

Chapter 8

On the use of Grossone Methodology for Handling Priorities in Multi-Objective Evolutionary Optimization

Leonardo Lai, Marco Cococcioni, Lorenzo Fiaschi, Kalyanmoy Deb

Abstract This chapter 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 two relevant subclasses of MPL-MOPs, namely optimization problems having the objective functions organized as *priority chains* or *priority levels*. A priority chain (PC) is a sequence of objectives ordered lexicographically by importance; conversely, a priority level (PL) is a group of objectives having the same importance in terms of optimization, but a lexicographic ordering exists between the PLs. After describing these problems and discussing why the standard algorithms are inadequate, an innovative approach to deal with them is introduced: it leverages the Grossone Methodology, a recent theory that allows handling priorities by means of infinite and infinitesimal numbers. Most interestingly, this technique can be easily embedded in most of the existing evolutionary algorithms, without altering their core logic. Three algorithms for MPL-

Marco Cococcioni

Department of Information Engineering, University of Pisa, Largo Lucio Lazzarino 1, Pisa, Italy,
e-mail: marco.cococcioni@unipi.it

Leonardo Lai

Department of Information Engineering, University of Pisa, Largo Lucio Lazzarino 1, Pisa, Italy,
e-mail: leonardo.lai@live.com

Lorenzo Fiaschi

Department of Information Engineering, University of Pisa, Largo Lucio Lazzarino 1, Pisa, Italy,
e-mail: lorenzo.fiaschi@phd.unipi.it

Kalyanmoy Deb

Department of Electrical and Computer Engineering, Michigan State University, 428 S. Shaw Lane,
East Lansing, MI, USA,
e-mail: kdeb@egr.msu.edu

MOPs are shown: the first two, called PC-NSGA-II and PC-MOEA/D, are the generalization of NSGA-II and MOEA/D, respectively, in the presence of PCs; the third, named PL-NSGA-II, generalizes instead NSGA-II when PLs are present. Several benchmark problems, including some from the real world, are used to evaluate the effectiveness of the proposed approach. The generalized algorithms are compared to other famous evolutionary ones, either priority-based or not, through a statistical analysis of their performances. The experiments show that the generalized algorithms are consistently able to produce more solutions and of higher quality.

8.1 Introduction

Evolutionary algorithms [12] are known as one of the most effective strategies to solve real problems requiring the simultaneous optimization of two or more functions, thanks to their ability to produce a group of diverse trade-off solutions rather than focusing on a single one. However, researchers have identified critical issues that arise when the number of objectives grows, often over three or four [35, 42]. The most troublesome issues when 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 the canonical recombination operators. Many proposals attempt to address and mitigate these problems: decomposition [53], Pareto dominance alternatives [30], custom recombination operators [14], new diversity management mechanisms [1] or even many-objective specific frameworks [13, 31, 52].

While the growth in dimension raises computational challenges, the multi- or many-objective optimization problems originate from practical situations, so it's often the case that not all the objectives are of equal importance to decision makers. In other words, some objectives may be likely organized on the basis of their preference, e.g. as series of objectives ordered by priority or as multiple groups of objectives ranked by importance. One important requirement of such real world scenarios is that no objective shall be completely discarded already at the formulation stage because of its preference level. As an example, a recent problem from an automobile industry featuring six objectives clearly states – the weight objective was most important to minimize, but designers were also interested in solutions having a trade-off in other five objectives [21]. In situations like this, 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 should be taken into account during the optimization phase. This allows to guide the optimization search better, and make it more computationally efficient. Also, by focusing on the trade-off between similar priority objectives, one can obtain more fine-grained insights about the important objectives.

The domain of interest are therefore *Mixed Pareto-Lexicographic Multi-objective Optimization Problems* (abbreviated as MPL-MOPs), a broad class of problems that assume a structured precedence relationship among some objectives. Different priority structures map to different subclasses of MPL-MOPs. In this chapter, the focus is on two of them: PC-MPL-MOPs, where the objectives are organized in chains of priority [27], and PL-MPL-MOPs, where the objectives are grouped in levels of priority [28, 29]. Their models are discussed extensively in Sections 8.5-8.7 and 8.8-8.11, respectively. Sections 8.2 and 8.4 introduce them briefly: the former is a brief recap of EMO, while the second delves further into the nature of MPL-MOPs. Conclusions are drawn in Section 8.12.

8.2 Multi-objective Optimization and Evolutionary Algorithms

Multi-objective optimization problems (MOPs) are typically stated as:

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

where Ω represents an arbitrary input space. Finding the Pareto optimal solutions in MOPs is non-trivial, and many different methods exist, from mathematical-programming ones to EMO algorithms. Being population-based, the latter are able to deal with a large number of solutions at the same time, whereas other approaches require the same procedure to be repeated over and over (often only changing the starting point). Also, evolutionary algorithms proved to be effective to solve hard problems too, for instance those with many decision variables or local optima. Because of this, EMO has acquired popularity: noteworthy examples are *NSGA-II*, *SPEA2* and *MOEA/D* [15, 53, 55].

Recently, the research focus has shifted towards problems with a large number of objectives, known as *many-objective optimization problems* (*MaOPs*) [32]. 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 [19]. 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.

All these challenges have led to the development of sophisticated methods specific for MaOPs, including evolutionary algorithms that are often variations of their MOP counterpart. A remarkable example is *NSGA-III* [13], which enhances the fundamental ideas of *NSGA-II* to deal better with many-objective problems. Although the above issues can be mitigated by the new approaches, they can hardly be eliminated at all.

8.3 Metrics to assess the efficacy of EMO algorithms

Several performance indicators have been proposed by researchers to evaluate the quality of a non-dominated solution set [23, 37]. One of the most popular is the *hypervolume indicator* [56], which is Pareto compliant, i.e., the better the Pareto front approximation, the better the performance indicator value. It has been shown that a Pareto non-compliant indicator may result in misleading performance results [37, 48]. However, the hypervolume indicator becomes often computationally unfeasible as the objective space dimension scales up.

Two other well known metrics are the *generational distance* (GD) and the *inverted generational distance* (IGD), described in [50] and [7] respectively. Both assess the quality of a non-dominated objective vector set $A = \{a_1, \dots, a_n\}$ with respect to a reference Pareto set $Z = \{z_1, \dots, z_m\}$. The analytical definition of GD is:

$$GD(A) = \frac{1}{n} \left(\sum_{i=1}^n d(a_i, Z)^p \right)^{\frac{1}{p}}$$

where d is the Euclidean distance from a_i to its nearest reference point in Z , and $p \in \mathbb{N}$. The IGD metrics is the inverted variation, defined as:

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

where d now represents the Euclidean distance from z_i to its nearest objective vector in A . This work uses the stabler indicator $\Delta(A) = \max\{GD(A), IGD(A)\}$, first proposed in [48]. In analogy with [26], p is set to 1, and this choice is bivariate: i) it makes the meaning of GD and IGD interpretable (the average of Euclidean distances); ii) it has often been used in the literature. As any other metric for EMO algorithms evaluation [33], $\Delta(\cdot)$ is not impeccable, but covers at least convergence and diversity of obtained solutions. In future works concerning PL-MPL-MOPs, we plan to include more metrics among those reviewed in [33].

To make a fair and reliable comparison of the algorithms and their performance, various non-parametric statistical tests have been carried out, specifically: i) the Friedman's test ii) the Iman-Davenport's test iii) the Holm's test. Even though the resulting values are not reported here due to lack of space (they can be found in [27] and [28]), it is sufficient to say that the statistical separation and hypotheses have been thoroughly tested and validated.

8.4 Mixed Pareto-Lexicographic Optimization

MPL-MOPs refers to a broad class of problems sharing these features:

- Multiple objectives are involved (multi-objective)
- Some objectives have precedence over some others (lexicographic)
- Some objectives cannot be compared to each other by priority (Pareto)

The mathematical formulation of a MPL-MOP is:

$$\begin{aligned} \min & f_1(x), f_2(x), \dots, f_m(x), \\ \text{s.t. } & x \in \Omega \\ & \varphi(f_1, f_2, \dots, f_m) \end{aligned}$$

where m is the number of objective functions, Ω is the feasible variables space, also called decision space or search space, and $\wp(\cdot)$ is a generic priority structure over the objectives. In the absence of $\wp(\cdot)$, the optimization is implicitly assumed to be of Paretian type, and solved with the usual EMO algorithms. At the opposite of Pareto optimization, another well-known corner case of MPL-MOPs is represented by lexicographic problems [36, 39], where $\wp(\cdot)$ arranges the objectives according to a total order of importance. Lexicographic problems can be denoted as:

$$\begin{aligned} \text{lex min } & f_1(x), f_2(x), \dots, f_m(x) \\ \text{s.t. } & x \in \Omega \end{aligned} \tag{8.1}$$

The expression indicates the minimization of the objectives $f_1, f_2, \dots, f_m$ ranked lexicographically: f_1 holds absolutely priority over f_2 , which in turn has absolute priority over f_3 , and so on.

The more general case where $\wp(\cdot)$ is an arbitrary function (i.e., a generic MPL-MOP) has not received enough research attention, despite the applicability to potentially many real-world scenarios, like [21]. Some studies proposed ad-hoc approaches to deal with generic priority relations, including linear scalarization [12] or ϵ -constrained methods [12]. Other scalarization strategies can be found in [22, 38]. Despite being more effective than general purpose ones, these algorithms are not purposely designed to cope with MPL-MOPs, so they have significant shortcomings. For instance, in scalarized problems the weights must be chosen carefully, otherwise the solutions may be wrong or very inaccurate [12], sometimes, the numerical stability of the algorithm may get worse too. Moreover, scalarized approaches typically find only one solution at a time, with dramatic consequences on the overall computation time. Even the simplest priority structure in (8.1) hides non-trivial complexities. A common approach, called preemptive method, is to solve the single-objective subproblems sequentially: 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 be in the optimal set of the primary objective. The same procedure is iterated over the objectives, in order of importance. However it suffers from the growing size of the set of constraints, which makes it increasingly harder to locate feasible solutions. Also the constraints may have an arbitrary form, hard to handle.

The word *mixed* in MPL-MOPs reflects the non-homogeneous nature of these problems, suggesting at the same time an hybrid approach to deal with them. However, the relationship between Pareto optimality and priorities strongly depends on the structure of the specific problem, which in turn affects the mathematical description of the problem itself. Algo-

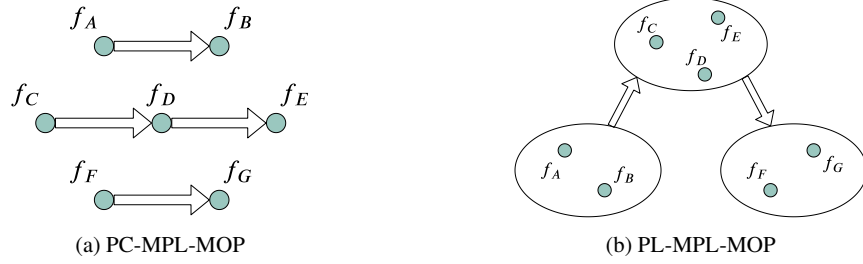


Fig. 8.1: Examples of two different classes of MPL-MOPs: Dots represent objectives, arrows precedences, and ovals groups of objectives.

rithms tailored around this model can potentially converge faster and find better solutions. The next sections introduce two of the significant families of MPL-MOPs. The first one is characterized by chain-like priority structures [27], where the objectives are indeed partitioned and a lexicographic ordering (a chain) exists within each partition. No preference ordering is defined between the chains; therefore, their simultaneous optimization is Paretian. The above problems are called Priority-Chain MPL-MOPs, or PC-MPL-MOPs in short. As opposed to them, Priority-Levels MPL-MOPs (PL-MPL-MOPs in brief) feature a $\varphi(\cdot)$ that arranges the objectives into levels [28, 29]: no priorities exist within each level, and their functions are optimized concurrently in the Paretian way. On the other hand, a lexicographic priority scheme is defined across the levels themselves, that is the Pareto optimization of a level has priority over that of another level. Figure 8.1 illustrates two examples of PC- and PL-MPL-MOPs.

8.5 The Priority Chains Model

The analytical model of a PC-MPL-MOP [27] is:

$$\min \begin{bmatrix} \text{lex min } f_1^{(1)}(x), f_1^{(2)}(x), \dots, f_1^{(p_1)}(x) \\ \text{lex min } f_2^{(1)}(x), f_2^{(2)}(x), \dots, f_2^{(p_2)}(x) \\ \vdots \\ \text{lex min } f_m^{(1)}(x), f_m^{(2)}(x), \dots, f_m^{(p_m)}(x) \end{bmatrix} \quad \text{s.t. } x \in \Omega. \quad (8.2)$$

where $\text{lex min } f_i^{(1)}, f_i^{(2)}, \dots, f_i^{(p_i)}$ indicates the minimization of the objectives $f_i^{(1)}, f_i^{(2)}, \dots, f_i^{(p_i)}$ according to the lexicographic ordering: $f_i^{(1)}$ has priority over $f_i^{(2)}$, which in turn has priority over $f_i^{(3)}$, and so on. Each lex min

block can be seen as a “container” of objectives ranked by importance: for clarity, it is referred to as *macro-objective* and denoted as $\underline{f}_i$. On the other hand, the outer min indicates the minimization (in the Pareto sense) of the macro-objectives. So (8.2) can be rewritten as

$$\min \begin{bmatrix} \underline{f}_1(x) \\ \underline{f}_2(x) \\ \vdots \\ \underline{f}_m(x) \end{bmatrix} \quad \text{s.t. } x \in \Omega$$

where $\underline{f}_1, \underline{f}_2, \dots, \underline{f}_m$ are the macro-objectives.

Moving from standard objectives to macro-objectives can be fairly intuitive. When comparing the same macro-objective, say $\underline{f}_i$, of two different solutions, the one with the lowest value of the primary objective, i.e., $f_i^{(1)}$ is preferred. If the two values happen to be equal, then the preference is determined by the secondary objective, or the tertiary if they match again, and so on. In other words, secondary objectives are useful to compare solutions with equal values on the relatively more important objectives, consistently with the lexicographic ordering. Note the absence of any mutual precedence between the objectives of different macro-objectives, they are unrelated. This property implies that swapping e.g. two secondary objectives of a PC-MPL-MOPs would modify the problem itself. This is not the case for PL-MPL-MOPs.

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

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

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 objective does 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, $1 < 7$, $4 = 4$ and $2 < 9$.

8.5.1 Need of a new approach for PC-MPL-MOPs

Before introducing a novel approach for PC-MPL-MOPs (Section 8.6), it is important to first highlights the limits of the standard Paretian ones. An non-priority-aware optimizer like NSGA-II can be essentially used in two ways for PC-MPL-MOPs: ignoring the priorities or the secondary objectives. Unfortunately, both alternatives have critical drawbacks.

8.5.1.1 Ignoring secondary objectives

Ignoring all the objectives except the most important ones simplifies the problem at the cost of discarding important information, hence 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 large, a common situation in many-objectives tasks, these strategy fails with a notable performance degradation.

8.5.1.2 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, as 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.

8.5.2 Grossone for PC-MPL-MOPs

A very useful application of Grossone Methodology [44] is a ploy to reformulate lexicographic problems in a way that allows performing actual numerical computations by means of macro-objectives, something that is not possible with standard model. Grossone has been successfully applied in several other optimization problems, such as in linear and non-linear programming [8, 9, 11], in global optimization [46], in large-scale unconstrained optimization [10], in machine learning [2, 3, 20] and in many other areas. In addition, the Grossone Methodology has been shown to be independent from non-standard analysis [45]. The use made of Grossone here is rather simple: given the lexicographic problem $\text{lex min } f^{(1)}(x), f^{(2)}(x), \dots, f^{(p)}(x)$, a G-scalar is built using each objective $f^{(j)}$ as a G-digit relative to the G-power $\mathbb{1}^{1-j}$, an idea firstly proposed in [43]. Thus that problem can be rewritten as

$$\min f^{(1)}(x) + \mathbb{1}^{-1} f^{(2)}(x) + \dots + \mathbb{1}^{1-p} f^{(p)}(x)$$

or equivalently $\min \underline{f}(x)$. The higher the priority, the higher the G-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 G-scalars. Since all the four basic operations are well-defined for G-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 [5, 6]. So, PC-MPL-MOPs can be easily reformulated with Grossone as

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

8.6 PC-NSGA-II and PC-MOEA/D

This section aims at generalizing the original NSGA-II [15] and MOEA/D [53] algorithms in order to allow them to solve PC-MPL problems. As said, this result will be achieved by extending them to handle G-scalars, giving birth to *PC-NSGA-II* and *PC-MOEA/D*, respectively. Of course these new versions must be equipped with the redefinition of the four elementary operations too, in order to operate with G-scalars. Different algorithms could be chosen as well, e.g. SPEA2, but NSGA-II and MOEA/D were eventually picked because they are popular, simple and parameter-less, in the sense that they do not introduce other parameters than those typically required by genetic algorithms. 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 8.7.1, 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, the algorithms core procedures have been renamed, e.g., *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.

Consider its pseudocode in Algorithm 1:

- *PC_fast_nondominated_sort*: the nondominance operator $\leq$ (underlined) replaces the old $<$. This is very similar to the traditional definition of dominance, but the comparisons are now between macro-objectives (G-scalars) instead of single objectives (reals). Its formal definition is:

Definition 8.1 Pareto-Lexicographic dominance (PC-dominance).

Given two solutions A and B , A dominates B (written $A \leq B$) if:

$$A \leq B \iff \begin{cases} \underline{f}_i(x_A) \leq \underline{f}_i(x_B) & \forall i = 1 \dots m \\ \exists j : \underline{f}_j(x_A) < \underline{f}_j(x_B) \end{cases} \quad (8.3)$$

The underlining notation has general validity: every time it appears, from now on, it indicates G-scalar quantities or operations between them. This remains consistent with the use made in Section 8.7. Please, notice that the lexicographic contribution is implicitly performed within the comparisons between two G-scalars.

- *PC_crowded_comparison_operator*: it works just like the old $<_n$ operator described in [15], but the last comparison is now performed between G-scalars because of the nature of *PC_crowding_distance*. It is denoted by $\leq_n$.
- *PC_crowding_distance*: it is a G-scalar, since it is computed by means of operations (subtractions, divisions, etc.) between G-scalars. G-operations, by design, preserve the relative importance between the objectives, and such property is reflected in the crowding distance computation too. Being the finite component of the macro-objectives, primary objectives 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. Notice that the crowding distance of the extreme points can be initialized to ①, similarly to NSGA-II where it is set to ∞ . If we represent the priorities with non-positive G-powers only (a non-restrictive assumption) ① is always guaranteed to be greater than any other crowding distance value.

Similarly, MOEA/D can be extended by means of G-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 G-scalar case. The new function works similarly to its old version except for the dominance operator, which becomes the $\leq$ described in (8.3) instead of the standard $<$.

Algorithm 1 PC-NSGA-II algorithm

```

1: procedure PC_NSII
2:    $R_t = P_t \cup Q_t$ 
3:   /* fast_nondominated_sort with gross-scalar objectives */
4:    $F = PC\_fast\_nondominated\_sort(R_t, 0)$ 
5:    $P_{t+1} = \emptyset$ 
6:    $i = 1$ 
7:   while  $|P_{t+1}| + |F_i| \leq N$  do
8:     /* Crowding distance with gross-scalars */
9:      $PC\_crowding\_distance\_assignment(F_i)$ 
10:     $P_{t+1} = P_{t+1} \cup F_i$ 
11:     $i = i + 1$ 
12:    /* Sort by PC_crowded_comparison_operator */
13:     $sort(F_i, \leq_n)$ 
14:     $P_{t+1} = P_{t+1} \cup F_i[1 : (N - [P_{t+1}])]$ 
15:     $Q_{t+1} = make\_new\_pop(P_{t+1})$ 
16:     $t = t + 1$ 

```

```

1: procedure PC_FAST_NONDOMINATED_SORT( $P$ )
2:   for all  $p \in P$  do
3:      $S_p = \emptyset, n_p = 0$ 
4:     for all  $q \in P$  do
5:       if  $p \leq q$  then ▷ PC-dominance being used here
6:          $S_p = S_p \cup \{q\}$ 
7:       else if  $q \leq p$  then ▷ PC-dominance used also here
8:          $n_p = n_p + 1$ 
9:       if  $n_p = 0$  then
10:         $prank = 1$ 
11:         $F_1 = F_1 \cup \{p\}$ 
12:       $i = 1$ 
13:      while  $F_i \neq \emptyset$  do
14:         $Q = \emptyset$ 
15:        for all  $p \in F_i$  do
16:          for all  $q \in S_p$  do
17:             $n_q = n_q - 1$ 
18:            if  $n_q = 0$  then
19:               $qrank = i + 1$ 
20:               $Q = Q \cup \{q\}$ 
21:             $i = i + 1$ 
22:             $F_i = Q$ 

```

```

1: procedure PC_CROWDING_DISTANCE_ASSIGNMENT( $F$ )
2:    $n = |F|$  ▷ number of solution in the current front  $F$ 
3:   for all  $i \in F$  do
4:      $F[i]_{dist} = 0$ 
5:   for  $j = 1 \dots m$  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  $i = 2 \dots (n - 1)$  do
9:       /* the PC_crowding_distance is a gross-scalar */
10:       $F[i]_{dist} = F[i]_{dist} + \frac{f_j(F[i+1]) - f_j(F[i-1])}{f_j^{max} - f_j^{min}}$ 

```

Algorithm 2 PC-MOEA/D algorithm

```

1: procedure PC_MOEA/D
2:    $EP = \emptyset$ 
3:   for  $i = 1 \dots N$  do
4:      $\lambda^{i_1}, \dots, \lambda^{i_T} = find\_closest\_weights(\lambda_i, T)$ 
5:      $B_i = \{i_1, \dots, i_T\}$ 
6:    $x^1, \dots, x^N = initialize\_population(N)$ 
7:   for  $i = 1 \dots N$  do
8:      $FV^i = F(x^i)$ 
9:    $z_1, \dots, z_m = initialize\_ref\_point(m)$ 
10:  while stop_criteria() = False do
11:    for  $i = 1 \dots N$  do
12:       $k, l = rand(B_i, 2)$ 
13:       $y = make\_new\_sol(x^k, x^l)$ 
14:       $y' = mutate(y)$ 
15:      for  $j = 1 \dots m$  do
16:        if  $z_j < \underline{f}_j(y')$  then
17:           $z_j = \underline{f}_j(y')$ 
18:      for all  $j \in B_i$  do
19:        if  $\underline{g}^{te}(y' | \lambda^j, z) \leq \underline{g}^{te}(x^j | \lambda^j, z)$  then
20:           $x^j = y'$ 
21:           $FV^j = F(y')$ 
22:       $DD = PC\_find\_dominated\_sol(EP, F(y'))$ 
23:       $DG = PC\_find\_dominating\_sol(EP, F(y'))$ 
24:       $EP = EP \setminus DD$ 
25:      if  $DG = \emptyset$  then
26:         $EP = EP \cup F(y')$ 

```

8.7 Test cases for PC-MPL MOPs

Many test problems were proposed to assess the performance of multi-objective optimization algorithms [16, 50, 54]. However, very few articles include problems where priorities exist among the objectives, and none specifically for the MPL class. The following sections contain some new PC-MPL benchmarks to verify the effectiveness of the PC-generalizations of Section 8.6. More problems are also discussed in [27]. Note that this study aims only at validating the conceptual superiority of priority-structure-aware algorithms design versus their native counterparts or other priority-based algorithms; instead, it is out of the scope of this chapter to analyze the raw performance of the PC-algorithms, which ultimately depends on the backbone MOP algorithm (NSGA-II or MOEA/D here). A detailed analysis concerning the identification and construction of interesting and challenging PC-MPL benchmarks is left for future studies.

All the experiments use SBX crossover and polynomial mutation for real-encoded genotypes, and two-point crossover and flip-bit mutation with probability $\frac{1}{l}$ (bitstring length l) for binary-encoded ones. The experiments are repeated 50 times for each benchmark, with 500 epochs per run before halting the algorithm. The initial population is made of 100 random individuals. In addition to PC-NSGA-II and PC-MOEA/D, six more algorithms compete on each benchmark: NSGA-II and MOEA/D ignoring the secondary objectives (Section 8.5.1.1), NSGA-II and MOEA/D ignoring the priorities and a posteriori filtering (Section 8.5.1.2), Tan et al. [49], Chang et al. [4].

8.7.1 Test Problem: PC-1

PC-1 is a PC-MPL 01-knapsack problem, featuring 6 objectives (2 chains of 3 objectives):

$$\begin{aligned}
& \max \begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix} \\
& \text{s.t. } Wx \leq C \\
& \underline{f}_j(x) = \underline{V}_j x \\
& \underline{V}_j = V_j^{(1)} + \mathbb{1}^{-1} V_j^{(2)} + \mathbb{1}^{-2} V_j^{(3)} \\
& V_1^{(1)} = [7, 1, 5, 2, 9, 4, 4, 2, 8, 2, 1, 6, 9, 5, 4, 4, 9, 5, 3, 3]^T \\
& V_1^{(2)} = [3, 1, 6, 6, 7, 2, 5, 2, 1, 9, 1, 8, 9, 6, 7, 4, 4, 5, 9, 4]^T \\
& V_1^{(3)} = [2, 1, 3, 5, 8, 9, 5, 7, 1, 6, 4, 7, 9, 5, 1, 5, 5, 4, 4, 2]^T \\
& V_2^{(1)} = [1, 7, 3, 6, 2, 3, 7, 7, 9, 9, 5, 3, 5, 3, 1, 8, 6, 1, 9, 3]^T \\
& V_2^{(2)} = [2, 6, 4, 7, 7, 4, 2, 4, 9, 4, 3, 3, 6, 8, 2, 8, 1, 6, 8, 8]^T \\
& V_2^{(3)} = [8, 9, 5, 9, 8, 9, 4, 9, 7, 5, 4, 5, 3, 4, 4, 7, 3, 8, 2, 4]^T \\
& W = [4, 8, 6, 5, 5, 5, 7, 4, 8, 4, 4, 1, 8, 7, 4, 3, 5, 1, 9, 5]^T \\
& C = 50, \quad x_i \in \{0, 1\} \quad i = 1, \dots, 20
\end{aligned} \tag{8.4}$$

To give an idea of how important the secondaries are in the optimization process, consider what happens when running PC-NSGA-II and NSGA-II (primary-only) on this problem. PC-NSGA-II gives the result shown in Fig. 8.2a, which includes 19 distinct solutions. The True Pareto front (computed through brute-force enumeration), consists of 22 solutions (see [27]). A comparison between the two highlights that, out of the 19 found solutions, 16 are true optima, 2 differ from their closest true optimum for one secondary, 1 for a primary. As for the 6 missing true optima, another solution with the same primary values was found for 2 of them, while for the other 4 the primaries error is ≤ 3 .

With NSGA-II, the secondary objectives are computed only at the end of the optimization process, and unused during the evolution phase. Not only it finds fewer solutions (Fig. 8.2b), but also of lower quality. Out of 11, 8 are true optima, 2 are sub-optimal with respect to primaries, 1 to secondaries. This shortage of solutions from NSGA-II is the consequence of its inability to preserve “apparently horizontal/vertical” solutions, which differ in their infinitesimal components only.

The results are analyzed quantitatively with the $\Delta(\cdot)$ metric, already introduced in Section 8.3. Table 8.1 reports the mean and standard deviation of the performance measurements collected over 50 runs. It turns out that the PC algorithms do better than their standard counterparts. Specifically, PC-MOEA/D turns out to be the best performing algorithm, immediately followed by MOEA/D without secondaries. This confirms that PC algorithms inherit benefits and some limitations from their underlying

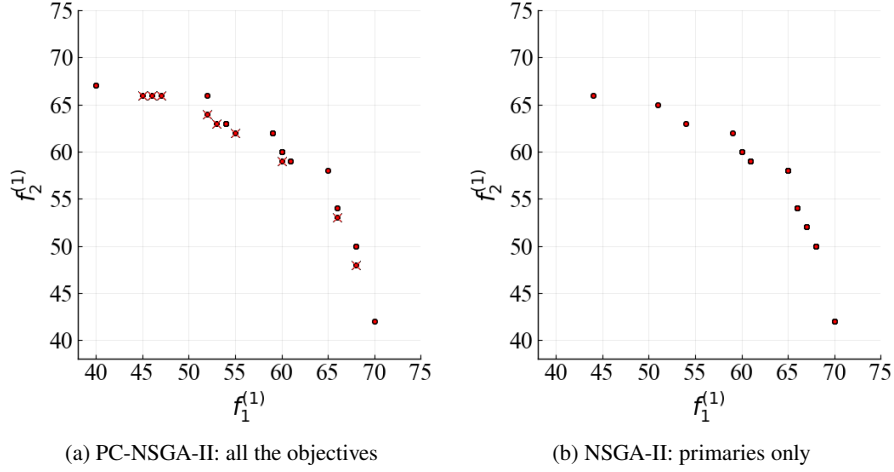


Fig. 8.2: Primary objectives of PC-1, after 300 generations with 400 individuals. PC-NSGA-II solutions that would be instead discarded by NSGA-II (i.e. missing on the right plot) are marked with a cross.

optimizer too. To clarify, based on the empirical evidences, it is reasonable to say that MOEA/D works consistently better than NSGA-II on this specific benchmark (PC-1). Indeed, not only PC-MOEA/D gets a lower score (i.e., mean) than PC-NSGA-II, but also MOEA/D without secondaries beats NSGA-II without secondaries, and the same is true 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 without secondaries usually perform similarly to their PC variation, at least on the primary objectives, since both try to optimize them in a Pareto manner. These considerations justify why PC-MOEA/D is at the top of the ranking and why MOEA/D without secondaries is second.

8.7.2 Test Problem: PC-3

The PC-3 test case is a modified version of POL, a well-known benchmark based on the Poloni's study [41], part of the Van Veldhuizen's suite [50]:

Table 8.1: Mean and Std of metric $\Delta(\cdot)$ on PC-1 after 50 repetitions.

Algorithm	Mean	Std
PC-MOEA/D	$0.36 + 0.25\mathbb{I}^{-1} + 0.99\mathbb{I}^{-2}$	$0.31 - 0.07\mathbb{I}^{-1} + 1.11\mathbb{I}^{-2}$
MOEA/D pre	$0.54 + 0.43\mathbb{I}^{-1} + 3.22\mathbb{I}^{-2}$	$0.19 - 0.22\mathbb{I}^{-1} + 1.37\mathbb{I}^{-2}$
MOEA/D post	$1.44 - 0.22\mathbb{I}^{-1} + 0.88\mathbb{I}^{-2}$	$0.54 + 0.15\mathbb{I}^{-1} + 0.59\mathbb{I}^{-2}$
PC-NSGA-II	$1.6 + 0.17\mathbb{I}^{-1} + 1.2\mathbb{I}^{-2}$	$0.74 + 0.19\mathbb{I}^{-1} + 0.44\mathbb{I}^{-2}$
NSGA-II pre	$1.95 + 0.57\mathbb{I}^{-1} + 2.37\mathbb{I}^{-2}$	$0.91 + 0.21\mathbb{I}^{-1} + 0.53\mathbb{I}^{-2}$
NSGA-II post	$1.79 + 0.08\mathbb{I}^{-1} - 0.77\mathbb{I}^{-2}$	$0.33 - 0.02\mathbb{I}^{-1} + 0.88\mathbb{I}^{-2}$
Tan et al.	$12.83 + 2.71\mathbb{I}^{-1} + 4.03\mathbb{I}^{-2}$	$0.73 + 0.04\mathbb{I}^{-1} + 1.2\mathbb{I}^{-2}$
Chang et al.	$18.01 + 4.45\mathbb{I}^{-1} - 4.42\mathbb{I}^{-2}$	$2.53 + 2.31\mathbb{I}^{-1} + 3.48\mathbb{I}^{-2}$

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

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 slightly shifted towards the line $x_1 = 0$; the magnitude of such effect depends on α .

Figure 8.4 shows the results of PC-NSGA-II for PC-3. As expected, the shape of the Pareto front is similar to the POL one, the most noticeable difference being that it appears as a set of disconnected points due to the discretization (floor function). As it is often the case, there are some solutions in the Pareto frontier of primary objectives with equal values of $f_1^{(1)}$ but different values of $f_2^{(1)}$, which graphically appear as vertically-

Table 8.2: Mean and Std of metric $\Delta(\cdot)$ on PC-3 after 50 repetitions.

Algorithm	Mean	Std
PC-NSGA-II	0.01	$0.02 + 0.01\textcircled{1}^{-1}$
NSGA-II pre	$0.34 + 0.09\textcircled{1}^{-1}$	$0.09 + 0.02\textcircled{1}^{-1}$
PC-MOEA/D	$0.59 - 0.02\textcircled{1}^{-1}$	$0.58 - 0.02\textcircled{1}^{-1}$
NSGA-II post	$0.65 - 0.08\textcircled{1}^{-1}$	$0.16 - 0.07\textcircled{1}^{-1}$
MOEA/D pre	$0.89 + 0.05\textcircled{1}^{-1}$	$1.4 - 0.01\textcircled{1}^{-1}$
MOEA/D post	$1.23 - 0.03\textcircled{1}^{-1}$	1.48
Tan et al.	$1.34 - 0.82\textcircled{1}^{-1}$	$0.28 - 0.16\textcircled{1}^{-1}$
Chang et al.	$38.06 - 7.83\textcircled{1}^{-1}$	$3.0 - 0.09\textcircled{1}^{-1}$

aligned points. While the picture seems to show only a few dozens of solutions, there are in fact 700 fighting for optimality, but many of them are so close to each other that they overlap. Using the standard approach of removing priorities, running NSGA-II and eventually filtering by priority-aware nondominance, the obtained results are those of Figure 8.3. 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 final set of NSGA-II (Figure 8.3), 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 farther away from the real Pareto front (thus worse) than those found by PC-NSGA-II. Essentially, it happens because NSGA-II completely ignores the priorities of the objectives, not taking advantage of that valuable information. On the other hand, PC-NSGA-II is able to exploit it, succeeding to find more accurate solutions.

Table 8.2 reports the mean and the standard deviation of the algorithms performances on the PC-3 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 third and very close to the second.

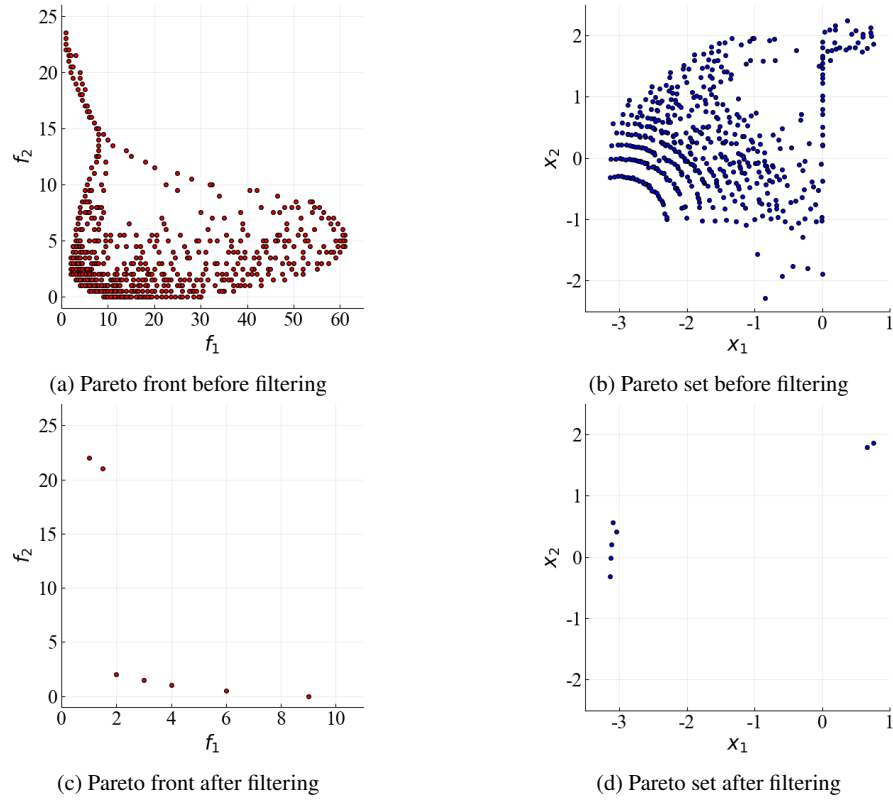


Fig. 8.3: Obtained solutions by NSGA-II for PC-3

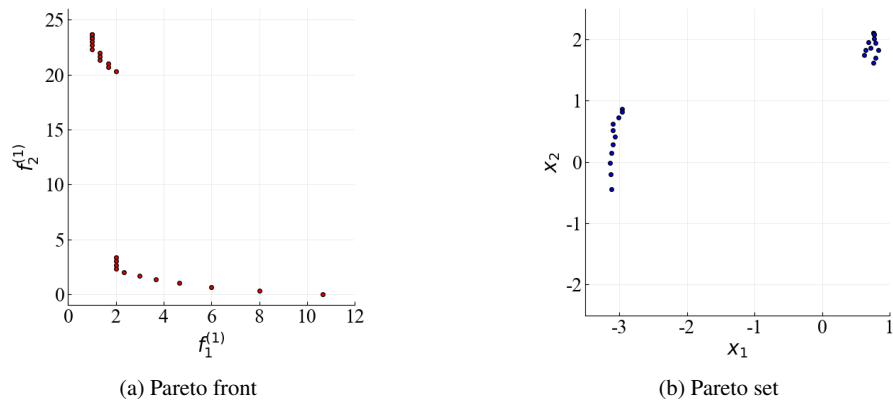


Fig. 8.4: Solutions obtained by PC-NSGA-II for PC-3

8.8 The Priority Levels Model

PL-MPL-MOPs [28, 29] allow one to consider groups of conflicting objectives ordered by priority. For instance, among the seven objectives of Fig 8.1b, 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 is lexicographically more important than the latter. The objectives f_F and f_G are in the same priority level, which has lesser priority than the other ones.

Definition 8.2 (PL-MPL-MOP)

Any MPL-MOP satisfying the two conditions below is a PL-MPL-MOP:

- Some groups of objectives have clear precedence over some others and they can be totally ordered according to it
- Within each group, objectives cannot be compared to each other on the basis of importance

Finally, the mathematical formulation of a PL-MPL-MOP is:

$$\text{lex min} \left[\min \begin{pmatrix} f_1^{(1)}(x) \\ f_2^{(1)}(x) \\ \vdots \\ f_{m_1}^{(1)}(x) \end{pmatrix}, \min \begin{pmatrix} f_1^{(2)}(x) \\ f_2^{(2)}(x) \\ \vdots \\ f_{m_2}^{(2)}(x) \end{pmatrix}, \dots, \min \begin{pmatrix} f_1^{(p)}(x) \\ f_2^{(p)}(x) \\ \vdots \\ f_{m_p}^{(p)}(x) \end{pmatrix} \right], \quad (8.5)$$

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

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 [40]. Of course, the formulation above 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.

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

The automotive crashworthiness problem [34] 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, also their mutual relation may vary across manufacturers. 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 vehicles. A possible approach is to reformulate the problem in a PL-fashion, as illustrated in (8.6). The model splits the original 3-objective optimization task in two PLs: the first (high priority) contains the mass and the acceleration, while the second (lower priority) focuses on safety while still preserving the goal of maximizing the acceleration.

$$\begin{aligned} \text{lex min } & \left[\min \begin{pmatrix} \text{mass}(x) \\ -\text{accel}(x) \end{pmatrix}, \min \begin{pmatrix} \text{toe}(x) \\ -\text{accel}(x) \end{pmatrix} \right] \\ \text{s.t. } & x \in [1, 3]^5. \end{aligned} \quad (8.6)$$

Conversely, other firms may prefer to focus on safe cars. Such problem will be analyzed in particular in the experimental section:

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

8.8.2 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 and which recall a lot those in Section 8.5.1.

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. Only at the end of the process, the objectives from the other levels are exploited to rank the obtained solutions. This scheme, called here *prefiltering*, is motivated by the fact that primary objectives have the highest impact during the optimization. The weakness, however, is the little to no influence of the less important objectives in the result. Unlike the postfiltering method, it puts indeed a lot of responsibility on the primary objectives.

Section 8.10 presents a new algorithm that is able to overcome these difficulties, that does not split the optimization into sub-tasks, uses all the objectives concurrently, yet still privileges those belonging to higher levels. Again, the approach uses the Grossone Methodology.

8.8.3 Enhancing PL-MPL problems by means of Grossone

The idea to implement PL-aware algorithms is the similar to the one in Section 8.5.2 in the way it uses $\textcircled{1}$ for lexicographic ranking. However, here the $\textcircled{1}$ powers represent scaling factors for groups of objectives, i.e., multi-objective optimization tasks rather than single functions. PL-MPL-MOPs can be thus reformulated as:

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

Without going into the mathematical details, the optimization problem can be thought has a minimization problem (and not a lexicographic one) of the PLs, whose contribution is weighted in accordance to their priority. The level with the highest priority is weighted by $\textcircled{1}^0$ ($=1$, omitted), the second one by $\textcircled{1}^{-1}$, and so on until no more PLs remain, thereby giving infinitely more importance to the higher levels. As a practical example, the Grossone-based rewriting of the problem in (8.6), called PL-Crash, is

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

With such reformulation of the problem, only one more ingredient is needed to propose a PL extension of an EMO algorithm for solving PL-MPL-MOPs. Such element is a novel, Grossone-based definition of dominance, that will be presented in the next section. After that, the improvement of some fundamental routines of the a standard EMO algorithm in order to handle PL-MPL-MOPs in an effective way is discussed.

8.9 Handling Precedence in PL-MPL-MOPs

8.9.1 A New Definition of Dominance: PL-dominance

As the original Pareto-optimality definition is not enough to cope with PL-MPL-MOPs, a new one is proposed to take the priority structure into account and guide the search towards better optima. Unlike PC-dominance, creating a good *PL-dominance* is not trivial [28], as it must be well-defined (the transitive property requires special attention) and be effective in practice. The idea behind the PL-dominance proposed here is to extend the concept of *non-dominated fronts*. According to the 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. The new definition introduces the concept of “subfronts”, which generalize “fronts” and consist in nested fronts determined by the priorities. The procedure to partition the population into subfronts, described below, may resemble in some aspects the preference-based ranking scheme by Fonseca and Fleming [17].

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. Grossone Methodology comes in handy,

letting one uniquely identify any subfront within the hierarchy by means of a G-scalar index. For instance, F_i where $i = 2\mathbb{1}^0 + 7\mathbb{1}^{-1} + 3\mathbb{1}^{-2}$ 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). The formal definition is:

Definition 8.3 (PL-Dominance)

Given a PL-MPL-MOP and 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 : $A <^* B \iff i < j$.

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.

The transitivity of the proposed *PL-Dominance* can be proven:

Proposition 8.1 *PL-dominance in Definition 8.3 is transitive.*

Proof Let S be a set of distinct solutions already partitioned in subfronts. Following the notation above, indicate the partitioning in subfronts as follows: $\forall X \in S, X \in F_x$. In order to demonstrate that $<^*$ is transitive, the following must hold: $\forall X, Y, Z \in S, X <^* Y \wedge Y <^* Z \Rightarrow X <^* Z$. One can show that it is always true by repeatedly applying Definition 8.3. Indeed, $X <^* Y \wedge Y <^* Z \Rightarrow x < y \wedge y < z \Rightarrow x < y < z$. $\square$

8.9.2 PL-Dominance in PL-Crash Problem

For a better comprehension of how PL-Dominance works, this subsection illustrates its step-by-step application in the case of PL-Crash problem (8.6). In accordance with Definition 8.3, the optimal Pareto front shall consist of all the solution falling in the subfront indexed by the G-scalar $1+1\mathbb{1}^{-1}$. 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 [13]) as described in the next lines. First, all the solutions are projected on the mass-acceleration plane and only the Pareto efficient survives. Figure 8.5 reports this process: the green dots indicate

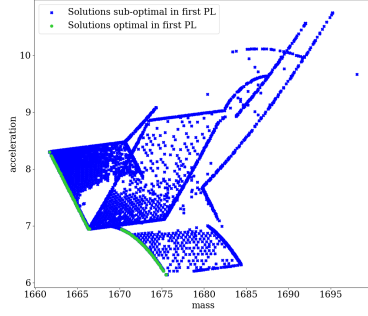


Fig. 8.5: Original optimal solutions on the mass-acceleration plan. First PL in green.

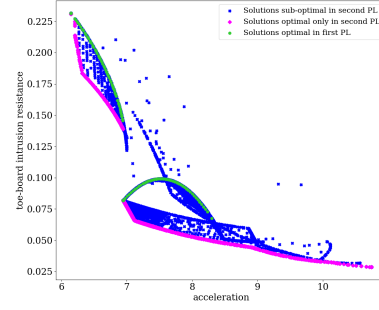


Fig. 8.6: Original optima on the toe-acceleration plan. First PL optima in green, second in magenta.

the individuals which are preserved and therefore form the parent front, the one indexed by the G-scalar 1. Then, the survived solutions are projected on the toe-acceleration plane (second priority level, Figure 8.6), and only those which are non-dominated also there can be considered truly optimal. Such individuals, highlighted in red in Figure 8.7, are the only ones which belong to the subfront $1 + 1\textcircled{1}^{-1}$, since they are the non-dominated solutions (including secondaries) of the non-dominated ones in the first PL.

If one wants to retrieve the sub-optimal solutions belonging to the $1 + 2\textcircled{1}^{-1}$ 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 $2 + \textcircled{1}^{-1}$ subfront, s/he should eliminate all the solutions in the front 1 (regardless the infinitesimals), and again repeat the procedure just illustrated. Figure 8.8 shows the solutions of the PL-MPL-MOP problem (red points) in the entire three-dimensional objective space (marked with blue points). No known domination check on the entire objective space is able to promote the red points only. This further highlights the novelty in the PL-dominance definition as well as its effectiveness in PL-MPL-MOPs. Note that the postponed filtering only works when the whole front of the non-prioritized problem is known, which may not be the case for high-dimensional problems 8.8.2. To overcome these issues, next section introduces a generalization of NSGA-II, enhanced with Grossone Methodology and PL-Dominance in order to transparently and directly cope with any PL-MPL-MOP.

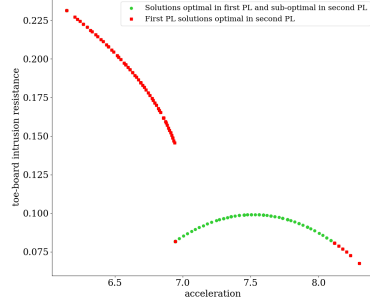


Fig. 8.7: Original optimal solutions on the toe-acceleration plan: in red the optimal solutions for both PLs.

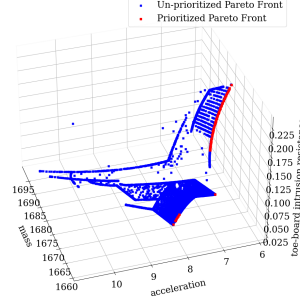


Fig. 8.8: Effect of the priority on the 3D Pareto front. Only the red solutions are truly optimal.

8.10 Algorithm for PL: PL-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, it is presented for NSGA-II only, yet nothing prevents the very same approach to be exploited in other frameworks as well. NSGA-II core operations are already illustrated Section 8.6 and are reported here too to simplify the reading and help the reader understanding: (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.

8.10.1 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,

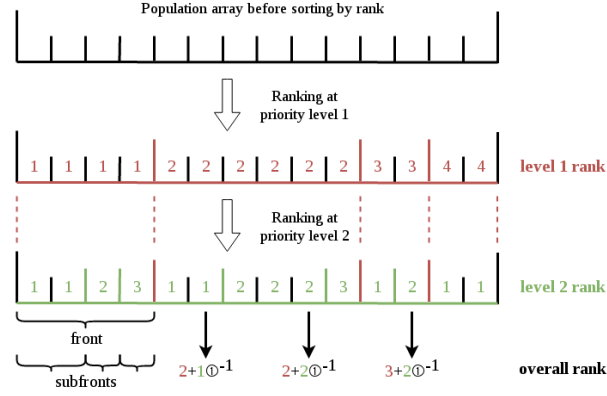


Fig. 8.9: An example of ranking procedure based on Definition 8.3.

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 8.9. Refer to Algorithm 3-4 for the corresponding pseudocode.

Algorithm 3 Priority Levels fast non-dominated sort.

```

1: /* This function is recursive */
2: /* P is the population, lvl is the current level to consider */
3: procedure PL_FAST_NONDOMINATED_SORT(P, lvl)
4:   /* Base case of recursion */
5:   if lvl < min_lvl then return
6:   /* The first iteration also initializes all ranks to 0 */
7:   if lvl == 0 then
8:     for all p ∈ P do
9:       prank = 0
10:  /* Ranking within the priority level to determine subfronts */
11:  F(lvl) = fast_nondom_sort_in_level(P, lvl)
12:  /* Repeat for every subfront found */
13:  for all Fi ∈ F(lvl) do
14:    /* In the next priority level */
15:    Fi = PL_fast_nondominated_sort(Fi, lvl-1)

```

Algorithm 4 Fast non-dominated sort in each level

```

1: procedure FAST_NONDOM_SORT_IN_LEVEL( $P, lvl$ )
2:   for all  $p \in P$  do
3:      $S_p = \emptyset$ 
4:      $n_p = 0$ 
5:     for all  $q \in P$  do
6:       /* non-dominance at specified priority level */
7:       if  $p \prec^{lvl} q$  then
8:          $S_p = S_p \cup \{q\}$ 
9:       else if  $q \prec^{lvl} p$  then
10:         $n_p = n_p + 1$ 
11:     if  $n_p == 0$  then
12:       /* First subfront within P */
13:        $\underline{p}_{rank} = \underline{p}_{rank} + \textcircled{1}^{lvl}$ 
14:        $F_1 = F_1 \cup \{p\}$ 
15:       /* Start from the first subfront */
16:        $\underline{i} = \textcircled{1}^{lvl}$ 
17:       while  $F_{\underline{i}} \neq \emptyset$  do
18:          $Q = \emptyset$ 
19:         for all  $p \in F_{\underline{i}}$  do
20:           for all  $q \in S_p$  do
21:              $n_q = n_q - 1$ 
22:             if  $n_q == 0$  then
23:               /* q belongs to the i-th subfront */
24:                $\underline{q}_{rank} = \underline{q}_{rank} + \underline{i} + \textcircled{1}^{lvl}$ 
25:                $Q = Q \cup \{q\}$ 
26:             /* Index of the next subfront */
27:              $\underline{i} = \underline{i} + \textcircled{1}^{lvl}$ 
28:            $F_{\underline{i}} = Q$ 

```

The proposed *PL_crowding_distance_assignment* is described in Algorithm 5, and it basically consists in a G-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 $\textcircled{1}^0$; then, the same happens considering the second level objectives only, with weight $\textcircled{1}^{-1}$; this is repeated for every level, decreasing the G-weight after each step; finally, all the partial results are added together to form a G-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 5 Priority Levels crowding distance assignment.

```

1: /*  $F$  is a leaf-front in the hierarchy of population fronts partitioning */
2: procedure PL_CROWDING_DIST_ASSIGNMENT( $F$ )
3:    $n = |F|$ 
4:   for all  $i \in F$  do
5:      $F[i]_{dist} = 0$ 
6:   /* For each level of priority  $q$ ;  $p$  is the index of the last level */
7:   for  $q = 1 \dots p$  do
8:     for  $j = 1 \dots m_q$  do
9:        $F = \text{sort}(F, f_j^{(q)})$ 
10:      /* +Inf means "full-scale" (IEEE 754 standard) */
11:       $F[1]_{dist} += +\text{Inf} \textcircled{1}^{1-q}$ 
12:       $F[n]_{dist} += +\text{Inf} \textcircled{1}^{1-q}$ 
13:      for  $i = 2 \dots (n - 1)$  do
14:        /* PL_crowd_dist_ass. has infinitesimal parts */
15:         $F[i]_{dist} += \textcircled{1}^{1-q} \frac{f_j^{(q)}(F[i+1]) - f_j^{(q)}(F[i-1])}{f_j^{(q)max} - f_j^{(q)min}}$ 

```

8.10.2 PL-NSGA-II

By incorporating both *PL_fast_nondominated_sort* and *PL_crowding_distance_assignment* into NSGA-II, one obtains an enhanced version that is also capable of handling PL-MPL-MOPs in a proper way; the name of this algorithm is *PL-NSGA-II*. Now, 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 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, which is another very promising research direction. The pseudocode of PL-NSGA-II is specified in Algorithm 6.

Algorithm 6 PL-NSGA-II algorithm.

```

1: /*  $P_0$  is the starting population,  $T$  is the total number of iterations */
2: /* The function returns the approximated Pareto front  $P_T$  */
3: procedure PL_NSGA_II ( $P_0, T$ )
4:   for  $t = 0 \dots T - 1$  do
5:      $Q_t = \text{make\_new\_pop}(P_t)$ 
6:      $R_t = P_t \cup Q_t$ 
7:     /*  $F$  is the set of all the non-dominated subfronts */
8:      $F = \text{PL\_fast\_nondom\_sort}(R_t, 0)$ 
9:      $P_{t+1} = \emptyset$ 
10:    /*  $i$  is a G-index, a.k.a. G-scalar */
11:     $\underline{i} = 1 \textcircled{1}^0$ 
12:    while  $|P_{t+1}| + |F_{\underline{i}}| \leq N$  do
13:      /* Crowding distance is computed within subfront  $F_{\underline{i}}$  */
14:       $\text{PL\_crowding\_dist\_assignment}(F_{\underline{i}})$ 
15:       $P_{t+1} = P_{t+1} \cup F_{\underline{i}}$ 
16:      /* Move to the next subfront */
17:       $\underline{i} = \text{next\_index}(F, \underline{i})$ 
18:       $\text{Sort}(F_{\underline{i}}, \prec_n^*)$ 
19:       $P_{t+1} = P_{t+1} \cup F_{\underline{i}}[1 : (N - [P_{t+1}])]$ 
return  $P_T$ 

```

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. This algorithm was chosen for its simplicity and similarity to the Grossone-based generalization of NSGA-II.

8.11 Test cases for PL-MPL-MOPs

As for PC-MPL-MOPs, there is a lack of existing PL-MPL-MOPs test problems too. The following paragraphs deal with both synthetic problems and a real-world one, which is an adaptation of the 10-objective GAA problem [51]. The latter allows to introduce of levels of priority quite naturally (given the insights of a domain expert, and it has thus been renamed PL-GAA. Finally, the PL-Crash problem already introduced in Section 8.8.1 is revisited.

The performance of PL-NSGA-II is compared against seven classic EMO algorithms: 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 [47] (Schmiedle et al.), and vii) the dynamic prioritized goal-based approach proposed in [49] (Tan et al.), already considered in Section 8.7 for priority chains. NSGA-III prefiltered is not considered as an interesting option because the number of objectives at the first level is too low to justify the use of a many-objective optimizer. Note that the performances of direction-based algorithms such as NSGA-III and MOEA/D would be better if the priority information was somehow used to guide the search: this idea is left for a future study. As in Section 8.7, SBX crossover and polynomial mutation operators are adopted, each algorithm run 50 times per benchmark.

8.11.1 Test problem: PL-1

PL-1 has five objectives, two of which are less important than the others.

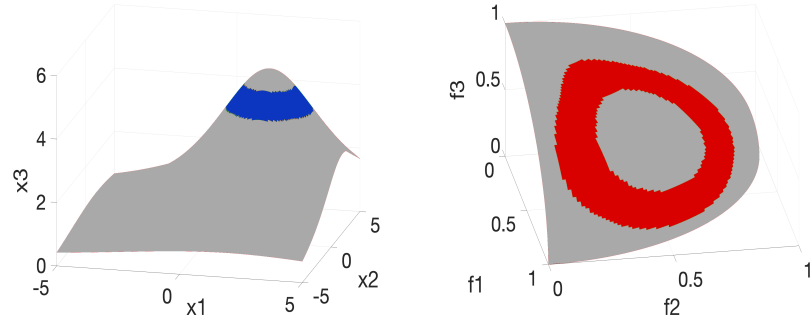
$$\begin{aligned}
& \min \left[\begin{pmatrix} f_1^{(1)}(x) \\ f_2^{(1)}(x) \\ f_3^{(1)}(x) \end{pmatrix} + \mathbb{1}^{-1} \begin{pmatrix} f_1^{(2)}(x) \\ f_2^{(2)}(x) \end{pmatrix} \right], \\
& f_1^{(1)}(x) = \cos\left(\frac{\pi}{10}x_1\right) \cos\left(\frac{\pi}{10}x_2\right), \\
& f_2^{(1)}(x)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

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 $g(x) = 0$) is an octant of a unit sphere. In the variable space, it is generated by the following set of points:

$$\mathcal{S} := \left\{ (x_1, x_2, x_3) \in \mathbb{R}^3 \mid x_1 \in [0, 5], x_2 \in [0, 5], x_3 = 45 / (9 + (x_1 - 2)^2 + (x_2 - 2)^2) \right\},$$

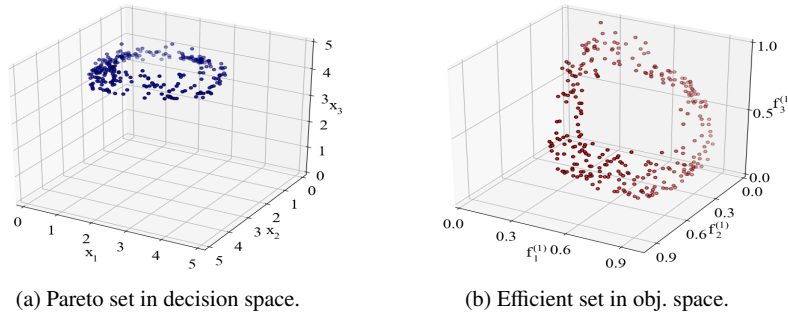
which corresponds to a bell-shaped curve on the x_3 -axis. Figure 8.10a reports it in gray. Among all Pareto-optimal solutions for the primary objectives, i.e., the points in $\mathcal{S}$, secondary objectives $f_1^{(2)}$ and $f_2^{(2)}$ make non-dominated only the points within the blue annular region. Moving to the objective space, the entire positive octant represents the efficient set in the primary objective space, while only the red region is it also for the whole problem, i.e., considering also the secondaries (see Figure 8.10b).

The solution found by PL-NSGA-II for PL-1 is the one in Figure 8.11: the quality of the solutions found by PL-NSGA-II is evident. Indeed, the set of optimal points both in the decision (Figure 8.11a) and in the objective space (Figure 8.11b) seems to closely fit the theoretical expectations of Figure 8.10. To obtain it, PL-NSGA-II run for 500 generations with a population of 100 individuals, and the same holds for all the other algorithms. Table 8.3 reports the mean and the standard deviation of their performance. 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 stabler.



(a) Bell-shaped surface satisfying $g(x) = 0$. (b) Resulting efficient set in objective space.

Fig. 8.10: PL-1 theoretical Pareto set (blue) and efficient set (red).



(a) Pareto set in decision space.

(b) Efficient set in obj. space.

Fig. 8.11: Obtained solutions by PL-NSGA-II for PL-1.

Table 8.3: PL-1: Mean and Std. Dev. for metric $\Delta(\cdot)$.

Algorithm	Mean	Std
PL-NSGA-II	$0.002 + 0.43\mathbb{D}^{-1}$	$0.002 - 0.09\mathbb{D}^{-1}$
MOEA/D-post	$0.01 + 2.18\mathbb{D}^{-1}$	$0.01 + 0.33\mathbb{D}^{-1}$
MOEA/D-pre	$0.01 + 2.94\mathbb{D}^{-1}$	$0.01 + 0.90\mathbb{D}^{-1}$
NSGA-II-pre	$0.01 + 3.53\mathbb{D}^{-1}$	$0.01 + 0.53\mathbb{D}^{-1}$
NSGA-II-post	$0.15 + 2.42e4\mathbb{D}^{-1}$	$0.03 - 6.94e3\mathbb{D}^{-1}$
NSGA-III-post	$0.49 + 1.47e3\mathbb{D}^{-1}$	$0.23 + 8.11e3\mathbb{D}^{-1}$
Tan et al.	$6.63 + 6.62e4\mathbb{D}^{-1}$	$7.16 - 3.42e4\mathbb{D}^{-1}$
Schmiedle et al.	$8.92 + 4.10e4\mathbb{D}^{-1}$	$6.8 - 7.12e3\mathbb{D}^{-1}$

8.11.2 PL-GAA: a real-world problem

The GAA benchmark [51] is a difficult many-objective problem from a real-world scenario. It challenges the decision maker to tune 27 con-

strained 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 one 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, the problem has been reformulate dividing the 10 objectives in 4 PLs, obtaining the new problem PL-GAA:

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

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 [51], the decision-maker can tolerate a diversity level not greater than 0.05. Thus, the PFPF output space is discretized 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 ($\text{PFPF} \in [0, 0.05]$) 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 [51], 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).

In the experiments, due to the complexity of the problem, each algorithm run with 200 individuals for 2,000 generations. Moreover, due

Table 8.4: PL-GAA: Mean and Std. Dev. for metric $\Delta(\cdot)$ (NSGA-II-pre and MOEA/D-pre not reported, being their use meaningless on this problem).

Algorithm	Mean	Std. Dev.
PL-NSGA-II	$4.29e3\textcircled{-1} + 1.91e4\textcircled{-2} + 21.82\textcircled{-3}$	$1.65e4\textcircled{-1} + 3.23e4\textcircled{-2} + 4.04e4\textcircled{-3}$
Schmiedle et al.	$0.06 + 1.00e4\textcircled{-1} + 7.74e4\textcircled{-2} + 19.7\textcircled{-3}$	$0.24 - 1.30e3\textcircled{-1} + 5.63e8\textcircled{-2} + 3.08e12\textcircled{-3}$
NSGA-II-post	$1.24 + 1.12e5\textcircled{-1} + 6.66e5\textcircled{-2} + 482.83\textcircled{-3}$	$1.36 - 1.12e4\textcircled{-1} + 2.55e9\textcircled{-2} + 2.08e13\textcircled{-3}$
NSGA-III-post	$3.9 + 3.24e4\textcircled{-1} + 4.13e5\textcircled{-2} + 214.36\textcircled{-3}$	$4.99 + 1.13e4\textcircled{-1} + 8.48e7\textcircled{-2} - 1.91e11\textcircled{-3}$
Tan et al.	$102.55 + 8.90e4\textcircled{-1} + 5.48e5\textcircled{-2} + 421.65\textcircled{-3}$	$1.22e2 + 1.23e4\textcircled{-1} + 1.92e7\textcircled{-2} - 1.99e9\textcircled{-3}$
MOEA/D-post	$113.08 + 8.23e4\textcircled{-1} + 8.66e5\textcircled{-2} + 1.47e3\textcircled{-3}$	$26.11 - 5.64e3\textcircled{-1} + 9.56e7\textcircled{-2} + 2.01e10\textcircled{-3}$

to the presence of constraints also on the objectives, the pre-filtered approaches have been excluded from the performance comparison, since they necessarily perform poorly. Table 8.4 reports the algorithms average performance (the performance of NSGA-II-pre and MOEA/D-pre have not been reported, being their use on this problem clearly meaningless). 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 is, the higher the average decision maker dissatisfaction is; the higher the variance is, the higher the randomness of the performance quality is.

The numerical evidences tell us that only two algorithms are able to respect such a constraint, namely PL-NSGA-II (always) and Schmiedle et al. (most of the cases). This fact is quite interesting indeed, while PL-NSGA-II is designed to properly master the Pareto dominance also in the presence of priority levels (PL-dominance), Schmiedle et al. does not use the Pareto dominance at all. Instead, it leverages on the *favoring* binary relation (see [47]), suggesting to address to this aspect the reasons of its high performances and capability to satisfy the PFPF's constraint. 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.

8.11.3 Test problem: PL-Crash

This section provides brief experimental results on the PL-Crash problem already discussed in the previous sections of this work. Table 8.5 contains

Table 8.5: PL-Crash: Mean and Std. Dev. for metric $\Delta(\cdot)$.

Algorithm	Mean	Std
PL-NSGA-II	$0.002 + 2.99e - 5 \textcircled{1}^{-1}$	$0.02 + 2.00e - 4 \textcircled{1}^{-1}$
NSGA-II-post	$1.00 + 0.52 \textcircled{1}^{-1}$	$0.22 - 0.001 \textcircled{1}^{-1}$
NSGA-II-pre	$1.12 + 0.57 \textcircled{1}^{-1}$	$0.23 - 0.01 \textcircled{1}^{-1}$
MOEA/D-post	$14.29 + 1.80 \textcircled{1}^{-1}$	$11.76 - 0.37 \textcircled{1}^{-1}$
MOEA/D-pre	$19.50 + 1.43 \textcircled{1}^{-1}$	$12.12 - 0.30 \textcircled{1}^{-1}$
Tan et al.	$29.96 + 1.73 \textcircled{1}^{-1}$	$16.97 + 0.48 \textcircled{1}^{-1}$
NSGA-III-post	$94.51 + 4.51 \textcircled{1}^{-1}$	$44.14 + 1.55 \textcircled{1}^{-1}$
Schmiedle et al.	$143.14 + 10.76 \textcircled{1}^{-1}$	$32.06 + 2.23 \textcircled{1}^{-1}$

the mean and standard deviation values of the $\Delta(\cdot)$ metric. Again, PL-NSGA-II significantly outperforms the competitors.

8.12 Conclusions

In this chapter the novel and very general class of Mixed Pareto-Lexicographic Multi-Objective Problems has been introduced and discussed. The peculiarity of these problems is to have the objective functions organized according to a priority structure, and different structures lead to different models. In particular, two models have been deeply investigated: PC-MPL-MOPs, where the objective are organized in chains of priority; and PL-MPL-MOPs, where the functions are grouped in levels which are ordered by priority. In order to solve the intrinsic limits of standard approaches when dealing with these two classes of MPL-MOPs, the Grossone Methodology has been leveraged to achieve two results: i) a reformulation of the models which is easier to be processed by a computer; ii) the generalization of standard EMO algorithms in order to make them able to cope with these more general problems. The effectiveness of the generalizations has been tested on several benchmark and compared with several well-known algorithms (Grossone-based algorithms have been evaluated on an Infinity Computer simulator implemented by the authors). The results testify that the additional information about priority relations among objectives can be exploited to significantly improve the algorithms search, guiding them towards higher quality solutions than those provided by conventional multi-objective algorithms.

References

1. Adra, S. F., Fleming, P. J.: Diversity management in evolutionary many-objective optimization. *IEEE Trans. on Evolutionary Computation* **15**, 183–195 (2011)
2. Astorino, A., Fuduli, A.: Spherical separation with infinitely far center. *Soft Computing* **24**(23), 17751–17759 (2020)
3. Cavoretto, R., De Rossi, A., Mukhametzhonov, M. S., Sergeyev, Y. D.: On the search of the shape parameter in radial basis functions using univariate global optimization methods. *Journal of Global Optimization* **79**(2), 305–327 (2021)
4. Chang, P.-C., Hsieh, J.-C., Lin, S.-G.: The development of gradual-priority weighting approach for the multi-objective flowshop scheduling problem. *International Journal of Production Economics* **79**, 171–183 (2002)
5. Cococcioni, M., Pappalardo, M., Sergeyev, Y. D.: Towards Lexicographic Multi-Objective Linear Programming using Grossone Methodology. *Proc. of the 2nd Intern. Conf. “Numerical Computations: Theory and Algorithms”*. AIP Conference Proceedings **1776**, 90040 (2016)
6. Cococcioni, M., Pappalardo, M., Sergeyev, Y. D.: Lexicographic Multi-Objective Linear Programming using Grossone Methodology: Theory and Algorithm. *Applied Mathematics and Computation* **318**, 298–311 (2018)
7. Coello Coello, C. A., Sierra, M. R.: A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. *Mexican Int. Conf. on Art. Int.*, 688–697 (2004)
8. De Cosmis, S., De Leone, R.: The use of Grossone in Mathematical Programming and Operations Research. *Applied Mathematics and Computation* **218**(16) 8029–8038 (2012)
9. De Leone, R.: Nonlinear Programming and Grossone: Quadratic Programming and the role of Constraint Qualifications. *Applied Mathematics and Computation* **218**(16) 290–297 (2018)
10. De Leone, R., Fasano, G., Roma, M., Sergeyev, Y. D.: Iterative Grossone-Based Computation of Negative Curvature Directions in Large-Scale Optimization. *Journal of Optimization Theory and Applications* **186**, 554–589 (2020).
11. De Leone, R., Fasano G., Sergeyev, Y. D.: Planar methods and grossone for the Conjugate Gradient breakdown in nonlinear programming. *Computational Optimization and Applications* **71**(1), 73–93 (2018)
12. Deb, K.: *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons **16** (2001)
13. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Trans. on Evolutionary Computation* **18**, 577–601 (2014)
14. Deb, K., Joshi, D., Anand, A.: Real-coded evolutionary algorithms with parent-centric recombination. In *Proc. of the 2002 Congress on Evol. Comp.* **1**, 61–66 (2002)
15. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. on Evolutionary Computation*, **6**, 182–197 (2002)
16. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multi-objective optimization. *Evolutionary Multiobjective Optimization*, 105–145 (2005)
17. Fonseca, C. M., Fleming, P. J.: 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* **28**, 26–37 (1998)
18. García, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms’ behaviour: a case study on the CEC’2005 special session on real parameter optimization. *Journal of Heuristics* **15**, 617 (2009)
19. Garza-Fabre, M., Pulido, G. T., Coello Coello, C.A.: Ranking methods for many-objective optimization. *Mexican International Conference on Artificial Intelligence*, 633–645 (2009)
20. Gaudioso, M., Giallombardo, G., Mukhametzhonov M.: Numerical infinitesimals in a variable metric method for convex nonsmooth optimization. *Applied Mathematics and Computation* **318**, 312–320 (2018)
21. Gaur, A., Khaled Talukder, A., Deb, K., Tiwari, S., Xu, S., Jones, D.: Unconventional optimization for achieving well-informed design solutions for the automobile industry. *Engineering Optimization* **52**, 1542–1560 (2020)

22. Gergel, V., Grishagin, V., Israfilov, R.: Adaptive Dimensionality Reduction in Multiobjective Optimization with Multiextremal Criteria. *Machine Learning, Optimization, and Data Science*, 129–140 (2019)
23. Hansen, M. P., Jaszkievicz, A.: Evaluating the quality of approximations to the non-dominated set. IMM, Department of Mathematical Modelling, TU Denmark (1994)
24. Holm, S.: A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, 65–70 (1979)
25. Iman, R. L., Davenport, J. M.: Approximations of the critical region of the fbietkan statistic. *Communications in Statistics-Theory and Methods* **9**, 571–595 (1980)
26. Ishibuchi, H., Masuda, H., Tanigaki, Y., Nojima, Y.: Modified distance calculation in generational distance and inverted generational distance. *International Conference on Evolutionary Multi-Criterion Optimization*, 110–125 (2015)
27. Lai, L., Fiaschi, L., Cococcioni, M.: Solving mixed Pareto-Lexicographic multi-objective optimization problems: The case of priority chains. *Swarm and Evolutionary Computation* **55**, 100687 (2020)
28. Lai, L., Fiaschi, L., Cococcioni, M., Deb, K.: Solving Mixed Pareto-Lexicographic Multi-Objective Optimization Problems: The Case of Priority Levels. *IEEE Transactions on Evolutionary Computation* (2021)
29. Lai, L., Fiaschi, L., Cococcioni, M., Deb, K.: Handling Priority Levels in Mixed Pareto-Lexicographic Many-Objective Optimization Problems. In *Proc. of the 2021 Int. Conf. on Evolutionary Multi-Criterion Optimization*, Shenzhen, China, 362–374, (2021)
30. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation* **10**, 263–282 (2002)
31. Li, K., Deb, K., Kwong, S.: An Evolutionary Many-Objective Optimization Algorithm Based on Dominance and Decomposition. *IEEE Trans. Evol. Comp.* **19**, 694–716 (2015)
32. Li, H., Deb, K., Zhang, Q., Suganthan, P. N., Chen, L.: Comparison between MOEA/D and NSGA-III on a set of novel many and multi-objective benchmark problems with challenging difficulties. *Swarm and Evolutionary Computation* **46**, 104–117 (2019)
33. Li, M., Yao, X.: Quality evaluation of solution sets in multiobjective optimisation: A survey. *ACM Computing Surveys (CSUR)* **52**, 1–38 (2019)
34. Liao, X., Li, Q., Yang, X., Zhang, W., Li, W.: Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Structural and multidisciplinary optimization* **35**, 561–569 (2008)
35. Khare, V., Yao, X., Deb, K.: Performance scaling of multi-objective evolutionary algorithms. *Int. Conf. on Evolutionary Multi-Criterion Optimization*, 376–390 (2003)
36. Khosravani, S., Jalali, M., Khajepour, A., Kasaiezadeh, A., Chen, S. K., Litkouhi, B.: Application of Lexicographic Optimization Method to Integrated Vehicle Control Systems. *IEEE Trans. on Industrial Electronics* **65**, 9677–9686 (2018)
37. Knowles, J., Corne, D.: On metrics for comparing nondominated sets. *IEEE Proceedings of the 2002 Congress on Evolutionary Computation* **1**, 711–716 (2002)
38. Marler, R. T., Arora, J. S.: Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization* **26**, 369–395 (2004)
39. Marques-Silva, J., Argelich, J., Graça, A., Lynce, I.: Boolean lexicographic optimization: algorithms & applications. *Annals of Mathematics and Art. Int.* **62**, 317–343 (2011)
40. Miettinen, K.: *Nonlinear Multiobjective Optimization*. Springer Science, NY (1999)
41. Poloni, C.: Hybrid GA for multi objective aerodynamic shape optimisation. *Genetic Algorithms in Engineering and Computer Science*, 397–415 (1995)
42. Purshouse, R. C., Fleming, P. J.: Evolutionary many-objective optimisation: An exploratory analysis. *The 2003 IEEE Congress on Evol. Comput.* **3**, 2066–2073 (2003)
43. Sergeyev, Y. D.: The Olympic medals ranks, lexicographic ordering, and numerical infinities. *The Mathematical Intelligencer* **37**(2), 4–8 (2015)
44. Sergeyev, Y. D.: Numerical infinities and infinitesimals: Methodology, applications, and repercussions on two Hilbert problems. *EMS Surveys Math. Science* **4**(2), 219–320 (2017)

45. Sergeyev, Y. D.: Independence of the Grossone-Based Infinity Methodology from Non-standard Analysis and Comments upon Logical Fallacies in Some Texts Asserting the Opposite. *Foundations of Science*, **24**(1), 153–170 (2019)
46. Sergeyev, Y. D., Nasso M. C., Mukhametzhano, M. S. , Kvasov, D. E.: Novel local tuning techniques for speeding up one-dimensional algorithms in expensive global optimization using Lipschitz derivatives. *Journal of Computational and Applied Mathematics* **383**, 113134 (2021)
47. Schmiedle, F., Drechsler, N., Große, D., Drechsler, R.: Priorities in multi-objective optimization for genetic programming. *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, 129–136 (2001)
48. Schutze, O., Esquivel, X., Lara, A., Coello Coello, C. A.: Using the averaged Hausdorff distance as a performance measure in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation* **16**, 504–522 (2012)
49. Tan, K. C., Khor, E. F., Lee, T. H., Sathikannan, R.: An evolutionary algorithm with advanced goal and priority specification for multi-objective optimization. *Journal of Artificial Intelligence Research* **18**, 183–215 (2003)
50. Van Veldhuizen, D. A.: Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Air Force Institute of Technology Wright Patterson AFB, OH, USA (1999)
51. Wang, L., Ng, A. H. C., Deb, K.: Multi-objective evolutionary optimisation for product design and manufacturing. Springer Nature (2011)
52. Yuan, Y., Xu, H., Wang, B., Yao, X.: A new dominance relation-based evolutionary algorithm for many-objective optimization. *IEEE Trans. on Evol. Comp.* **20**, 16–37 (2016)
53. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. on Evolutionary Computation* **11**, 712–731 (2007)
54. Zhang, Q., Zhou, A., Zhao, S., Suganthan, P. N., Liu, W., Tiwari, S.: Multiobjective optimization test instances for the CEC 2009 special session and competition. University of Essex, Colchester, and Nanyang technological University, Singapore, TR **264** (2008)
55. Zitzler, E., Laumanns, M., Thiele, L. :SPEA2: Improving the strength Pareto evolutionary algorithm. TIK-report 103 (2001)
56. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms: a comparative case study. *International conference on parallel problem solving from nature*, 292–301 (1998)