along the efficient set.

H. Results on PL-Crash

Finally, let us briefly present the results concerning the problem we referred throughout the whole study: PL-Crash. Table XI contains the mean and standard deviation values of the $\Delta(\cdot)$ metric, Table XII shows the ranking of the algorithms (Friedman: 363.69; Iman and Davenport: 490.73), while Table XIII gathers the Holm test results.

TABLE XI: PL-Crash: Mean and Std. Dev. for metric $\Delta(\cdot)$.

Algorithm	Mean	Std
PL-NSGA-II	$\mathbf{0.002 + 2.99e5①^{-1}}$	$\mathbf{0.02 + 2.00e4①^{-1}}$
NSGA-II-post	$1.00 + 0.52①^{-1}$	$0.22 - 0.001①^{-1}$
NSGA-II-pre	$1.12 + 0.57①^{-1}$	$0.23 - 0.01①^{-1}$
Lex-NSGA-II	$6.53 + 0.81①^{-1}$	$6.28 + 0.12①^{-1}$
MOEA/D-post	$14.29 + 1.80①^{-1}$	$11.76 - 0.37①^{-1}$
MOEA/D-pre	$19.50 + 1.43①^{-1}$	$12.12 - 0.30①^{-1}$
GOAL	$29.96 + 1.73①^{-1}$	$16.97 + 0.48①^{-1}$
NSGA-III-post	$94.51 + 4.51①^{-1}$	$44.14 + 1.55①^{-1}$
GRAPH	$143.14 + 10.76①^{-1}$	$32.06 + 2.23①^{-1}$

TABLE XII: PL-Crash, Ranking produced by the Friedman test.

Algorithm	Ranking
PL-NSGA-II	**1.00 (best)**
NSGA-II-post	2.30
NSGA-II-pre	2.96
Lex-NSGA-II	4.28
MOEA/D-pre	5.80
MOEA/D-post	5.36
GRAPH	6.46
NSGA-III-post	8.04
GOAL	8.80 (worst)

TABLE XIII: PL-Crash, Holm's test ($\alpha = 0.05$ and $SE = 0.548$), used to assess whether PL-NSGA-II is statistically better or not.

i	Algorithm	$z = (R_0 - R_i)/SE$	P-value	Is PL-NSGA-II stat. better?
1	NSGA-II-post	2.373464	0.017622	Yes
2	NSGA-II-pre	3.578454	0.000691	Yes
3	Lex-NSGA-II	5.988433	0	Yes
4	MOEA/D-post	7.960235	0	Yes
5	MOEA/D-pre	8.763561	0	Yes
6	GRAPH	9.968551	0	Yes
7	NSGA-III-post	12.853223	0	Yes
8	GOAL	14.240786	0	Yes

Again, PL-NSGA-II clearly outperforms all the competitors, showing a strong consistency and a sharp statistical separation, thus indicating the stability and effectiveness of the proposed PL-NSGA-II procedure.

VII. Conclusions and Future Studies

In this paper, we have shown how it is possible to adapt a Pareto-dominance based EMO algorithm to address problems where the objectives can be partitioned into different levels of importance supplied by the decision-makers. Specifically, we have proposed a generalization of the Pareto-dominance

in presence of priority levels, namely the PL-dominance. By means of numerical experiments and quantitative considerations, we have highlighted the critical importance of exploiting the priority information when available, which greatly improves the quality of the solutions. PL-NSGA-II, the generalized version of NSGA-II proposed in this study, has managed to outperform the competitors in all the four benchmark problems, including two real-world problems (one of the latter has high-dimensional decision and objective spaces). As a future work, we plan to implement the PL extension for both NSGA-III and MOEA/D, namely PL-NSGA-III and PL-MOEA/D. In addition, we want to explore other mixed Pareto-lexicographic problems, like those featuring a combination of priority chains and priority levels at the same time. To evaluate EMO algorithms, an appropriate scalable test suite, capable of testing various aspects of hierarchical preference level based problems needs to be developed, and a deeper analysis of performance evaluating metrics is needed as well.

ACKNOWLEDGEMENTS

This work has been partially funded by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence). The authors are glad to acknowledge the aerospace engineer Daniele Gottini for his insightful suggestions on how to prioritize the objectives of PL-GAA benchmark.

APPENDIX

GD AND IGD METRICS FOR PROBLEMS OF THIS STUDY

In this appendix, the performances of each algorithm are further detailed with separate presentation of the $\Delta(\cdot)$ metric in GD and IGD, in Tables XIV-XVII. For each benchmark, a table illustrates the mean of the latter two measures, presenting the algorithms in the same order as they appear in the original ranking. Since on each run $\Delta(\cdot)$ assumes the maximum between GD and IGD, the mean values reported in Section VI are never smaller than the ones reported here.

TABLE XIV: PL-A: Mean for metrics GD and IGD.

Algorithm	GD	IGD
PL-NSGA-II	$\mathbf{1.89e\text{-}3 + 0.16①^{-1}}$	$\mathbf{1.53e\text{-}3 + 0.74①^{-1}}$
MOEA/D-post	$4.60e\text{-}3 + 0.79①^{-1}$	$7.79e\text{-}3 + 2.34①^{-1}$
MOEA/D-pre	$9.44e\text{-}5 + 0.06①^{-1}$	$23.76 + 18.63①^{-1}$
NSGA-II-pre	$1.15e\text{-}3 + 0.47①^{-1}$	$0.02 + 2.96①^{-1}$
Lex-NSGA-II	$0.03 + 601.36①^{-1}$	$0.05 + 107.68①^{-1}$
NSGA-II-post	$0.14 + 3.74e4①^{-1}$	$0.12 + 3.81e3①^{-1}$
NSGA-III-post	$0.02 + 1437.62①^{-1}$	$0.49 + 1.47e3①^{-1}$
GRAPH	$6.52 + 6.90e4①^{-1}$	$0.51 + 5.09e4①^{-1}$
GOAL	$8.92 + 4.10e4①^{-1}$	$0.27 + 1.44e3①^{-1}$

TABLE XV: PL-B, Mean for metrics GD and IGD.

Algorithm	GD	IGD
PL-NSGA-II	$\mathbf{5.8e\text{-}3 + 0.02①^{-1} + 0.10①^{-2}}$	$\mathbf{0.22 + 0.03①^{-1} + 0.23①^{-2}}$
MOEA/D-post	$0.02 + 0.24①^{-1} + 0.45①^{-2}$	$0.22 + 0.04①^{-1} + 0.57①^{-2}$
Lex-NSGA-II	$0.09 + 8.14①^{-1} + 310.72①^{-2}$	$0.25 + 0.78①^{-1} + 1.05①^{-2}$
NSGA-II-pre	$0.15 + 25.18①^{-1} + 0.90①^{-2}$	$0.22 + 1.30①^{-1} + 0.80①^{-2}$
MOEA/D-pre	$0.14 + 16.36①^{-1} + 0.77①^{-2}$	$0.23 + 1.72①^{-1} + 0.76①^{-2}$
GRAPH	$0.03 + 1.90①^{-1} + 0.93①^{-2}$	$0.36 + 1.26①^{-1} + 1.25①^{-2}$
NSGA-III-post	$0.35 + 3.51①^{-1} + 151.01①^{-2}$	$0.29 + 0.98①^{-1} + 0.75①^{-2}$
NSGA-II-post	$0.96 + 76.44①^{-1} + 1.24①^{-2}$	$0.34 + 23.56①^{-1} + 1.61①^{-2}$
GOAL	$81.61 + 74.62①^{-1} + 1.80①^{-2}$	$34.80 + 55.07①^{-1} + 2.21①^{-2}$

REFERENCES

[1] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001, vol. 16.

[2] V. Khare, X. Yao, and K. Deb, "Performance scaling of multi-objective evolutionary algorithms," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2003, pp. 376–390.

[3] R. C. Purshouse and P. J. Fleming, "Evolutionary many-objective optimisation: An exploratory analysis," in *The 2003 Congress on Evol. Comput. (CEC'03)*, vol. 3. IEEE, 2003, pp. 2066–2073.

[4] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[5] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, "Combining convergence and diversity in evolutionary multiobjective optimization," *Evolutionary computation*, vol. 10, no. 3, pp. 263–282, 2002.

[6] K. Deb, D. Joshi, and A. Anand, "Real-coded evolutionary algorithms with parent-centric recombination," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, vol. 1. IEEE, 2002, pp. 61–66.

[7] S. F. Adra and P. J. Fleming, "Diversity management in evolutionary many-objective optimization," *IEEE Trans. on Evolutionary Computation*, vol. 15, no. 2, pp. 183–195, 2011.

[8] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints." *IEEE Trans. Evol. Comp.*, vol. 18, no. 4, pp. 577–601, 2014.

[9] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition." *IEEE Trans. Evolutionary Computation*, vol. 19, no. 5, pp. 694–716, 2015.

[10] Y. Yuan, H. Xu, B. Wang, and X. Yao, "A new dominance relation-based evolutionary algorithm for many-objective optimization," *IEEE Trans. on Evolutionary Computation*, vol. 20, no. 1, pp. 16–37, 2016.

[11] A. Gaur, A. Khaled Talukder, K. Deb, S. Tiwari, S. Xu, and D. Jones, "Unconventional optimization for achieving well-informed design solutions for the automobile industry," *Engineering Optimization*, vol. 52, no. 9, pp. 1542–1560, 2020.

[12] L. Lai, L. Fiaschi, and M. Cococcioni, "Solving Mixed Pareto-Lexicographic Multi-Objective Optimization Problems: The Case of Priority Chains," *Swarm and Evol. Comp.*, p. 100687, 2020.

[13] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[14] K. Miettinen, *Nonlinear Multiobjective Optimization*. Springer, 1999.

[15] X. Liao, Q. Li, X. Yang, W. Zhang, and W. Li, "Multiobjective optimization for crash safety design of vehicles using stepwise regression model," *Structural and multidisciplinary optimization*, vol. 35, no. 6, pp. 561–569, 2008.

[16] Y. D. Sergeyev, "Numerical infinities and infinitesimals: Methodology, applications, and repercussions on two Hilbert problems," *EMS Surveys Math. Sci*, vol. 4, no. 2, pp. 219–320, 2017.

[17] M. Cococcioni and L. Fiaschi, "The Big-M method with the numerical infinite M," *Optimization Letters*, 2020, doi:10.1007/s11590-020-01644-6.

[18] M. Cococcioni, M. Pappalardo, and Y. Sergeyev, "Lexicographic Multi-Objective Linear Programming using Grossone Methodology: Theory and Algorithm," *Applied Mathematics and Computation*, vol. 318, pp. 298–311, 2018.

[19] M. Cococcioni, A. Cudazzo, M. Pappalardo, and Y. D. Sergeyev, "Solving the Lexicographic Multi-Objective Mixed-Integer Linear Programming Problem using Branch-and-Bound and Grossone Methodology," *Comm. in Nonlin. Sc. and Num. Sim.*, vol. 84, p. 105177, 2020.

[20] S. De Cosmis and R. De Leone, "The use of Grossone in Mathematical Programming and Operations Research," *Applied Mathematics and Computation*, vol. 218, no. 16, pp. 8029–8038, 2012.

[21] Y. Sergeyev, "Solving ordinary differential equations by working with infinitesimals numerically on the Infinity Computer," *Applied Mathematics and Computation*, vol. 219(22), pp. 10 668–10 681, 2013.

[22] Y. Sergeyev, M. Mukhametzhanov, F. Mazzia, F. Iavernaro, and P. Amodio, "Numerical methods for solving initial value problems on the

TABLE XVI: PL-GAA: Mean for metrics GD and IGD.

Algorithm	GD	IGD
PL-NSGA-II	$\mathbf{4.18e3①^{-1} + 2.03e4①^{-2} + 22.16①^{-3}}$	$\mathbf{4.25e3①^{-1} + 1.90e4①^{-2} + 22.20①^{-3}}$
GRAPH	$0.06 + 9.91e3①^{-1} + 7.83e4①^{-2} + 19.69①^{-3}$	$0.05 + 2.73e4①^{-1} + 6.47e4①^{-2} + 18.05①^{-3}$
Lex-NSGA-II	$0.10 + 1.20e5①^{-1} + 5.56e5①^{-2} + 311.78①^{-3}$	$0.10 + 1.20e5①^{-1} + 5.56e5①^{-2} + 311.78①^{-3}$
NSGA-II-post	$0.10 + 1.19e5①^{-1} + 5.62e5①^{-2} + 310.74①^{-3}$	$0.09 + 2.30e5①^{-1} + 6.79e5①^{-2} + 411.39①^{-3}$
NSGA-III-post	$3.6 + 2.13e4①^{-1} + 4.09e5①^{-2} + 215.31①^{-3}$	$3.6 + 2.10e4①^{-1} + 4.10e5①^{-2} + 226.17①^{-3}$
GOAL	$101.87 + 8.33e4①^{-1} + 5.47e5①^{-2} + 432.70①^{-3}$	$101.90 + 8.39e4①^{-1} + 5.43e5①^{-2} + 432.68①^{-3}$
MOEA/D-post	$112.18 + 8.40e4①^{-1} + 5.38e5①^{-2} + 432.64①^{-3}$	$112.20 + 8.30e4①^{-1} + 5.55e5①^{-2} + 432.84①^{-3}$

TABLE XVII: PL-Crash: Mean for metrics GD and IGD.

Algorithm	GD	IGD
PL-NSGA-II	$\mathbf{0.002 + 2.87e5①^{-1}}$	$\mathbf{1.69e4 + 3.09e6①^{-1}}$
NSGA-II-post	$0.97 + 0.52①^{-1}$	$0.70 + 0.49①^{-1}$
NSGA-II-pre	$1.12 + 0.57①^{-1}$	$0.50 + 0.48①^{-1}$
Lex-NSGA-II	$0.79 + 0.74①^{-1}$	$6.51 + 0.78①^{-1}$
MOEA/D-post	$2.13 + 0.70①^{-1}$	$14.06 + 1.93①^{-1}$
MOEA/D-pre	$1.06 + 0.27①^{-1}$	$19.49 + 1.50①^{-1}$
GRAPH	$0.07 + 0.04①^{-1}$	$29.96 + 1.73①^{-1}$
NSGA-III-post	$15.09 + 8.36①^{-1}$	$94.51 + 4.51①^{-1}$
GOAL	$143.14 + 10.76①^{-1}$	$2.40 + 0.72①^{-1}$

Infinity Computer," *International Journal of Unconventional Computing*, vol. 12(1), pp. 3–23, 2016.

[23] P. Amodio, F. Iavernaro, F. Mazzia, M. Mukhametzhanov, and Y. 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*, vol. 141, pp. 24–39, 2017.

[24] L. Fiaschi and M. Cococcioni, "Numerical Asymptotic Results in Game Theory using Sergeyev's Arithmetic of Infinity," *International Journal on Unconventional Computing*, vol. 14, pp. 1–25, 2018.

[25] ——, "Non-Archimedean Game Theory: A Numerical Approach," *Applied Mathematics and Computation*, 2020, doi:10.1016/j.amc.2020.125356.

[26] D. Rizza, "Supertasks and numeral systems," in *Proc. of the 2nd Intern. Conf. "Numerical Computations: Theory and Algorithms"*, Y. Sergeyev, D. Kvasov, F. Dell'Accio, and M. Mukhametzhanov, Eds. New York: AIP Publishing, 2016, vol. 1776, p. 090005.

[27] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms – Part I: A unified formulation," *IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 28, no. 1, pp. 26–37, 1998.

[28] L. Wang, A. H. Ng, and K. Deb, *Multi-objective evolutionary optimisation for product design and manufacturing*. Springer, 2011.

[29] F. Schmiedle, N. Drechsler, D. Große, and 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.

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

[31] 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," 2017.

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

[33] C. A. C. Coello and M. R. Sierra, "A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm," in *Mexican Int. Conf. on Art. Int.* Springer, 2004, pp. 688–697.

[34] M. R. Sierra and C. A. C. Coello, "A new multi-objective particle swarm optimizer with improved selection and diversity mechanisms," *Technical Report of CINVESTAV-IPN*, 2004.

[35] R. L. Iman and J. M. Davenport, "Approximations of the critical region of the fbietkan statistic," *Communications in Statistics-Theory and Methods*, vol. 9, no. 6, pp. 571–595, 1980.

[36] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian journal of statistics*, pp. 65–70, 1979.

[37] O. Schutze, X. Esquivel, A. Lara, and C. A. C. Coello, "Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization," *IEEE Trans. Evol. Comp.*, vol. 16, no. 4, pp. 504–522, 2012.

[38] M. Li and X. Yao, "Quality evaluation of solution sets in multiobjective optimisation: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–38, 2019.