UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Máster Universitario en Investigación en Inteligencia Artificial

Trabajo Fin de Máster

Self-Adaptive Polynomial Mutation in Multi-Objective Evolutionary Algorithms

Autor: **JOSÉ LUIS CARLES BOU**

Dirigido por: **SEVERINO FERNÁNDEZ GALÁN**

Curso 2021-2022 – Convocatoria de junio

Contents

Self-Adaptive Polynomial Mutation in Multi-Objective Evolutionary Algorithms

Jose L. Carles
MSc Student at UNED

May 23, 2022

Abstract

Evolutionary multi-objective optimization is a field that has experienced a rapid growth in the last two decades. Although an important number of new multi-objective evolutionary algorithms have been designed by the scientific community, the popular Non-Dominated Sorting Genetic Algorithm (NSGA-II) remains as a widely used baseline for performance comparison purposes. Since every evolutionary algorithm needs several parameters to be set up in order to operate, parameter control constitutes a crucial task for the effective and efficient performance of multi-objective evolutionary algorithms. However, despite the advancements in parameter control for evolutionary algorithms, NSGA-II has been mainly used in the literature with fine-tuned static parameters. This paper introduces a novel and computationally lightweight self-adaptation mechanism for controlling the *distribution index* parameter of the *polynomial mutation* operator usually employed by NSGA-II in particular and by multi-objective evolutionary algorithms in general. Additionally, the classical NSGA-II using polynomial mutation with a static distribution index is compared with this new version utilizing a self-adapted parameter. The experiments carried out over twenty-five benchmark problems using three quality indicators (hypervolume, generalized spread, and modified inverted generational distance) show that the proposed self-adaptive mutator variant outperforms its static counterpart in most of the cases. This result supports the potential of self-adaptive parameter control in multi-objective evolutionary algorithms.

Keywords: multi-objective evolutionary algorithm, NSGA-II, polynomial mutation, distribution index self-adaptation

1 Introduction

A high percentage of real-world optimization problems are multi-objective by nature. In this kind of problems, the objectives being optimized are usually in conflict with each other. Thus, rather than a unique optimal solution, there is a set of them known as the Pareto set. Multi-Objective Evolutionary Algorithms (MOEAs) have the ability to effectively approximate this Pareto set, as reported in the scientific literature over more than three decades.

The Non-Dominated Sorting Genetic Algorithm II (NSGA-II) was proposed by Deb *et al.* in 2000 [DAPM00] as an exploration tool that employs a genetic algorithm to solve multi-objective problems (MOPs). NSGA-II is nowadays one of the most used and cited MOEAs. It has been applied to a broad variety of optimization problems since its inception. The NSGA-II algorithm creates a random population, selects individuals and applies genetic operations on them, ranks and sorts each individual based on its non-domination level, and applies a crowding distance operation to keep the evolving population diverse.

The fine-tuning of the parameters of an evolutionary algorithm is a key aspect that directly impacts on both its efficacy and its efficiency. Obtaining the optimal set of parameters for the problem at hand is a hard and time-consuming task. Parameter tuning and control in evolutionary algorithms has been a relevant research topic since their creation. In the literature, when NSGA-II is used as a baseline algorithm in benchmarks, its parameters are usually configured with static values. Normally, simulated binary crossover (SBX) and polynomial mutation (PLM) are carried out with fixed probabilities and distribution indices. However, Hamdan demonstrated in [Ham12] that the fixed distribution index utilized in PLM does not always provide the best performance results. Furthermore, Deb *et al.* proposed in [DSO07] a method for self-adapting the SBX operator in order to improve the algorithm performance. Therefore, these two relevant works show that it is not fair to use NSGA-II for comparative evaluation purposes when its parameters are kept static.

This work proposes a novel and computationally lightweight mechanism that self-adapts one of the parameters of the PLM operator, its distribution index η_m, producing an increased efficacy over the results observed in traditional NSGA-II. We compare the performance of this new algorithm variant to that of plain NSGA-II utilizing the static configuration widely found in the literature. A set of twenty-five benchmark problems from different test suites are used (DTLZ, WFG, ZDT, and other key problems), which experimentally confirm that this novel technique provides better results for several quality indicators (hypervolume, generalized spread, and modified inverse generational distance).

The rest of this paper is structured as follows. Section 2 introduces several concepts on evolutionary multi-objective optimization and reviews related work. Section 3 describes the novel NSGA-II variant. Experimental results and analysis are presented in Section 4. Finally, Section 5 concludes the paper and suggests some future research directions.

2 Background and related work

In the present section, we initially cover some important concepts used in multi-objective optimization. Then, we review the latest work on: (i) parameter control techniques in evolutionary algorithms and (ii) quality metrics for measuring and comparing the performance of MOEAs.

2.1 Multi-objective optimization concepts

An MOP is defined as the simultaneous optimization of several objective functions over a tuple of decision variables. Without loss of generality, if minimization is considered for all the objectives, this can be formally expressed as follows:

$$\text{minimize } \bar{y} = f(\bar{x}) = (f_1(\bar{x}), f_2(\bar{x}), ..., f_m(\bar{x})) \tag{1}$$

where $\bar{x} = \langle x_1, x_2, ..., x_n\rangle \in X \subseteq \mathbb{R}^n$ is called a *decision vector*, X is an n-dimensional *decision space*, $\bar{y} = \langle y_1, y_2, ..., y_m\rangle \in Y \subseteq \mathbb{R}^m$ with $m \geq 2$ is an *objective vector*, Y represents an m-dimensional *objective space*, n is the number of decision variables, m is the number of objective functions, and f_i corresponds to the ith objective function.

The set of decision vectors whose objectives cannot be improved in any direction without the degradation of another direction is called the Pareto optimal set. The concept of Pareto optimality can be defined as follows. Given a multi-objective minimization problem and two decision vectors $\bar{a}, \bar{b} \in X$, then $\bar{a}$ is said to *dominate* $\bar{b}$ (written as $\boldsymbol{\bar{a} \prec \bar{b}}$) if and only if $\bar{a}$ is no worse than $\bar{b}$ in every objective and $\bar{a}$ is strictly better than $\bar{b}$ in at least one objective:

$$\begin{aligned} \forall i \in \{1, 2, ..., m\} &\ :\ f_i(\bar{a}) \leq f_i(\bar{b}) \\ \text{and } \exists j \in \{1, 2, ..., m\} &\ :\ f_j(\bar{a}) < f_j(\bar{b}) \end{aligned} \tag{2}$$

If neither $\bar{a}$ dominates $\bar{b}$ nor $\bar{b}$ dominates $\bar{a}$, $\bar{a}$ and $\bar{b}$ are said to be non-comparable, also stated as $\bar{a} \sim \bar{b}$. All the decision vectors which are not dominated by any other in a given set are called *non-dominated* regarding the set. The non-dominated vectors in the entire search space are called *Pareto optimal solutions* and form the *Pareto optimal solution set* or just the *Pareto set*:

$$PS := \{\bar{x} \in X : \nexists\ \bar{x}' \in X,\ \bar{x}' \prec \bar{x}\} \tag{3}$$

and the projection of the *Pareto set* in the objective space is known as the *Pareto front*:

$$PF := \{(f_1(\bar{x}), f_2(\bar{x}), ..., f_m(\bar{x})) : \bar{x} \in PS\} \tag{4}$$

If we relax the domination condition, we can say that $\bar{a}$ *weakly dominates* $\bar{b}$, or $\bar{a} \preceq \bar{b}$, if $\bar{a}$ is no worse than $\bar{b}$ in every objective:

$$\forall i \in \{1, 2, ..., m\}\ :\ f_i(\bar{a}) \leq f_i(\bar{b}) \tag{5}$$

We can also consider the concept of *strict domination*, saying that $\bar{a}$ *strictly dominates* $\bar{b}$, or $\bar{a} \prec\prec \bar{b}$, if $\bar{a}$ is better than $\bar{b}$ in every objective:

$$\forall i \in \{1, 2, ..., m\} \ : \ f_i(\bar{a}) < f_i(\bar{b}) \tag{6}$$

The set of objective vectors $A \subset \mathbb{R}^m$ is called an *approximation set* if any of the elements in A does not dominate any other vector in the set [HJ98]. Usually, the ultimate objective when solving MOPs is not to find the real PF but a *good* approximation to it [RLB15]. Several tools to evaluate the quality of these non-dominated sets are explained in Section 2.3.

Even though the terms "solution" and "objective vector" have been used interchangeably [NLA$^+$08, LAC16, IISN18], we prefer to keep the term solution for a vector in the decision space and the term objective vector for a solution projected into the objective space in order to avoid misunderstandings [TI20].

2.2 Parameter tuning and control

Once the general scheme of an evolutionary algorithm is established, the researcher has to set the population size, choose the genetic operators for parent selection, crossover, mutation and survivor selection, and fix their probabilities. These specific parameters, or *configuration*, determine the behavior of the algorithm, guide its search and impact directly on the efficacy and efficiency achieved by the algorithm [HBY22]. It has been experimentally demonstrated that the use of different selection and variation operators as well as the setting of their parameters have an influence on the performance of evolutionary algorithms [SP97, Ham12, DSO07]. It is also true that the parameters might need distinct values at different stages of the execution process in order to adapt the exploration and exploitation intensities to the landscape it is dealing with [Bä92].

In the course of evolutionary algorithms history, different approaches to addressing the necessity of parameter dynamism have been suggested. The first one, and probably the most extended in the literature, is the utilization of a static and promising configuration (known as *parameter tuning*). Researchers use a well known set of operators and parameter values that provide good results for the problem at hand or get them adjusted after some trial-and-error search. But as the number of combinations of tested parameters can be really huge for manual adjustment, several automated tools have been developed. One of the most important tools used nowadays is *irace*, where an iterated execution of the F-race algorithm [BYBS10, LIDLC$^+$16] is implemented relieving the researcher of the tedious and time consuming task of repetitive manual parameter adjustment. Another interesting alternative is employing metaheuristic techniques, like the Meta-GA used by Grefenstette [Gre86], in which an external evolutionary algorithm adjusts the algorithm parameters.

Starting just when evolutionary algorithms were created [Jon75, Gre86], but specially in the last thirty years, a lot of effort has been invested in methods for dynamically controlling the parameters during execution (*parameter control*). In 1995, Angeline [Ang95] presented a taxonomy of these techniques differentiating the type of control based on its level of application (over the full population, the individuals, or their genes). Smith and Fogarty [SF97] proposed a new classification scheme trying to differentiate what is being adapted (parameters or operators), the scope of the adaptation (if it is applied to the full population, individual or gene level) and the tools being applied to make the adaptation. Hinterding, Michalewicz and Eiben [HME97, EHM99] introduced a new classification method based on the type of adaptation and distinguished between deterministic, adaptive and self-adaptive methods. Note that this is still the most widely used taxonomy and the basis, with few and slight variations, of recent work [ZCZ+12, PPSN19, DD20]:

- Eiben *et al.* define *deterministic control* as the adjustment of a parameter using a deterministic rule that does not get any feedback from the search. Usually, it utilizes a rule based on the number of evaluations of the fitness function or on the number of generations passed so far. This kind of methods are very common due to the simplicity of implementation [MMPO09, HAA+19].
- *Adaptive control* uses some characteristic or feedback from the search process to adjust the value of the parameters. Among the adaptive methods, very well known and simple strategies like the 1/5 Rechenberg's success rule [Rec71] can be found. Alternatively, more complex methods are based on learning principles brought from fields like reinforcement learning or deep learning [EHKS07, ASS+19].
- *Self-adaptive control* is close to the underlying idea of evolutionary optimization as the parameters are encoded into the chromosomes and evolve with the rest of variables by applying genetic operations. Thus, the best parameters generate high-quality solutions which survive in future generations [SF96, DSO07, BCSP17, RW20].

Due to the vast amount of publications on parameter control in evolutionary algorithms, several exhaustive reviews have been made available to researchers and practitioners [EHM99, KHE15, AM16], and recent reviews can be found in [PPSN19], [HLY20], [LPN+21] (also covering swarm inspired algorithms), and [HBY22] (focused on differential evolution). In addition, an interesting theoretical approach is developed in [DD20]. Finally, a complete compendium on parameter control and its application is offered in [Pap21] and an interesting compilation about parameter setting is included in [LLM07].

NSGA-II [DAPM00], proposed by Deb *et al.* in 2000, is a well known, competent, and extensively used genetic algorithm for solving MOPs that attains near-optimal, diverse, and uniformly distributed solution sets [TCMG09, RGD+22]. Even though NSGA-II is a mature algorithm, it is still being utilized as a baseline to test new algorithms performance [IYM03, ZLZL19, LLJ22]. Its population size, crossover and mutation operators and probabilities, selection pressure, and even the distribution indices used by the original SBX and PLM operators have to be set in advance.

Although the number of publications about self-adaptation in MOEAs is relatively low compared to single-objective algorithms [AM16], several authors have shown that self-adaptation can improve the applicability of MOEAs to online decision support in real-world problems [ZLD+10]. Deb *et al.* [DSO07] proposed a self-adaptive method for adapting the distribution index under SBX crossover, SA-SBX, used typically in NSGA-II due to the fact that a fixed index does not always produce the best performance results, specially when dealing with multi-modal problems. This parameter, η_c, defines the shape of the probability distribution function which governs the spread of offspring solutions given the parents. The method, applied both to mono and multi-objective problems, produced promising results. In 2010, Zeng *et al.* [ZLD$^+$10] improved Deb's idea of a self-adapting distribution index in the binary crossover operator by using the diversity of the offspring solutions to dynamically control the parameter.

Despite the fact that several studies have focused on the mutation operator by adapting the mutation rate [BS96, YU06], by changing the mutation probability distribution function like in evolution strategies and evolutionary programming [LY04, TY07], or even by changing the mutation operator itself during the evolution [KYL09], very few studied the idea of changing the probability distribution index in PLM, η_m, as Deb did in SA-SBX for crossover. Furthermore, Hamdan [Ham12, Ham14] demonstrated that different values for η_m from the ones found in the literature provided better performance results.

Following the ideas of Deb, Zeng, and Hamdan, this work investigates whether or not self-adapting the mutation distribution index of the typical PLM operator used in NSGA-II could actually outperform the classical static configuration. In this regard, Section 3 proposes a novel and simple self-adaptive mutation operator which, as shown in Section 4, outperforms a static configuration when applied to several benchmark problems.

2.3 Quality indicators

Trying to compare the outputs of two multi-objective optimizers is not trivial as we are comparing two sets of non-dominated vectors or approximation sets. Several *performance metrics*, or *quality indicators*, are defined over an approximation set (*unary metrics*) or between two approximation sets (*binary metrics*) producing a scalar value utilized to compare the quality of the output of different algorithms when solving MOPs [ZTL$^+$03, RLB15].

Ideally, the obtained non-dominated solutions of an approximation set should be as close as possible to the Pareto front, well-distributed, and widely spread [OJS03]. Therefore, it is common to find the metrics capturing the quality of approximation sets grouped as cardinality, accuracy, and diversity metrics:

- *Cardinality metrics* utilize the number of non-dominated vectors in an approximation set, since algorithms producing larger approximation sets are usually preferred.

- *Accuracy metrics* (or convergence metrics) measure the distance between the approximation set and the theoretical optimal front or a reference set if the real Pareto front is unknown [OJS03]. Several methods for calculating this *distance* have been proposed, like the ones defined later in this section for the GD, IGD, and IGD^+ indicators.

- *Diversity metrics* try to capture how well the objective vectors in the approx-

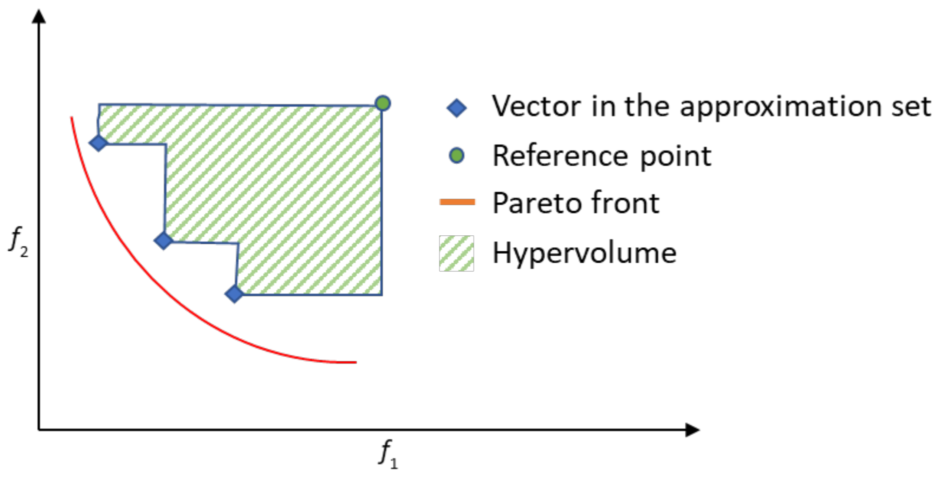

Figure 1: Hypervolume covering the shaded green area in a bi-objective minimization problem.

imation set are distributed (the relative distance between them) and spread (the range of values covered by these vectors) [RLB15].

- *Convergence-diversity metrics* map both the proximity of the objective vectors to the PF and the diversity of these vectors to a scalar value [JOZF14].

Over the last few decades, a large number of papers have been published studying a vast list of available indicators. We remark some of the most cited surveys authored by Riquelme *et al.* [RLB15] studying fifty-four indicators, Li and Yao [LY19] covering one hundred indicators, and Audet *et al.* [ABC+20] describing fifty-seven metrics. The papers published by Hansen and Jaszkiewicz [HJ98] and Zitzler *et al.* [ZTL+03] are relevant for analyzing the theoretical aspects of these quality indicators in a formal way.

Choosing the right indicator, or a set of them, to assess the performance of an MOEA turns out to be as difficult as developing the MOEA itself. Several authors [Kno02, KC02, OJS03, JOZF14, WAY+16] have criticized the usually misleading and sometimes inconsistent results provided by these metrics, and help in the selection of the right one. Following the information provided by these authors, we have finally chosen three metrics from the myriad of them in order to evaluate our new algorithm: the hypervolume indicator, the generalized spread measure and the modified inverted generational distance.

2.3.1 Hypervolume

The *hypervolume indicator* (*HV*) was introduced by Zitzler and Thiele in 1999 [ZT99], and it is one of the most studied and used metrics due to some of its mathematical characteristics. *HV* captures in one single value the convergence and diversity of an approximation set, and it is a strictly monotonic metric with regards to the Pareto dominance concept [KC02, BZ11]. Larger values of *HV* indicate that the vectors in the approximation set are closer to the true PF and evenly distributed over it [JOZF14].

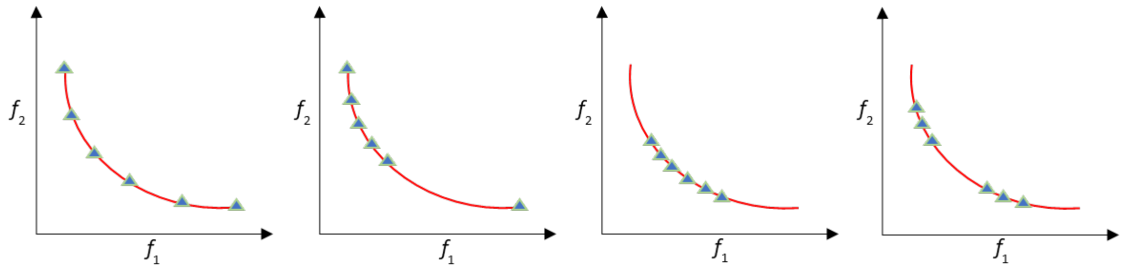

Figure 2: Examples of spread and distribution of vectors over the Pareto front.

HV calculates the hypervolume that is dominated by the objective vectors in the approximation set S bounded by a reference point $\bar{r}$ (see Figure 1, where HV measures the shaded green area). Although the selection of the reference point to calculate HV is critical [ABBZ09, IISN18], it is accepted that a point slightly worse in every objective than every point in S is enough to cover all the vectors in the approximation set. HV is calculated as follows:

$$HV(S,\bar{r}) = \Lambda\left(\bigcup_{i=1}^{|S|} v_i\right) \tag{7}$$

where Λ denotes the Lebesgue measure and v_i represents the box defined by vector i and reference point $\bar{r}$. Larger values of HV indicate better approximation sets.

2.3.2 Generalized spread

The *spread metric* (Δ), introduced by Deb *et al.* in 2000 [DAPM00], is another commonly used indicator in bi-objective MOEAs that tries to measure how uniformly the objective vectors in an approximation set are distributed over the Pareto front. Zhou *et al.* extended this indicator to m-objective problems in the *generalized spread metric* (Δ^*) [ZJZ+06] which is formulated as follows:

$$\Delta^*(S,P) = \frac{\sum_{i=1}^{m} d(e_i,S) + \sum_{\bar{x}\in P} |d(\bar{x},S) - \bar{d}|}{\sum_{i=1}^{m} d(e_i,S) + \bar{d}(|P|-m)}, \tag{8}$$

where e_i is the i-th extreme objective vector in the Pareto front P with the maximum value for the i-th objective function and

$$d(\bar{x},P) = \min_{y\in P, y\neq x} \|\bar{x}-\bar{y}\|_2, \tag{9}$$

$$\bar{d} = \frac{1}{|P|}\sum_{\bar{x}\in P} d(\bar{x},S) \tag{10}$$

The Δ and Δ^* metrics, unlike HV, require a known Pareto front. We can observe in Figure 2: (i) an example of a well distributed and well spread set of vectors in the left-hand graph, (ii) an example of a well spread but not well distributed set in the second graph, (iii) a well distributed but not well spread set of vectors in the

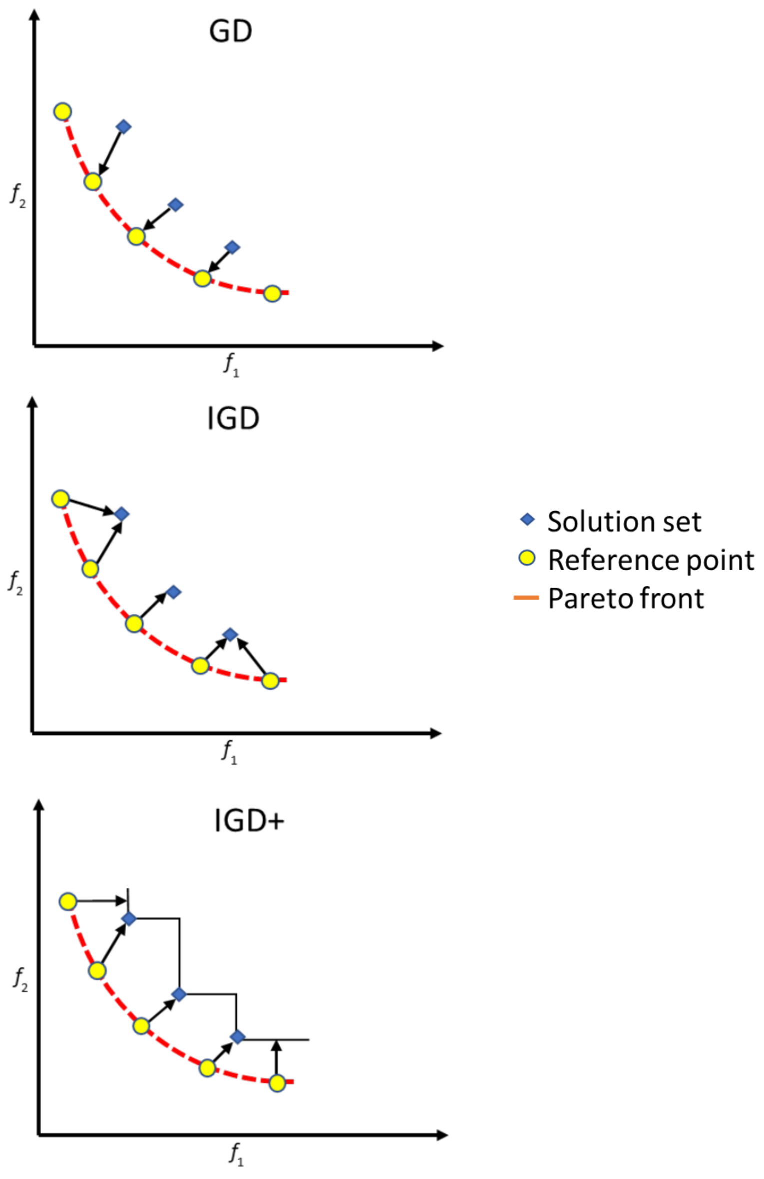

Figure 3: GD, IGD and IGD^+ calculation for an approximation set.

third graph, and (iv) a not well spread and not well distributed set in the right-hand graph. Lower values for spread and generalized spread metrics indicate better distributed and spread vectors.

2.3.3 Modified inverted generational distance

The *generational distance* metric (GD), proposed by Veldhuizen in 1998 [VL98], averages the distance from each objective vector in the approximation set to the nearest vector in the known PF (see the left-hand graph in Figure 3). Conversely, the *inverted generational distance* (IGD) [BT03, CS04] calculates the average distance from each vector in the reference or known PF to its nearest point in the approximation front (see the central graph in Figure 3). The modified inverted generational distance metric (IGD^+) was introduced by Ishibuchi *et al.* in 2015 [IMTN15] (see the right-hand graph in Figure 3). These three metrics are formulated in a similar way but provide different results:

$$GD(A,Z) = \frac{1}{|A|} \sum_{\bar{a} \in A} \min_{\bar{z} \in Z} \hat{d}(\bar{a}, \bar{z}) \tag{11}$$

$$IGD(A,Z) = \frac{1}{|Z|} \sum_{\bar{z} \in Z} \min_{\bar{a} \in A} \hat{d}(\bar{a}, \bar{z}) \tag{12}$$

$$IGD^{+}(A,Z) = \frac{1}{|Z|} \sum_{\bar{z} \in Z} \min_{\bar{a} \in A} \hat{d}^{+}(\bar{a}, \bar{z}) \tag{13}$$

where the distance $\hat{d}$ in GD and IGD is calculated as the Euclidean distance $d(\bar{a}, \bar{z}) = \sqrt{\sum_{i=1}^{m}(a_i - z_i)^2}$ and $\hat{d}^{+}(\bar{a}, \bar{z}) = \sqrt{\sum_{i=1}^{m}(max\{a_i - z_i, 0\})^2}$ in IGD^{+}. Ishibuchi *et al.* [IMTN15] found that this latter distance calculation in IGD^{+} assures that the indicator is always better for an approximation set that is dominating another (Pareto compatibility), thus providing more accurate results than IGD in some circumstances where this property does not hold. Lower values for this metric indicate a better set of solutions in terms of both convergence and diversity [TI20].

3 From static to self-adaptive polynomial mutation in NSGA-II

In this section, we first review the NSGA-II algorithm, which is widely used to perform the benchmarking of MOEAs. Next, we focus on the mutation phase of NSGA-II and explain the regular and the novel self-adaptive PLM operators in turn.

3.1 NSGA-II

The general pseudo-code for the non-dominated sorting genetic elitist algorithm (NSGA-II) is shown in Algorithm 1, and a graphical representation of how it works is displayed in Figure 4. The main loop works as follows:

1. Parents are selected, usually with binary tournament, utilizing a comparison based on non-domination ranking and crowding distance of parents. The non-domination rank segments the individuals in different fronts: F1 is the set (or front) of non-dominated individuals in the population, F2 is the set of non-dominated individuals in the population after excluding F1, and so on. The crowding distance, which is a density estimator of solutions surrounding a particular one, is used to break the ties when solutions have the same rank.
2. Genetic operators are applied to generate an offspring population that is joined to the original one (normally SBX is utilized for performing the recombination of selected parents and PLM for mutating their descendants).
3. Each individual of this new extended population, which is commonly twice the size of the original population, is evaluated to set its non-domination rank and crowding distance.

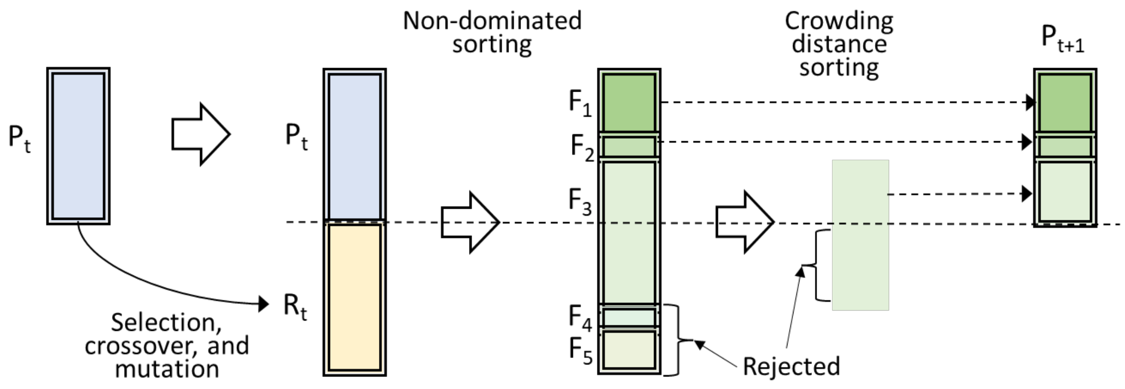

Figure 4: NSGA-II procedure, where $\{F_1, ..., F_5\}$ represent the sub-populations sorted by non-domination ranking. F_1 is the set (or front) of non-dominated individuals in the population, F_2 is the set of non-dominated individuals in the population after excluding F_1, and so on.

4. A sorting of the new extended population based on individual non-domination rank and crowding distance is performed.
5. A reduction to select the first half of the extended population is made.
6. The loop is repeated from 1 until the end condition is reached.

The use of non-domination rank and crowding distance comparisons in the selection of parents and in the ordering of the extended population assures that dominating individuals in less crowded regions are selected. Furthermore, the elitism is guaranteed by mixing the old population with the calculated offspring before performing the ordering of the extended population and selecting its best individuals. We refer the reader to the original paper [DAPM00] in order to get full details of the proposed algorithm.

3.2 Static polynomial mutation

The polynomial mutation operator over a decision vector applies a polynomial probability distribution function to generate a new decision vector starting from the current one. The pseudo-code is shown in Algorithm 2, where a perturbation following this distribution is applied with probability p_m to each variable of decision vector $\bar{x}$.

Note that the distribution function depends on the mutation distribution index η_m. Figure 5 shows an example of the impact of this parameter on the shape of the probability distribution function for a point centered at $x = 3$, where the probability of generating a closer point to the original value is larger when higher η_m values are used. Smaller values for that index tend to produce points located far from the original one.

3.3 Novel self-adaptive polynomial mutation

Following the findings of Hamdan [Ham12] mentioned in Section 2.2, we have verified that using different values for η_m has actually a direct impact on the performance

Algorithm 1 NSGA-II with traditional PLM

```
Input: $populationSize$ = size of evolving population
       $offspringSize$ = size of descendant population
       $p_c$ = crossover probability
       $p_m$ = mutation probability
       $\eta_m$ = static mutation distribution index
 1: $pPopu \leftarrow$ initializePopulation($populationSize$)
 2: $pPopu \leftarrow$ evaluateRankAndCrowdingDistance($pPopu$)
 3: while !$endCondition$ do
 4:     $qPopu \leftarrow [\ ]$
 5:     while sizeof($qPopu$) $<$ $offspringSize$ do
 6:         $parents \leftarrow$ selectParents($pPopu$)
 7:         $child1, child2 \leftarrow$ crossover($p_c, parents$)
 8:         $child1 \leftarrow$ mutate($p_m, child1, \eta_m$)
 9:         $child2 \leftarrow$ mutate($p_m, child2, \eta_m$)
10:         $qPopu$.add($child1$); $qPopu$.add($child2$)
11:     end while
12:     $rPopu \leftarrow pPopu \cup qPopu$
13:     $rPopu \leftarrow$ evaluateRankAndCrowdingDistance($rPopu$)
14:     $pPopu \leftarrow$ nonDominationSortAndReduce($rPopu$)
15: end while
```

Algorithm 2 PLM mutation of decision vector $\bar{x}$

```
Input: $\bar{x}$ = decision vector, $p_m$ = mutation probability, $\eta_m$ = mutation distribution index
Output: $\bar{x}$ = mutated decision vector
 1: for each $x \in \bar{x}$ do
 2:     if $U(0,1) < p_m$ then
 3:         ▷ $x^l$ and $x^u$ are the lower and upper bounds of decision variable $x$
 4:         $\delta_1 = \frac{x-x^l}{x^u-x^l}$, $\delta_2 = \frac{x^u-x}{x^u-x^l}$
 5:         $r \leftarrow U(0,1)$
 6:         if $r \leq 0.5$ then
 7:             $\delta_q = [2r + (1-2r)(1-\delta_1)^{\eta_m+1}]^{\frac{1}{\eta_m+1}} - 1$
 8:         else
 9:             $\delta_q = 1 - [2(1-r) + 2(r-0.5)(1-\delta_2)^{\eta_m+1}]^{\frac{1}{\eta_m+1}}$
10:         end if
11:         $x_m = x_c + \delta_q(x^u - x^l)$
12:         if $x_m < x^l$ then
13:             $x_m = x^l$
14:         end if
15:         if $x_m > x^u$ then
16:             $x_m = x^u$
17:         end if
18:         $x \leftarrow x_m$
19:     end if
20: end for
```

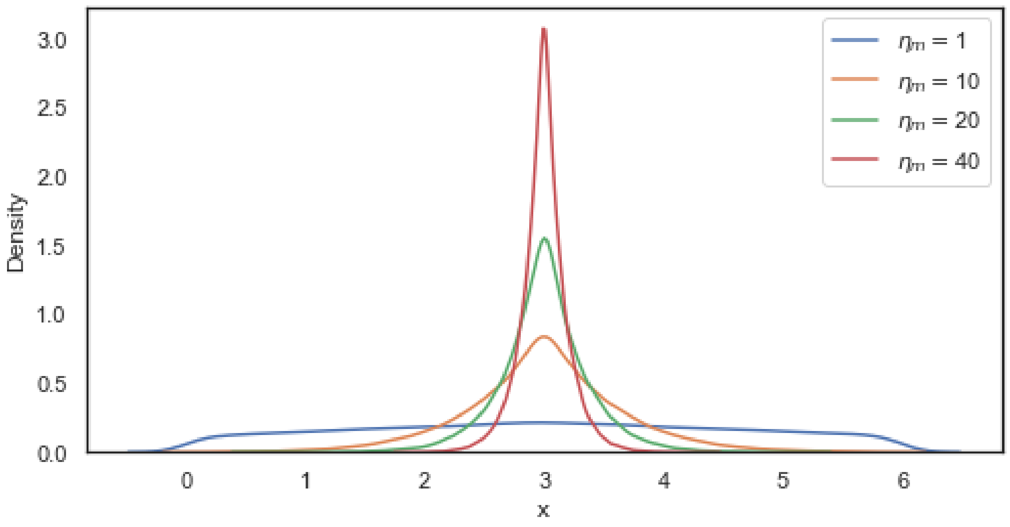

Figure 5: PLM distribution function centered at $x = 3$ for different η_m values.

of the algorithm. For example, Figure 6 shows the output of an experiment over an instance of a ZDT2 problem where the HV indicator captured and averaged over 30 runs is depicted for different values of η_m. We have observed that this behavior takes place in other problems as well.

The underlying idea of self-adaptive polynomial mutation follows the procedure utilized in evolutionary programming and evolution strategies, where the strategy parameters are stored in the genome of the population individuals and evolve with the rest of variables during the execution of the algorithm. Thus, we need to extend the representation of each individual from a decision vector $\langle x_1, x_2, ..., x_n \rangle$ to a new one including its specific mutation distribution index $\langle x_1, x_2, ..., x_n, \eta_m \rangle$. The initialization of individuals in the population (first generation) is performed by obtaining x_i and η_m values randomly chosen between their boundaries.

According to the new parameter self-adaptation strategy, we have slightly modified the original PLM algorithm by separating it in two parts: one responsible for the updating of the distribution index stored in the genome and a second one for updating the decision vector according to this mutated index. Thus, the new proposed scheme appears in Algorithm 3.

In Algorithm 3, it is important to note that the distribution index update of the selected parents is performed in line 7 before applying the mutation to the generated children in lines 9 and 10 that will use the specific η_m found in each child chromosome. The details of this updating procedure of the selected parents are included in Algorithm 4. Specifically, we average the indices of the selected parents (crossover phase) and apply a Gaussian perturbation to the averaged index (mutation phase). The resulting value is repaired before updating all the selected parents with it in order to avoid getting new η_m values outside of their bounds.

The main differences between the original proposal in Algorithm 1 and the novel one in Algorithm 3 are, on the one hand, the parameter updating mechanism applied in line 7 and, on the other hand, the use of the updated distribution index in lines

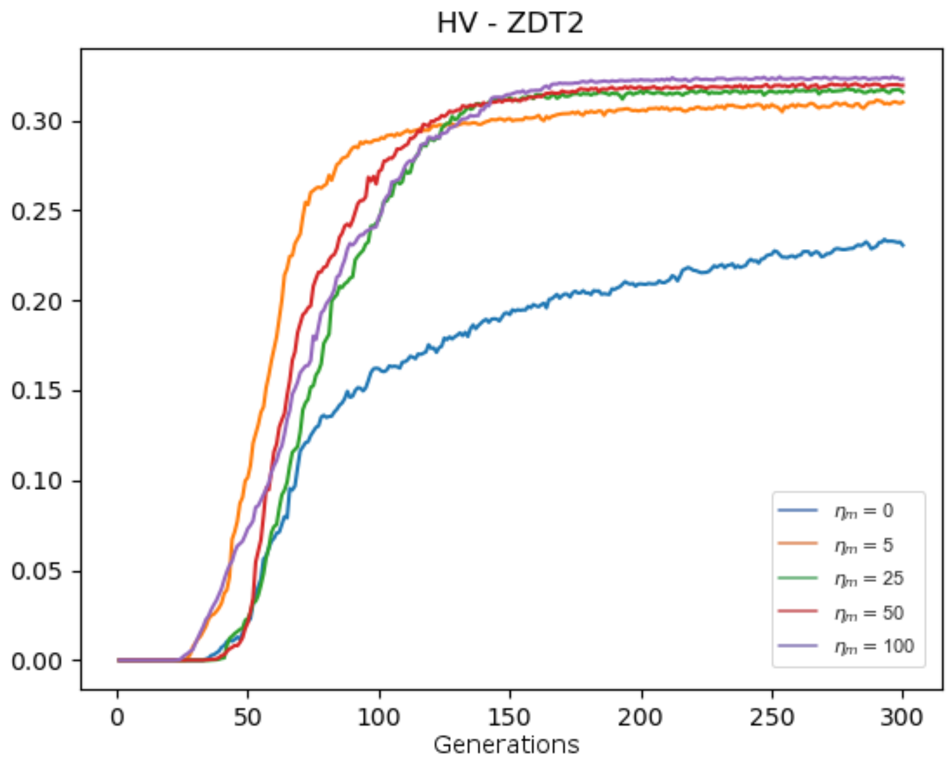

Figure 6: *HV* performance over ZDT2 problem for different η_m values.

Algorithm 3 NSGA-II with self-adaptive PLM

```
Input: $populationSize$ = size of evolving population
       $offspringSize$ = size of descendant population
       $p_c$ = crossover probability
       $p_m$ = mutation probability
       $\eta_m^l, \eta_m^u$ = lower and upper bounds for $\eta_m$
 1: $pPopu \leftarrow$ initializePopulation($populationSize, \eta_m^l, \eta_m^u$)
 2: $pPopu \leftarrow$ evaluateRankAndCrowdingDistance($pPopu$)
 3: while !$endCondition$ do
 4:     $qPopu \leftarrow$ [ ]
 5:     while sizeof($qPopu$) < $offspringSize$ do
 6:         $parents \leftarrow$ selectParents($pPopu$)
 7:         $\underline{\text{updateDistributionIndex}(parents, \eta_m^l, \eta_m^u)}$
 8:         $child1, child2 \leftarrow$ crossover($p_c, parents$)
 9:         $child1 \leftarrow$ mutate($p_m, child1, child1.\eta_m$)
10:         $child2 \leftarrow$ mutate($p_m, child2, child2.\eta_m$)
11:         $qPopu$.add($child1$); $qPopu$.add($child2$)
12:     end while
13:     $rPopu \leftarrow pPopu \cup qPopu$
14:     $rPopu \leftarrow$ evaluateRankAndCrowdingDistance($rPopu$)
15:     $pPopu \leftarrow$ nonDominationSortAndReduce($rPopu$)
16: end while
```

Algorithm 4 PLM Distribution Index update

```
Input: $parents$ = selected solutions
       $\eta_m^l, \eta_m^u$ = lower and upper bounds for $\eta_m$
Output: $parents$ = vectors with updated distribution indices
1: function updateDistributionIndex($parents, \eta_m^l, \eta_m^u$)
2:     $mutPar \leftarrow 0$
3:     for each $parent \in parents$ do
4:         $mutPar \leftarrow mutPar + parent.mutPar$
5:     end for
6:     $mutPar \leftarrow mutPar/\mathrm{sizeof}(parents)$
7:     $mutPar \leftarrow mutPar + \mathcal{N}(0,1)$
8:     $mutPar \leftarrow \mathrm{repair}(mutpar, \eta_m^l, \eta_m^u)$
9:     for each $parent \in parents$ do
10:        $parent.mutPar \leftarrow mutPar$
11:    end for
12: end function
```

9 and 10. The additional computational cost incurred when performing these two operations is linear with the number of parents due to the loops in the distribution index updating function (lines 3-5 and 9-11 of Algorithm 4).

4 Experimental evaluation

This section compares how the new self-adapted mutator performs with respect to the regular PLM used in NSGA-II in terms of efficacy by using the three selected quality indicators explained in Section 2.3. A diverse set of problems with different features is employed to evaluate the consistency of the results obtained after several independent executions.

4.1 Test problems

In order to analyze the performance of the new PLM implementation, we execute the algorithms over well-established sets of problems found in many studies in the field. We have used a significant number of problems, twenty-five, with different features: shape and continuity of the PF, number of decision variables (n) and objectives (m), and modality. This allows us to check the validity of our proposal and to determine whether it has any deficiencies depending on one or more of these characteristics.

The diverse set of functions that we have used in the experiments consists of the following families:

1. The scalable family of three-objective problem collection DTLZ 1-7 from Deb, Thiele, Laumanns, and Zitzler [DTLZ02].
2. Bi-objective unconstrained problems from the Walking Fish Group (WFG) test suite from Huband, Barone, While, and Hingston [HBWH05].
3. Bi-objective and unconstrained ZDT 1-4 and 6 problems from Zitzler, Deb and Thiele [ZDT99].

Table 1: Features of the tested problems: number of decision variables and objectives, separable (S) or non-separable (NS), uni-modal (U), multi-modal (M), deceptive multi-modal (D) and PF geometry.

Problem	n	m	**Separability**	**Modality**	**Shape**
DTLZ1	7	3	S	M	Linear, connected
DTLZ2	12	3	S	U	Convex, non-convex, connected
DTLZ3	12	3	S	M	Convex, non-convex, connected
DTLZ4	12	3	S	U	Non-convex, connected, biased
DTLZ5	12	3	S	U	Linear, degenerate
DTLZ6	12	3	S	U	Linear, degenerate
DTLZ7	22	3	S	M	Disconnected
WFG1	6	2	S	U	Convex, non-convex, connected
WFG2	6	2	NS	U	Convex, disconnected
WFG3	6	2	NS	U	Linear, degenerate
WFG4	6	2	S	M	Non-convex, connected
WFG5	6	2	S	D	Non-convex, connected
WFG6	6	2	NS	U	Non-convex, connected
WFG7	6	2	S	U	Non-convex, connected, biased
WFG8	6	2	NS	U	Non-convex, connected, biased
WFG9	6	2	NS	D	Non-convex, connected, biased
ZDT1	30	2	S	U	Convex, connected
ZDT2	30	2	S	U	Non-convex, connected
ZDT3	30	2	S	M	Disconnected
ZDT4	10	2	S	M	Convex, connected
ZDT6	10	2	S	M	Non-convex, connected
Kursawe	2	3	NS	M	Disconnected, degenerate, convex, non-convex
Schaffer	2	2	NS	U	Disconnected, convex, non-convex
Srinivas	2	2	NS	U	Convex, connected
Tanaka	2	2	NS	U	Disconnected, convex, non-convex

4. Other bi-objective and unconstrained problems like Kursawe [Kur99], Schaffer [Sch85], Srinivas [SD94], and Tanaka [TWFT95].

Table 1 shows the main features of the twenty-five selected problems. For each problem, we include the number of decision variables and objectives, whether the objective functions are separable or not, and its modality (uni-modal, multi-modal, or deceptive[1]). We also include the PF shape or geometry indicating the cases in which the PF is convex, non-convex or linear, whether the PF is connected or disconnected, and whether it has degenerated parts (PF with a dimension smaller than $m - 1$).

4.2 Algorithm execution

We have selected **jMetal**[2], a Java framework for developing multi-objective optimization algorithms created by Durillo and Nebro at the Universidad de Málaga [DN11], as the environment to perform our experimentation. This framework provides a large library of tested algorithms and facilitates the development of new ones. The hardware platform used to run the experiments was an Intel Core i5 with 8GB of RAM running Windows 11 operating system.

[1]A deceptive search space is characterized by the fact that most of it tends to guide the search towards areas which are far from the global optimum, thus leading to a suboptimal local optimum.

[2]`https://jmetal.github.io/jMetal/`

The baseline NSGA-II was parameterized using the default values suggested by jMetal. These default values are the most commonly found in the relevant literature. Specifically, we used the following parameter values:

- Population size was set to 300 individuals.
- The maximum number of evaluations was set to 25000.
- SBX was selected as the crossover operator.
- Probability $p_c = 0.9$.
- Crossover distribution index, $\eta_c = 20$.
- Regular PLM was used as the mutation operator.
- Mutation probability, $p_m = 1/n$.
- Static mutation distribution index, $\eta_m = 20$.

As far as the modified NSGA-II using the new mutation operator is concerned, it was configured exactly as the regular NSGA-II with the traditional PLM operator but, on this occasion, with a self-adaptive distribution index for the mutation process. The lower and upper boundaries for η_m were set to 1 and 100 respectively.

4.3 Results

Tables 2 through 4 include the experimental results for each quality indicator (HV, $IGD+$, and Δ^*) when both algorithms are applied to the twenty-five problems in Table 1. A summary of these results is displayed in Table 5.

As this work deals with stochastic algorithms, we average the captured quality indicator values over 30 independent executions for each problem instance and present them with their mean and variance. Figure 7 illustrates the statistical test methodology followed in order to guarantee that the obtained results are not due to randomness. We start by applying a Shapiro normality test to the data coming from each algorithm. With a negative answer (they do not follow a Gaussian distribution) a Wilcoxon test is then applied to see their means similarity. Otherwise, having assured a normal distribution for the data, a Levene test is executed to check the homocedasticity of the series; in the negative case where they do not have equal variances, we end by running a Welch test to check the similarity of their means. If, on the contrary, they have similar variances, we apply a paired Student t-test in order to see if they have similar means. We always consider a level of confidence of 99% in the statistical tests used in this work. Thus, with a significance level of 1% or p-value under 0.01, we are able to reject the null hypothesis of both algorithms performing similarly.

As mentioned earlier in this section, Tables 2-4 incorporate the mean and variance calculated over 30 independent executions of both algorithms. In the "Test result" column of each table, a "+" sign indicates that the new PLM variation outperforms the regular PLM with enough statistical confidence (as specified in the statistical test methodology described earlier in this section); conversely, a "−" sign is utilized when the regular PLM gives better results than the new PLM proposal. Finally, the "=" sign indicates that no difference exists in the performance of both PLM implementations.

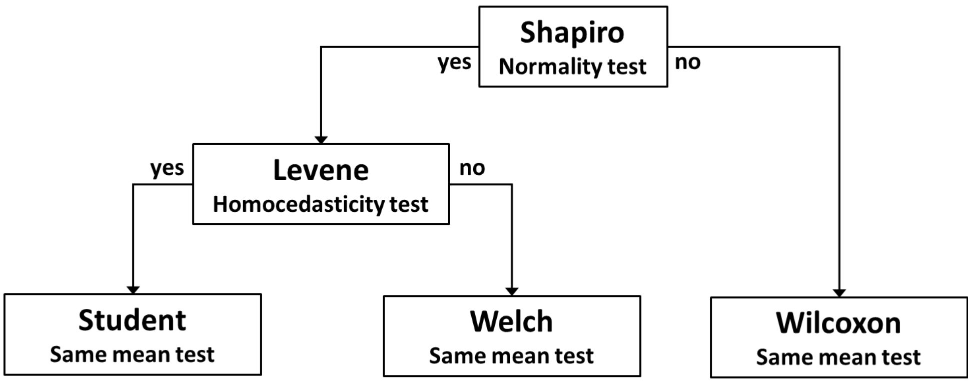

Figure 7: Statistical test methodology.

4.3.1 Hypervolume results

Table 2 collects the HV indicator results for the experiments. Notice that the self-adaptive PLM performs better than the regular operator in 19 problems and gives a similar performance regarding this indicator in other 4 problems. Worse results are observed only for 2 problems, WFG5 and WFG8. We also observe an important variance improvement in all the positive cases.

It is worth noticing the HV results obtained for DTLZ3 and DTLZ6 problems, where we indicate a zero HV mean and variance for the traditional PLM run. A zero HV is obtained when no solution is found inside the hypercube delimited by the reference point for any of the algorithm executions (in a minimization problem). However, the self-adaptive PLM is able to produce acceptable results in the same number of runs.

4.3.2 IGD^+ results

The data gathered for the IGD^+ indicator are presented in Table 3. We observe again that the self-adapted mutation operator performed equal to or better than the regular PLM operator in almost every problem (24 out of 25). One more time, WFG8 seems to be specially difficult to the new algorithm regarding this quality indicator. Both algorithms produce similar results for WFG4, WFG7, WFG9, and Srinivas problems.

We observe again similar variance improvements in the new algorithm, where we get a variance reduction in all the positive instances. Notice that the performance of the new algorithm for several of the positive cases is very remarkable (see DTLZ3, DTLZ6, and Schaffer problems).

4.3.3 Δ^* results

As reported in Table 4, when dealing with the spread and diversity of the generated solutions, our new operator does not give as impressive results as those obtained for the other two metrics. It performs better than the regular operator on 13 occasions, similarly in 11 problems, and worse in 1 instance. Once again, WFG5 seems to be a

Table 2: Hypervolume results obtained in the experiments (significance level $\alpha = 0.01$).

Problem	**NSGA-II + PLM mean(variance)**	**NSGA-II + SA-PLM mean(variance)**	**Test result**	**Statistical test**	**p-value**
DTLZ1	7.59439e-01(3.41000e-02)	7.89361e-01(4.53831e-03)	+	Wilcoxon	1.73e-06
DTLZ2	4.13529e-01(2.86745e-03)	4.17602e-01(2.15274e-03)	+	Student	5.86e-08
DTLZ3	0.00000e+00(0.00000e+00)	1.96749e-01(1.21227e-01)	+	Wilcoxon	3.35e-03
DTLZ4	4.13958e-01(2.17084e-03)	4.15810e-01(2.17415e-03)	+	Student	1.64e-03
DTLZ5	9.51122e-02(8.74617e-05)	9.52667e-02(4.95633e-05)	+	Student	1.23e-11
DTLZ6	0.00000e+00(0.00000e+00)	9.62387e-02(4.28614e-05)	+	Wilcoxon	1.73e-06
DTLZ7	3.00091e-01(1.86355e-03)	3.12930e-01(1.26271e-03)	+	Student	5.62e-38
WFG1	4.97375e-01(6.43569e-02)	6.28866e-01(2.16320e-02)	+	Wilcoxon	1.73e-06
WFG2	5.63116e-01(1.48591e-03)	5.64961e-01(1.25583e-04)	+	Welch	1.81e-07
WFG3	4.96341e-01(8.75053e-04)	4.96995e-01(3.08140e-04)	+	Welch	4.50e-04
WFG4	2.21115e-01(1.85144e-04)	2.21198e-01(1.60946e-04)	=	Student	6.98e-02
WFG5	2.01125e-01(3.98137e-03)	1.98224e-01(6.72931e-05)	−	Wilcoxon	1.24e-05
WFG6	2.03611e-01(6.92356e-03)	2.07126e-01(6.81412e-03)	+	Wilcoxon	9.27e-03
WFG7	2.12706e-01(1.69775e-04)	2.12740e-01(7.24156e-05)	=	Wilcoxon	7.50e-01
WFG8	1.70780e-01(1.54466e-02)	1.55995e-01(1.56837e-02)	−	Wilcoxon	5.67e-03
WFG9	2.42450e-01(8.73721e-04)	2.42810e-01(8.09038e-04)	=	Wilcoxon	5.45e-02
ZDT1	6.53995e-01(1.07650e-03)	6.63513e-01(1.72262e-04)	+	Welch	3.10e-30
ZDT2	3.13921e-01(2.64180e-03)	3.29930e-01(5.96829e-04)	+	Wilcoxon	1.73e-06
ZDT3	5.07723e-01(1.17048e-03)	5.15760e-01(1.48757e-04)	+	Welch	1.17e-26
ZDT4	2.30579e-01(1.55613e-01)	5.55161e-01(1.11938e-01)	+	Wilcoxon	2.13e-06
ZDT6	1.57056e-01(2.41919e-02)	4.01091e-01(6.71092e-04)	+	Welch	5.69e-31
Kursawe	4.03120e-01(1.26516e-04)	4.03276e-01(5.75611e-05)	+	Welch	2.66e-07
Schaffer	3.74026e-01(2.18289e-01)	8.13777e-01(1.79589e-02)	+	Wilcoxon	1.73e-06
Srinivas	5.43385e-01(7.44869e-05)	5.43396e-01(6.32421e-05)	=	Student	5.14e-01
Tanaka	3.08598e-01(5.32528e-04)	3.10192e-01(2.82957e-04)	+	Welch	2.27e-18

Table 3: IGD^+ results obtained in the experiments (significance level $\alpha = 0.01$).

Problem	**NSGA-II + PLM IGD^+ mean(variance)**	**NSGA-II + SA-PLM IGD^+ mean(variance)**	**Test result**	**Statistical test**	**p-value**
DTLZ1	4.29294e-02(1.79080e-02)	2.52563e-02(3.46030e-03)	+	Wilcoxon	1.73e-06
DTLZ2	2.24472e-02(7.49704e-04)	2.03312e-02(6.22110e-04)	+	Student	3.38e-17
DTLZ3	6.31840e+00(2.14560e+00)	9.29473e-01(7.80514e-01)	+	Wilcoxon	1.73e-06
DTLZ4	1.84729e-02(1.87342e-03)	1.72746e-02(1.38996e-03)	+	Wilcoxon	8.22e-03
DTLZ5	1.29969e-03(9.20083e-05)	1.18259e-03(5.07067e-05)	+	Welch	2.16e-07
DTLZ6	1.45038e+00(9.45727e-02)	1.02208e-03(7.05488e-05)	+	Welch	3.56e-36
DTLZ7	2.28455e-02(8.48368e-04)	1.65048e-02(7.61342e-04)	+	Student	2.21e-37
WFG1	1.62578e-01(8.81688e-02)	4.24815e-03(1.44789e-02)	+	Wilcoxon	1.73e-06
WFG2	1.79572e-03(1.00515e-03)	5.68657e-04(8.42110e-05)	+	Welch	2.45e-07
WFG3	1.87519e-03(5.16401e-04)	1.52509e-03(1.81825e-04)	+	Welch	1.25e-03
WFG4	1.11489e-03(9.91284e-05)	1.09393e-03(9.13196e-05)	=	Student	3.98e-01
WFG5	2.44724e-02(3.20312e-03)	2.67584e-02(2.41543e-05)	=	Wilcoxon	2.43e-02
WFG6	7.32971e-03(5.32402e-03)	4.80630e-03(5.13637e-03)	+	Wilcoxon	7.73e-03
WFG7	1.16441e-03(1.15305e-04)	1.13743e-03(4.20012e-05)	=	Welch	2.36e-01
WFG8	2.74424e-02(1.08720e-02)	3.97090e-02(1.06094e-02)	−	Wilcoxon	2.11e-03
WFG9	2.05996e-03(5.22650e-04)	1.85940e-03(4.72033e-04)	=	Wilcoxon	5.98e-02
ZDT1	7.87085e-03(7.11770e-04)	1.68842e-03(9.80272e-05)	+	Welch	9.12e-30
ZDT2	1.23470e-02(1.83534e-03)	1.70104e-03(2.21003e-04)	+	Wilcoxon	1.73e-06
ZDT3	4.60996e-03(5.12205e-04)	9.99848e-04(5.51605e-05)	+	Welch	7.53e-27
ZDT4	3.83799e-01(1.95947e-01)	7.92631e-02(8.47617e-02)	+	Wilcoxon	2.13e-06
ZDT6	2.15281e-01(2.97713e-02)	2.85648e-03(3.97697e-04)	+	Welch	1.21e-26
Kursawe	9.45517e-04(5.39416e-05)	8.83722e-04(3.98815e-05)	+	Student	4.77e-06
Schaffer	3.96014e+00(7.55962e+00)	9.74580e-03(8.93751e-03)	+	Wilcoxon	1.73e-06
Srinivas	1.19705e-03(4.13415e-05)	1.21465e-03(3.84070e-05)	=	Student	9.29e-02
Tanaka	1.81649e-03(1.89739e-04)	9.63356e-04(8.69346e-05)	+	Welch	1.75e-24

Table 4: Δ^* results gathered in the experiments (significance level $\alpha = 0.01$).

Problem	NSGA-II + PLM Δ^* mean(variance)	NSGA-II + SA-PLM Δ^* mean(variance)	Test result	Statistical test	p-value
DTLZ1	1.03000e+00(2.42294e-01)	7.83823e-01(2.53935e-02)	+	Welch	5.36e-06
DTLZ2	6.83813e-01(3.49377e-02)	6.85980e-01(2.57095e-02)	=	Student	7.85e-01
DTLZ3	1.18850e+00(1.41623e-01)	9.82180e-01(1.55223e-01)	+	Student	1.41e-06
DTLZ4	6.83076e-01(2.09105e-02)	6.69204e-01(2.87127e-02)	=	Student	3.66e-02
DTLZ5	5.00491e-01(7.34210e-02)	4.58849e-01(3.76746e-02)	+	Wilcoxon	7.27e-03
DTLZ6	8.06492e-01(3.08745e-02)	5.28644e-01(2.07721e-02)	+	Student	1.81e-44
DTLZ7	7.76200e-01(3.07696e-02)	7.60299e-01(2.81390e-02)	=	Student	4.11e-02
WFG1	7.78357e-01(6.96307e-02)	5.83607e-01(2.04473e-02)	+	Wilcoxon	1.73e-06
WFG2	8.47435e-01(6.38581e-03)	8.47032e-01(6.29434e-03)	=	Student	8.07e-01
WFG3	3.74635e-01(2.01767e-02)	3.73888e-01(2.03391e-02)	=	Student	8.87e-01
WFG4	3.91242e-01(1.61785e-02)	3.86159e-01(1.39361e-02)	=	Student	1.97e-01
WFG5	4.21736e-01(4.61604e-02)	4.39391e-01(1.71102e-02)	−	Wilcoxon	4.39e-03
WFG6	3.80974e-01(1.73058e-02)	3.80257e-01(1.82474e-02)	=	Student	8.77e-01
WFG7	3.85675e-01(1.29152e-02)	3.80295e-01(1.78018e-02)	=	Student	1.85e-01
WFG8	1.06876e+00(1.32272e-01)	7.92715e-01(8.13220e-02)	+	Student	8.24e-14
WFG9	4.01281e-01(1.53825e-02)	4.05325e-01(1.93257e-02)	=	Student	3.74e-01
ZDT1	3.82825e-01(4.24148e-02)	3.54789e-01(1.74185e-02)	+	Wilcoxon	2.26e-03
ZDT2	5.84850e-01(4.84450e-02)	3.70502e-01(4.29535e-02)	+	Wilcoxon	1.73e-06
ZDT3	8.06824e-01(1.59158e-02)	7.95947e-01(8.19819e-03)	+	Student	1.53e-03
ZDT4	8.08177e-01(7.25930e-02)	7.40472e-01(1.63474e-01)	=	Student	4.26e-02
ZDT6	6.88152e-01(4.29740e-02)	3.30043e-01(1.85386e-02)	+	Welch	2.56e-34
Kursawe	6.13962e-01(1.67108e-02)	5.95343e-01(1.16735e-02)	+	Student	5.56e-06
Schaffer	7.59430e-01(3.01218e-01)	6.77871e-01(1.48243e-01)	=	Wilcoxon	1.53e-01
Srinivas	4.50891e-01(2.49322e-02)	4.06597e-01(1.94923e-02)	+	Student	2.23e-10
Tanaka	1.21786e+00(3.63926e-02)	9.54866e-01(4.35326e-02)	+	Student	4.20e-33

hard problem for our new algorithm; however, to our surprise, it surpassed regular PLM in the WFG8 problem.

The variance analysis produces similar conclusions. For all the problems where the new algorithm outperforms the traditional one, an important reduction of the variance is again measured.

4.3.4 Tests summary

The test results for each metric are compiled in Table 5, where we additionally highlight the table cells with a dark grayed background when our algorithm gives better results for a problem (and with a light grayed background when there is no difference in performance between them). A white background means that our algorithm is performing worse for that quality indicator.

It is easy to see that HV and IGD^+ are suggesting a similar behavior of the new algorithm. It surpasses the traditional one in 19 problems and provides similar results for other 4 or 5 problems. There is only a divergence in WFG5 problem, where HV indicates that the self-adaptive PLM is doing worse than the regular operator. Anyway, it seems to be consistent with the negative Δ^*, as we know that HV also captures the diversity of solutions.

Additionally, Δ^* results are not as good as those provided by the other indicators, but they are still quite encouraging. The new algorithm provides better results on 13 occasions, and in 11 cases it shows similar performance. Only in one problem,

Table 5: Tests summary for each problem and indicator.

Problem	HV	IGD^+	Δ^*
DTLZ1	+	+	+
DTLZ2	+	+	=
DTLZ3	+	+	+
DTLZ4	+	+	=
DTLZ5	+	+	+
DTLZ6	+	+	+
DTLZ7	+	+	=
WFG1	+	+	+
WFG2	+	+	=
WFG3	+	+	=
WFG4	=	=	=
WFG5	−	=	−
WFG6	+	+	=
WFG7	=	=	=
WFG8	−	−	+
WFG9	=	=	=
ZDT1	+	+	+
ZDT2	+	+	+
ZDT3	+	+	+
ZDT4	+	+	=
ZDT6	+	+	+
Kursawe	+	+	+
Schaffer	+	+	=
Srinivas	=	=	+
Tanaka	+	+	+
$+/=/-$	19/4/2	19/5/1	13/11/1

WFG5, the new operator provides poorer results. Reviewing the characteristics of that problem, we could point to its deceptive characteristic. Probably, it is easier for the new operator to fall in the deception front and not in the real PF.

5 Conclusions and future work

NSGA-II with a fine-tuned statically parameterized polynomial mutator is typically used as a baseline when comparing the performance of MOEAs. In this work, a novel and computationally lightweight polynomial mutator that self-adapts the distribution index η_m has been proposed. This new dynamic operator has been experimentally compared to its static counterpart over 25 problems coming from different well established test suites (DTLZ, WFG, ZDT, and other key problems) covering diverse features of typical multi-objective problems. Furthermore, its performance has been measured over three major quality indicators: hypervolume, generalized spread, and modified inverted generational distance. The experiments provide a clear indication that this proposed variation improves the efficacy of NSGA-II using the traditional PLM on most of the tested problems (it only seems to have difficulties with some highly deceptive problems like WFG5 and WFG8) and for all the three indicators.

This work also emphasizes an algorithm comparison methodology whose main characteristics are: (i) a large number of problems with different PS and PF features, (ii) the use of several quality indicators to measure the algorithm performance, and (iii) a strict statistical test procedure to analyze the output data.

Additionally, this paper remarks the importance of parameter setting in EAs and specially in MOEAs. It has been experimentally demonstrated that dynamically adapted parameters can improve the algorithm efficacy without implying a significant increase of computational cost. Simple dynamic methods, like the proposed self-adapted operator, can improve the performance of an MOEA and, potentially, alter the benchmarking results obtained using statically configured baselines (as observed using NSGA-II with regular PLM).

One direction of future work is studying how the algorithm could be improved when dealing with highly deceptive problems. Another future research direction is to verify if the results of the new self-adapted operator are maintained in many-objective scenarios. Finally, understanding why the algorithm makes only moderate performance improvement regarding the spread and diversity of the obtained solutions is another interesting research line.

6 Acknowledgments

We sincerely thank Prof. Mohammad M. Hamdan from the Yarmouk University in Jordan for the valuable comments regarding his articles about the distribution index used in polynomial mutation.

I am specially grateful to Dr. Severino F. Galán from UNED for all the guidance and support to develop this work. Thanks a lot for being always there.

References

[ABBZ09] Anne Auger, Johannes Bader, Dimo Brockhoff, and Eckart Zitzler. Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point. *Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms - FOGA '09*, 2009.

[ABC+20] Charles Audet, Jean Bigeon, Dominique Cartier, Sébastien Le Digabel, and Ludovic Salomon. Performance indicators in multiobjective optimization. *European Journal of Operational Research*, 292(2):397–422, 2020.

[AM16] Aldeida Aleti and Irene Moser. A Systematic Literature Review of Adaptive Parameter Control Methods for Evolutionary Algorithms. *ACM Computing Surveys (CSUR)*, 49(3):1–35, 2016.

[Ang95] Peter J. Angeline. Adaptive and Self-Adaptive Evolutionary Computations. In Marimuthu Palaniswami and Yianni Attikiouzel, editors, *Computational Intelligence: A Dynamic Systems Perspective*, pages 152—163. IEEE Press, 1995.

[ASS+19] Anne Auger, Thomas Stützle, Mudita Sharma, Alexandros Komninos, Manuel López-Ibáñez, and Dimitar Kazakov. Deep reinforcement learning based parameter control in differential evolution. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 709–717, 2019.

[Bä92] Thomas Bäck. The Interaction of Mutation Rate, Selection, and Self-Adaptation Within a Genetic Algorithm. In *Parallel Problem Solving from Nature 2, PPSN-II, Brussels, Belgium, September 28-30, 1992*. Elsevier, 1992.

[BCSP17] Peter A N Bosman, Andres Felipe Cruz-Salinas, and Jonatan Gomez Perdomo. Self-adaptation of genetic operators through genetic programming techniques. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 913–920. Association for Computing Machinery, 2017.

[BS96] Thomas Bäck and Martin Schütz. Intelligent mutation rate control in canonical genetic algorithms. In *Lecture Notes in Computer Science*, volume 1079, pages 158–167. Springer, 1996.

[BT03] Peter A. N. Bosman and Dirk Thierens. The Balance Between Proximity and Diversity in Multiobjective Evolutionary Algorithms. In *IEEE Transactions on Evolutionary Computation*, volume 7, no. 2, pages 174–188, 2003.

[BYBS10] Mauro Birattari, Zhi Yuan, Prasanna Balaprakash, and Thomas Stützle. F-race and iterated f-race: An overview. In *Experimental Methods for the Analysis of Optimization Algorithms*, pages 311–336, Berlin, Heidelberg, 2010. Springer.

[BZ11] Johannes Bader and Eckart Zitzler. HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. *Evolutionary Computation*, 19(1):45–76, 2011.

[CS04] Carlos A. Coello Coello and Margarita Reyes Sierra. A Study of the Parallelization of a Coevolutionary Multi-objective Evolutinoary Algorithm. In *MICAI 2004: Advances in Artificial Intelligence, Third Mexican International Conference on Artificial Intelligence*. Springer, 2004.

[DAPM00] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Mayarivan. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II. In *Parallel Problem Solving from Nature PPSN VI. PPSN 2000*, volume 1917 of *Lecture Notes in Computer Science*, pages 849–858. Springer Berlin Heidelberg, 2000.

[DD20] Benjamin Doerr and Carola Doerr. Theory of Parameter Control for Discrete Black-Box Optimization: Provable Performance Gains Through Dynamic Parameter Choices. In *Theory of Evolutionary Computation, Recent Developments in Discrete Optimization*, Natural Computing Series, pages 271–321. Springer, 2020.

[DN11] Juan J. Durillo and Antonio J. Nebro. jMetal: A Java framework for multi-objective optimization. In *Advances in Engineering Software*, volume 42, no. 10, pages 760–771. Elsevier, 2011.

[DSO07] Kalyanmoy Deb, Karthik Sindhya, and Tatsuya Okabe. Self-adaptive simulated binary crossover for real-parameter optimization. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, page 1187–1194. Association for Computing Machinery, 2007.

[DTLZ02] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Multi-Objective Optimization Test Problems. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02*, volume 1, pages 825–830. IEEE, 2002.

[EHKS07] A. E. Eiben, Mark Horvath, Wojtek Kowalczyk, and Martijn C. Schut. Reinforcement Learning for Online Control of Evolutionary Algorithms. In *Engineering Self-Organising Systems, 4th International Workshop, ESOA 2006*, pages 151–160. Springer, 2007.

[EHM99] A.E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[Gre86] John J. Grefenstette. Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):122–128, 1986.

[HAA$^+$19] Ahmad Hassanat, Khalid Almohammadi, Esra'a Alkafaween, Eman Abunawas, Awni Hammouri, and V B Surya Prasath. Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach. *Information*, 10(12):390, 2019.

[Ham12] Mohammad M. Hamdan. The Distribution Index in Polynomial Mutation for Evolutionary Multiobjective Optimisation Algorithms: An Experimental Study. *Proceedings of International Conference on Electronics Computer Technology*, 2012.

[Ham14] Mohammad M. Hamdan. Revisiting the Distribution Index in Simulated Binary Crossover Operator for Evolutionary Multiobjective Optimisation Algorithms. *2014 Fourth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, pages 37–41, 2014.

[HBWH05] Simon Huband, Luigi Barone, Lyndon While, and Phil Hingston. A Scalable Multi-objective Test Problem Toolkit. In *Evolutionary Multi-Criterion Optimization, Third International Conference, EMO 2005*, volume 3410 of *Lecture Notes in Computer Science*, pages 280–295. Springer, 2005.

[HBY22] Changwu Huang, Hao Bai, and Xin Yao. Online algorithm configuration for differential evolution algorithm. *Applied Intelligence*, pages 1–19, 2022.

[HJ98] Michael Pilegaard Hansen and Andrzej Jaszkiewicz. *Evaluating the quality of approximations to the non-dominated set.* Technical University of Denmark, Technical Report IMM-REP-1998-7, 1998.

[HLY20] Changwu Huang, Yuanxiang Li, and Xin Yao. A Survey of Automatic Parameter Tuning Methods for Metaheuristics. *IEEE Transactions on Evolutionary Computation*, 24(2):201–216, 2020.

[HME97] Robert Hinterding, Zbigniew Michalewicz, and Agoston E. Eiben. Adaptation in Evolutionary Computation: A Survey. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97).* IEEE, 1997.

[IISN18] Hisao Ishibuchi, Ryo Imada, Yu Setoguchi, and Yusuke Nojima. How to Specify a Reference Point in Hypervolume Calculation for Fair Performance Comparison. *Evolutionary Computation*, 26(3):411–440, 2018.

[IMTN15] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima. Modified distance calculation in generational dis tance and inverted generational distance. In *Proc. of 8th International Conference on Evolutionary Multi-Criterion Optimization*, pages 110–125. Springer, 2015.

[IYM03] Hisao Ishibuchi, Tadashi Yoshida, and Tadahiko Murata. Balance Between Genetic Search and LocalSearch in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223, 2003.

[Jon75] Kenneth A. De Jong. *Analysis of the beavior of a class of genetic adaptive systems.* PhD thesis, Computer and Communication Sciences Department, University of Michigan, 1975.

[JOZF14] Siwei Jiang, Yew-Soon Ong, Jie Zhang, and Liang Feng. Consistencies and Contradictions of Performance Metrics in Multiobjective Optimization. *IEEE Transactions on Cybernetics*, 44(12):2391–2404, 2014.

[KC02] Joshua Knowles and David Corne. On Metrics for Comparing Nondominated Sets. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02*, volume 1, pages 711–716. IEEE, 2002.

[KHE15] Giorgos Karafotias, Mark Hoogendoorn, and A. E. Eiben. Parameter Control in Evolutionary Algorithms: Trends and Challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187, 2015.

[Kno02] Joshua D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization.* PhD thesis, Department of Computer Science, University of Reading, 2002.

[Kur99] Frank Kursawe. A Variant of Evolution Strategies for Vector Optimization. In *Parallel Problem Solving from Nature, PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 193–197. Springer, 1999.

[KYL09] Imtiaz Korejo, Shengxiang Yang, and Change Li. A Comparative Study of Adaptive Mutation Operators for Genetic Algorithms. In *The VIII Metaheuristic International Conference*, Hamburg, Germany, 2009.

[LAC16] Edgar Manoatl Lopez, Carlos A, and Coello Coello. IGD+ -EMOA: A Multi-Objective Evolutionary Algorithm Based on IGD+. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 999–1006, 2016.

[LIDLC$^+$16] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[LLJ22] Qiang Long, Guoquan Li, and Lin Jiang. A novel solver for multi-objective optimization: dynamic non-dominated sorting genetic algorithm (DNSGA). *Soft Computing*, 26(2):725–747, 2022.

[LLM07] Fernando G. Lobo, Cláudio F. Lima, and Zbigniew Michalewicz. *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computation Intelligence.* Springer, 2007.

[LPN$^+$21] Marcelo Gomes Pereira de Lacerda, Luis Filipe de Araujo Pessoa, Fernando Buarque de Lima Neto, Teresa Bernarda Ludermir, and Herbert Kuchen. A systematic literature review on general parameter control for evolutionary and swarm-based algorithms. *Swarm and Evolutionary Computation*, volume 60, 2021.

[LY04] Chang-Yong Lee and Xin Yao. Evolutionary Programming Using Mutations Based on the Lévy Probability Distribution. *IEEE Transactions on Evolutionary Computation*, 8(1):1–13, 2004.

[LY19] Miqing Li and Xin Yao. Quality Evaluation of Solution Sets in Multiobjective Optimisation: A Survey. *ACM Computing Surveys (CSUR)*, 52(2):26, 2019.

[MMPO09] Efrén Mezura-Montes and Ana Gabriela Palomeque-Ortiz. Self-adaptive and Deterministic Parameter Control in Differential Evolution for Constrained Optimization. In *Constraint-Handling in Evolutionary Optimization*, volume 198 of *Studies in Computational Intelligence*, pages 95–120, 2009.

[NLA$^+$08] Antonio J. Nebro, Francisco Luna, Enrique Alba, Bernabé Dorronsoro, Juan J. Durillo, and Andreas Beham. AbYSS: Adapting Scatter Search to Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, 12(4):439–457, 2008.

[OJS03] Tatsuya Okabe, Yaochu Jin, and Bernhard Sendhoff. A Critical Survey of Performance Indices for Multi-Objective Optimisation. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03*, volume 2, pages 878–885. IEEE, 2003.

[Pap21] Gregor Papa. Applications of dynamic parameter control in evolutionary computation. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion)*, Proceedings of the Genetic and Evolutionary Computation Conference Companion, Lille, France, 2021. ACM.

[PPSN19] Rafael Stubs Parpinelli, Guilherme Felippe Plichoski, Renan Samuel Da Silva, and Pedro Henrique Narloch. A review of techniques for online control of parameters in swarm intelligence and evolutionary computation algorithms. *International Journal of Bio-Inspired Computation*, 13(1):1, 2019.

[Rec71] Ingo Rechenberg. *Evolutionsstrategie; Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* PhD thesis, Department of Process Engineering, Technical University of Berlin, 1971.

[RGD+22] Iman Rahimi, Amir H Gandomi, Kalyanmoy Deb, Fang Chen, and Mohammad Reza Nikoo. Scheduling by NSGA-II: Review and Bibliometric Analysis. *Processes*, 10(1):98, 2022.

[RLB15] Nery Riquelme, Christian Von Lücken, and Benjamín Barán. Performance Metrics in Multi-Objective Optimization. In *2015 Latin American Computing Conference (CLEI)*. IEEE, 2015.

[RW20] Amirhossein Rajabi and Carsten Witt. Self-Adjusting Evolutionary Algorithms for Multimodal Optimization. In *Proceedings of GECCO '20*, pages 1314–1322. ACM Press, 2020.

[Sch85] J. David Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100. L. Erlbaum Associates Inc., 1985.

[SD94] N. Srinivas and Kalyanmoy Deb. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[SF96] J. Smith and T.C. Fogarty. Self adaptation of mutation rates in a steady state genetic algorithm. In *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, pages 318–323. IEEE, 1996.

[SF97] J. E. Smith and T. C. Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Computing*, 1(2):81–87, 1997.

[SP97] Rainer Storn and Kenneth Price. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[TCMG09] K.C. Tan, S.C. Chiam, A.A. Mamun, and C.K. Goh. Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. *European Journal of Operational Research*, 197(2):701–713, 2009.

[TI20] Ryoji Tanabe and Hisao Ishibuchi. An Analysis of Quality Indicators Using Approximated Optimal Distributions in a 3-D Objective Space. *IEEE Transactions on Evolutionary Computation*, 24(5):853–867, 2020.

[TWFT95] M. Tanaka, H. Watanabe, Y. Furukawa, and T. Tanino. GA-based decision support system for multicriteria optimization. In *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, volume 2, pages 1556–1561. IEEE, 1995.

[TY07] Renato Tinós and Shengxiang Yang. Self-Adaptation of Mutation Distribution in Evolutionary Algorithms. In *2007 IEEE Congress on Evolutionary Computation*, pages 79–86. IEEE, 2007.

[VL98] David A. Van Veldhuizen and Gary B. Lamont. Evolutionary Computation and Convergence to a Pareto Front. In *Late-breaking papers book at the Genetic Programming 1998 Conference (GP-98)*, pages 221–228. Stanford University Bookstore, 1998.

[WAY+16] Shuai Wang, Shaukat Ali, Tao Yue, Yan Li, and Marius Liaaen. A Practical Guide to Select Quality Indicators for Assessing Pareto-Based Search Algorithms in Search Based Software Engineering. In *IEEE/ACM 38th IEEE International Conference on Software Engineering*. IEEE, 2016.

[YU06] Shengxiang Yang and Şima Uyar. Adaptive mutation with fitness and allele distribution correlation for genetic algorithms. In *Proceedings of the 2006 ACM symposium on Applied computing - SAC '06*, pages 940–944. ACM, 2006.

[ZCZ+12] Jun Zhang, Wei-Neng Chen, Zhi-Hui Zhan, Wei-Jie Yu, Yuan-Long Li, Ni Chen, and Qi Zhou. A survey on algorithm adaptation in evolutionary computation. *Frontiers of Electrical and Electronic Engineering*, 7(1):16–31, 2012.

[ZDT99] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 1999.

[ZJZ+06] Aimin Zhou, Yaochu Jin, Qingfu Zhang, Bernhard Sendhoff, and Edward Tsang. Combining Model-Based and Genetics-based Offspring Generation for Multi-Objective Optimization Using a Convergence Criterion. In *2006 IEEE International Conference on Evolutionary Computation*, pages 892–899. IEEE, 2006.

[ZLD+10] Fanchao Zeng, Malcolm Yoke Hean Low, James Decraene, Suiping Zhou, and Wentong Cai. Self-Adaptive Mechanism for Multi-objective Evolutionary Algorithms. In *Proceedings of the International MultiConference of Engineers and Computer Scientists 2010*, volume I, pages 7–12, 2010.

[ZLZL19] Zhibiao Zhao, Bin Liu, Chunran Zhang, and Haoran Liu. An improved adaptive NSGA-II with multi-population algorithm. *Applied Intelligence*, 49(2):569–580, 2019.

[ZT99] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[ZTL+03] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(13):117–132, 2003.