Soft Computing (2023) 27:17711–17727
https://doi.org/10.1007/s00500-023-09049-0

OPTIMIZATION

# Self-adaptive polynomial mutation in NSGA-II

Jose L. Carles-Bou[1] · Severino F. Galán[2]

Accepted: 20 July 2023 / Published online: 21 August 2023

**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 and implemented by the scientific community, the popular Non-Dominated Sorting Genetic Algorithm (NSGA-II) remains as a widely used baseline for algorithm performance comparison purposes and applied to different engineering problems. Since every evolutionary algorithm needs several parameters to be set up in order to operate, parameter control constitutes a crucial task for obtaining an effective and efficient performance in its execution. 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 show that the proposed modified NSGA-II with a self-adaptive mutator outperforms its static counterpart in more than 75% of the problems using three quality metrics (hypervolume, generalized spread, and modified inverted generational distance).

**Keywords** Multi-objective evolutionary algorithm · NSGA-II · Polynomial mutation · Distribution index self-adaptation

## 1 Introduction

Optimization problems are ubiquitous in nature (Kochenderfer and Wheeler 2019; Mohamed et al. 2022; Garg 2019; Kundu and Garg 2022a, b). A high number of real-world optimization problems are multi-objective, where the objectives being optimized are usually in conflict with each other. In this kind of optimization problems, rather than getting a unique optimal solution, there are usually several of them, maybe infinite, known as the Pareto set. Between other meta-heuristic approaches, Multi-Objective Evolutionary Algorithms (MOEAs) have the ability to effectively approximate this set as it has been reported in the scientific literature for more than three decades.

The Non-Dominated Sorting Genetic Algorithm II (NSGA-II) was proposed by Deb et al. (2000) as an exploration tool that employs a genetic algorithm to solve multi-objective optimization problems (MOPs). NSGA-II is still nowadays one of the most used and cited MOEAs and it has been applied to a broad variety of optimization problems since its inception in fields like economics, engineering or logistics. The NSGA-II algorithm creates a random population, selects individuals, and applies genetic operations on them, ranking and sorting individuals based on their non-domination level, and making use of a crowding distance operation to keep the evolving population diverse while aiming to generate a set of optimal solutions.

The fine-tuning of the parameters of an evolutionary algorithm is a key aspect that directly impacts on both its efficacy and its efficiency and it is usually a hard and time-consuming manual task done specifically for the particular problem being solved. Therefore, parameter tuning and control in evolutionary algorithms has been a relevant research topic since their inception.

In the literature, when NSGA-II is used as a baseline algorithm in benchmarks or even when it is applied to solve

✉ Jose L. Carles-Bou
jcarles5@alumno.uned.es, carles@ieee.org

Severino F. Galán
seve@dia.uned.es

1 Escuela Internacional de Doctorado EIDUNED, Universidad Nacional de Educación a Distancia UNED, Madrid, Spain

2 Department of Artificial Intelligence, UNED, Madrid, Spain

some specific engineering problem, 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 (Sharma et al. 2023; Liu et al. 2023; Wang et al. 2023; Ozcelikkan et al. 2022). However, Deb et al. (2007) a method for self-adapting the SBX crossover operator in order to improve the algorithm performance. Furthermore, Hamdan demonstrated in (Hamdan 2012) that the fixed distribution index broadly utilized in PLM does not always provide the best performance results. Consequently, these two relevant works show that it is not always 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 $\eta_m$, used in the regular NSGA-II producing an increased efficacy over the results observed in its traditional counterpart. We compare the performance of this new algorithm variant to that of plain NSGA-II utilizing the static configuration widely found in the literature over a set of twenty-five benchmark problems from different test suites (DTLZ, WFG, ZDT, and other key problems), which experimentally confirm that this novel technique provides better results in almost every presented problem for several quality indicators such as hypervolume, generalized spread, and modified inverse generational distance.

The main motivation and contributions made by this paper could be summarized as follows:

- Verify that the static parameterization of the regular PLM used in NSGA-II not always present the best performance results.
- Introduce a new NSGA-II variant including a self-adaptive polynomial mutation operator which improves the observed efficacy over the regular NSGA-II with static PLM implementations. The results are measured with different metrics (hypervolume, generalized spread, and modified inverse generational distance) trying to cover different aspects of the solutions provided by the algorithms.
- Use a quite large of known benchmark problems coming from families like DTLZ, WFG and ZDT to evaluate the performance of the proposed algorithm.
- Propose a strict statistical test methodology to validate the new algorithm optimizing the selected problems.

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 Sect. 4. Finally, Sect. 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 $i$th 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 (Hansen and Jaszkiewicz 1998). Usually, the ultimate objective when solving MOPs is not to find the real PF but a *good* approximation to it (Riquelme et al. 2015). Several tools to evaluate the quality of these non-dominated sets are explained in Sect. 2.3.

Even though the terms "solution" and "objective vector" have been used interchangeably (Nebro et al. 2008; Lopez et al. 2016; Ishibuchi et al. 2018), 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 (Tanabe and Ishibuchi 2020).

## 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 (Huang et al. 2022). 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 (Storn and Price 1997; Hamdan 2012; Deb et al. 2007). 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 (Back 1992).

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 (Birattari et al. 2010; López-Ibánez et al. 2016) is implemented relieving the researcher of the tedious and time consuming task of repetitive manual parameter adjustment. Another interesting alternative is employing meta-heuristic techniques, like the Meta-GA used by Grefenstette (Grefenstette 1986), in which an external evolutionary algorithm adjusts the algorithm parameters.

Starting just when evolutionary algorithms were created (Jong 1975; Grefenstette 1986), 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 (Angeline 1995) 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 (Smith and Fogarty 1997) 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 (Hinterding et al. 1997; Eiben et al. 1999) 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 (Zhang et al. 2012; Parpinelli et al. 2019; Doerr and Doerr 2020):

- 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 (Eiben et al. 1999). 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 (Mezura-Montes and Palomeque-Ortiz 2009; Hassanat et al. 2019).
- *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 (Rechenberg 1971) can be found. Alternatively, more complex methods are based on learning principles brought from fields like reinforcement learning or deep learning (Eiben et al. 2007; Auger et al. 2019).
- *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 (Smith and Fogarty 1996; Deb et al. 2007; Bosman et al. 2017; Rajabi and Witt 2020).

Due to the vast amount of publications on parameter control in evolutionary algorithms, several exhaustive reviews have been made available to researchers and practitioners (Eiben et al. 1999; Karafotias et al. 2015; Aleti and Moser 2016), and recent reviews can be found in (Parpinelli et al. 2019), (Huang et al. 2020), (Lacerda et al. 2021) (also covering swarm inspired algorithms), and (Huang et al. 2022) (focused on differential evolution). In addition, an interesting theoretical approach is developed in (Doerr and Doerr 2020). Finally, a complete compendium on parameter control and its application is offered in (Papa 2021) and an interesting compilation about parameter setting is included in (Lobo et al. 2007).

NSGA-II, proposed by Deb et al. (2000), is a well known, competent, and extensively used genetic algorithm for solving MOPs that attains near-optimal, diverse, and uniformly distributed solution sets (Tan et al. 2009; Rahimi et al. 2022). Even though NSGA-II is a mature algorithm, it is still being utilized as a baseline to test new algorithms performance (Ishibuchi et al. 2003; Zhao et al. 2019; Long et al. 2022; Sharma et al. 2023). 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 (Aleti and Moser 2016), several authors have shown that self-adaptation can improve the applicability of MOEAs to online decision support in real-world problems (Zeng et al. 2010). Deb et al. (2007) 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, $\eta_c$, 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. (2010) 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 (Back and Schutz 1996; Yang and Uyar 2006), by changing the mutation probability distribution function like in evolution strategies and evolutionary programming (Lee and Yao 2004; Tinós and Yang 2007), or even by changing the mutation operator itself during the evolution (Korejo et al. 2009), very few studied the idea of changing the probability distribution index in PLM, $\eta_m$, as Deb did in SA-SBX for crossover. Furthermore, Hamdan (Hamdan 2012, 2014) demonstrated that different values for $\eta_m$ 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, Sect. 3 proposes a novel and simple self-adaptive mutation operator which, as shown in Sect. 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 (Zitzler et al. 2003; Riquelme et al. 2015).

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 (Okabe et al. 2003). 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 (Okabe et al. 2003). 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 approximation set are distributed (the relative distance between them) and spread (the range of values covered by these vectors) (Riquelme et al. 2015).
- *Convergence-diversity metrics* map both the proximity of the objective vectors to the PF and the diversity of these vectors to a scalar value (Jiang et al. 2014).

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. (2015) studying fifty-four indicators, Li and Yao (Li and Yao 2019) covering one hundred indicators, and Audet et al. (2020) describing fifty-seven metrics. The papers published by Hansen and Jaszkiewicz (Hansen and Jaszkiewicz 1998) and Zitzler et al. (2003) 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 (Knowles 2002; Knowles and Corne 2002; Okabe et al. 2003; Jiang et al. 2014; Wang et al. 2016) 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 (Zitzler and Thiele 1999), 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 (Knowles and Corne 2002; Bader and Zitzler 2011). Larger values of $HV$ indicate that the vectors in the approximation set are closer to the true PF and evenly distributed over it (Jiang et al. 2014).

$HV$ calculates the hypervolume that is dominated by the objective vectors in the approximation set $S$ bounded by a reference point $\bar{r}$ (see Fig. 1, where $HV$ measures the shaded green area). Although the selection of the reference point to calculate $HV$ is critical (Auger et al. 2009; Ishibuchi et al. 2018), 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:

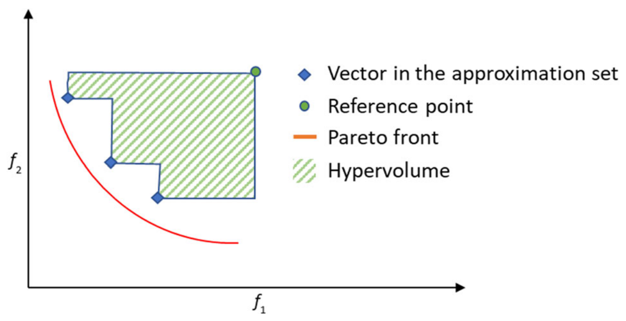

**Fig. 1** Hypervolume covering the shaded green area in a bi-objective minimization problem

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

where $\Lambda$ 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* ($\Delta$), introduced by Deb et al. (2000), 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. (2006) extended this indicator to $m$-objective problems in the *generalized spread metric* ($\Delta^*$) 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

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

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

The $\Delta$ and $\Delta^*$ metrics, unlike $HV$, require a known Pareto front. We can observe in Fig. 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 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 (Veldhuizen and Lamont 1998), 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 Fig. 3). Conversely, the *inverted generational distance* ($IGD$) (Bosman and Thierens 2003; Coello and Sierra 2004) 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 Fig. 3). The modified

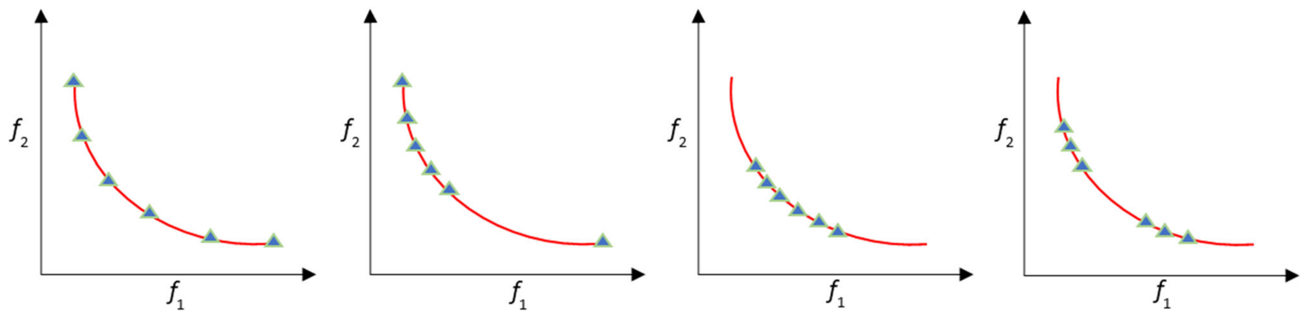

Fig. 2 Examples of spread and distribution of vectors over the Pareto front

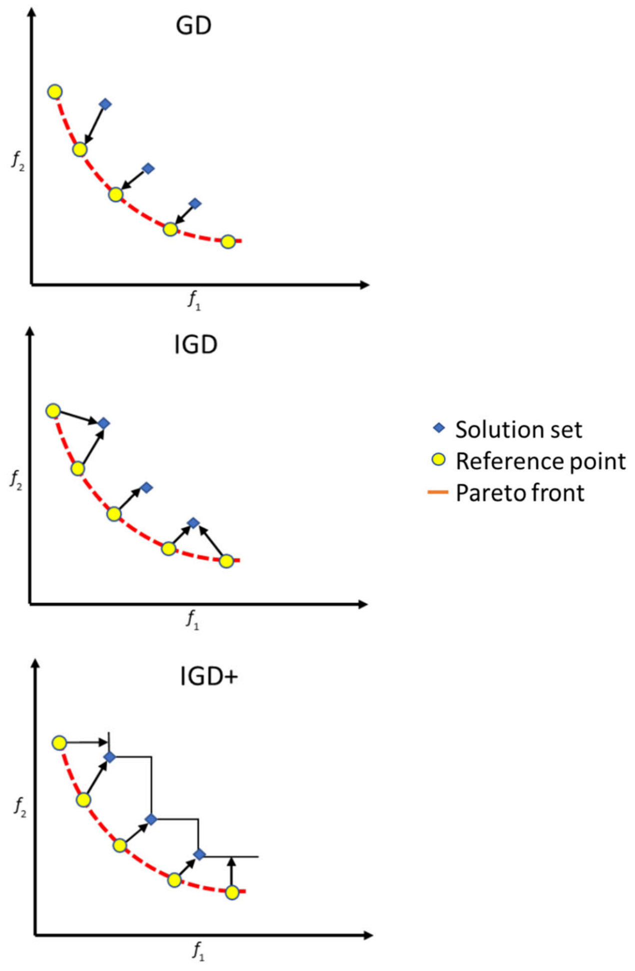

Fig. 3 $GD$, $IGD$ and $IGD^+$ calculation for an approximation set

inverted generational distance metric ($IGD^+$) was introduced by Ishibuchi et al. (2015) (see the right-hand graph in Fig. 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. (2015) 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 (Tanabe and Ishibuchi 2020).

## 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 Fig. 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.
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.

**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
```

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 (Deb et al. 2000) 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 $\bar{x}$.

**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 $r \le 0.5$ 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
```

Note that the distribution function depends on the mutation distribution index $\eta_m$. 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 $\eta_m$ 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 (Hamdan 2012) mentioned in Sect. 2.2, we have verified that using different values for $\eta_m$ has actually a direct impact on the performance of the algorithm. For example, Fig. 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 $\eta_m$. We have observed that this behavior takes place in other problems as well.

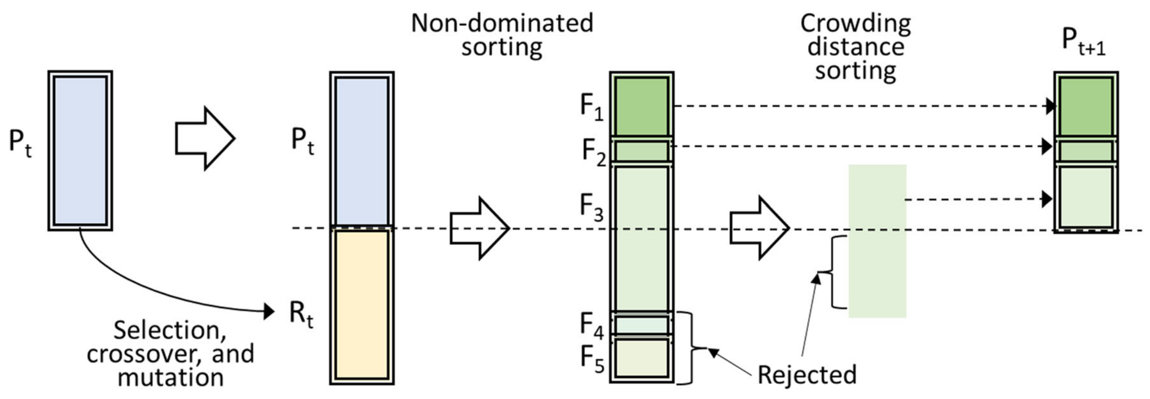

**Fig. 4** NSGA-II procedure, where {$F_1$, ..., $F_5$} represent the sub-populations sorted by non-domination ranking. $F_1$ is the set (or front) of non-dominated individuals in the population, $F_2$ is the set of non-dominated individuals in the population after excluding $F_1$, and so on

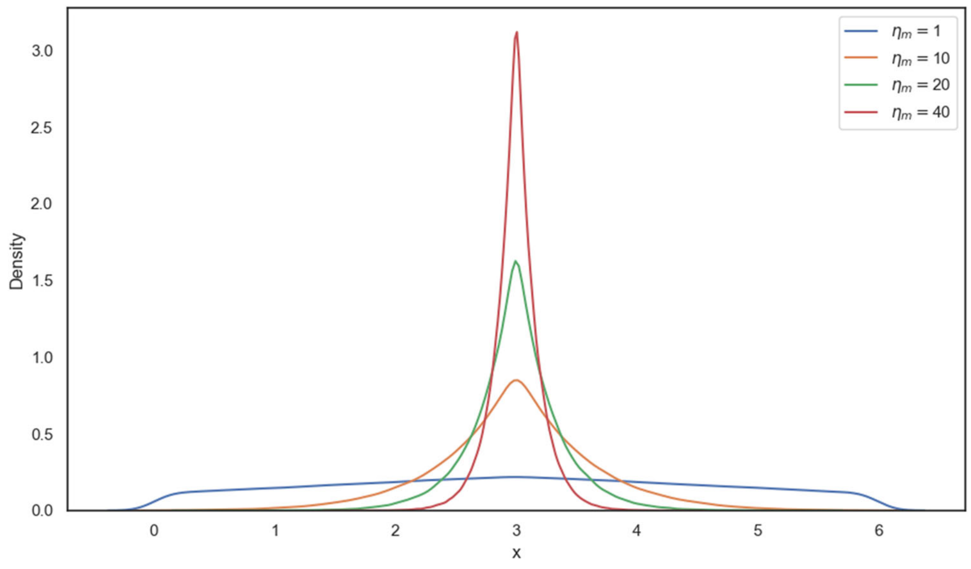

**Fig. 5** PLM distribution function centered at $x = 3$ for different $\eta_m$ values

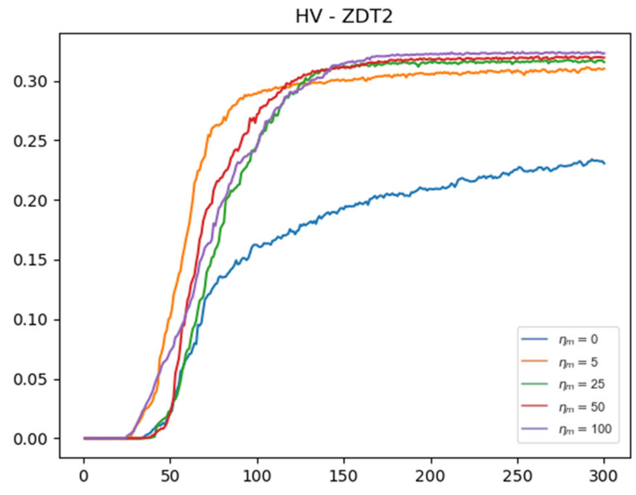

**Fig. 6** $HV$ performance over ZDT2 problem for different $\eta_m$ values

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 $\eta_m$ 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 $\eta_m$ found in each child

**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:         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
```

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 $\eta_m$ values outside of their bounds.

**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:     $mutPar \leftarrow mutPar$/sizeof($parents$)
7:     $mutPar \leftarrow mutPar + \mathcal{N}(0, 1)$
8:     $mutPar \leftarrow$ 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
```

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 of Algorithm 3, and on the other hand, the use of the updated distribution index in lines 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 Sect. 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 et al. (2002).
2. Bi-objective unconstrained problems from the Walking Fish Group (WFG) test suite from Huband et al. (2005).
3. Bi-objective and unconstrained ZDT 1-4 and 6 problems from Zitzler et al. (1999).
4. Other bi-objective and unconstrained problems like Kursawe (1991), Schaffer (1985), Srinivas (1994), and Tanaka et al. (1995).

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

[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/.

**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 |

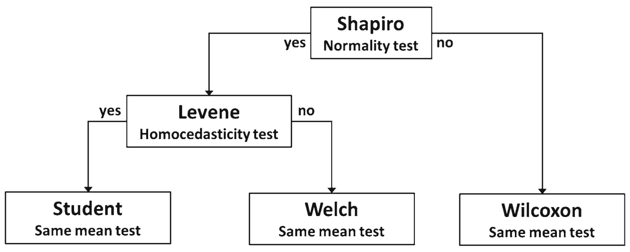

**Fig. 7** Statistical test methodology

and Nebro at the Universidad de Málaga (2011), 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.

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.
- Crossover 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 $\eta_m$ 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 $\Delta^*$) when both algorithms are applied to the twenty-five problems in Table 1. A summary of these results is displayed in Table 5.

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

| Problem | NSGA-II + PLM $HV$ mean(variance) | NSGA-II + SA-PLM $HV$ 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 |

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 homoscedasticity 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, 3 and 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.

### 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.

**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 |

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 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.

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

| Problem | NSGA-II + PLM $\Delta^*$ mean(variance) | NSGA-II + SA-PLM $\Delta^*$ 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 |

#### 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 "+" sign when our algorithm gives better results for a problem (and with a "=" sign when there is no difference in performance between them). A "-" sign 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 $\Delta^*$, as we know that $HV$ also captures the diversity of solutions.

Additionally, $\Delta^*$ 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, 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 or when it is applied to solve some MOPs. In this work, a new version of NSGA-II algorithm incorporating a novel and computationally lightweight polynomial mutator that self-adapts its distribution index $\eta_m$ has been proposed. The modified algorithm with the 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,

**Table 5** Tests summary for each problem and indicator

| Problem | $HV$ | $IGD^+$ | $\Delta^*$ |
|---|---|---|---|
| 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 |

generalized spread, and modified inverted generational distance.

The experiments provide a clear confirmation that the proposed variation improves the observed performance of the regular NSGA-II and PLM on almost all of the tested problems and for all the three covered indicators. The main advantage of this self-adaptive mutation and the modified NSGA-II are that they are really easy to implement and introduce a very small overhead in the optimization run-time.

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 the regular NSGA-II and, potentially, alter the benchmarking results obtained using statically configured parameters.

Currently, the main drawback of the new algorithm is the not specially good solution distribution measured through the Generalized Spread metric $\Delta^*$. Understanding why the algorithm makes only a moderate performance improvement regarding the spread and diversity of the obtained solutions should be a priority in future investigation. Adding other quality metrics, like Maximum Spread $S_p$ (Schott 1995; Riquelme et al. 2015) and Spacing $M_3^*$ (Zitzler et al. 1999), would be desirable to better understand these results.

One direction of future work is studying how the algorithm could be improved when dealing with highly deceptive problems. It would be also advisable to confront this new NSGA-II variant with some recent and state-of-art multi-objective algorithms like Multi-Objective Non-dominated Advanced Butterfly Optimization Algorithm (MONSBOA) described in (Sharma et al. 2023).

Finally, the possible application of this dynamic approach to new mutation strategies like the Lévy flight based found in the hybrid Lévy Slime Mould Algorithm Teaching-Learning Based Optimization (LSMA-TLBO) proposed in (Kundu and Garg 2022b) or the quantum inspired mutation found in the Quantum Mutation-based Backtracking Search Algorithm introduced in (Nama et al. 2022) should be studied.

**Author Contributions** All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Jose L. Carles-Bou. The first draft of the manuscript was written by Jose L. Carles-Bou and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

**Funding** The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

**Data Availability** The data that support the findings of this work are available from the corresponding author upon request.

## Declaration

**Conflict of interest** The authors declare that they have no conflict of interest either financial or non-financial.

**Research involving Human Participants and/or Animals** The authors declare no human or animals where involved in this research.

## References

Aleti A, Moser I (2016) A systematic literature review of adaptive parameter control methods for evolutionary algorithms. ACM Comput Surv (CSUR) 49(3):1–35. https://doi.org/10.1145/2996355

Angeline PJ (1995) Adaptive and self-adaptive evolutionary computations. In: Palaniswami M, Attikiouzel Y (eds) Computational intelligence: a dynamic systems perspective. IEEE Press, New York, pp 152–163

Audet C, Bigeon J, Cartier D, Digabel SL, Salomon L (2020) Performance indicators in multiobjective optimization. Eur J Oper Res 292(2):397–422. https://doi.org/10.1016/j.ejor.2020.11.016

Auger A, Bader J, Brockhoff D, Zitzler E (2009) Theory of the hypervolume indicator: optimal $\mu$-distributions and the choice of the reference point. In: Proceedings of the tenth ACM SIGEVO workshop on foundations of genetic algorithms - FOGA '09 https://doi.org/10.1145/1527125.1527138

Auger A, Stutzle T, Sharma M, Komninos A, López-Ibánez M, Kazakov D (2019) Deep reinforcement learning based parameter control in differential evolution. In: Proceedings of the genetic and evolutionary computation conference pp. 709–717. https://doi.org/10.1145/3321707.3321813

Back T (1992) The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In: Parallel problem solving from nature 2, PPSN-II. Elsevier, Brussels, Belgium

Back T, Schutz M (1996) Intelligent mutation rate control in canonical genetic algorithms. In: ISMIS '96: Proceedings of the 9th international symposium on foundations of intelligent systems. Springer, Berlin, Heidelberg, pp 158–167, https://doi.org/10.1007/3-540-61286-6_141

Bader J, Zitzler E (2011) HypE: an algorithm for fast hypervolume-based many-objective optimization. Evol Comput 19(1):45–76. https://doi.org/10.1162/evco_a_00009

Birattari M, Yuan Z, Balaprakash P, Stuzle T (2010) F-race and iterated f-race: an overview. In: Experimental methods for the analysis of optimization algorithms. Springer, Berlin, Heidelberg, pp 311–336, https://doi.org/10.1007/978-3-642-02538-9_13

Bosman PAN, Thierens D (2003) The balance between proximity and diversity in multiobjective evolutionary algorithms. IEEE Trans Evol Comput. https://doi.org/10.1109/tevc.2003.810761

Bosman PAN, Cruz-Salinas AF, Perdomo JG (2017) Self-adaptation of genetic operators through genetic programming techniques. In: Proceedings of the genetic and evolutionary computation conference. Association for Computing Machinery, Berlin, pp 913–920, https://doi.org/10.1145/3071178.3071214

Coello CAC, Sierra MR (2004) A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In: MICAI 2004: advances in artificial intelligence, third Mexican international conference on artificial intelligence. Springer, Mexico City, https://doi.org/10.1007/978-3-540-24694-7_71

Deb K, Agrawal S, Pratap A, Mayarivan T (2000) A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Parallel problem solving from nature PPSN VI, lecture notes in computer science, vol 1917. Springer, Berlin, pp 849–858, https://doi.org/10.1007/3-540-45356-3_83

Deb K, Thiele L, Laumanns M, Zitzler E (2002) Scalable multi-objective optimization test problems. In: Proceedings of the 2002 congress on evolutionary computation. CEC'02, vol 1. IEEE, Honolulu, pp 825–830, https://doi.org/10.1109/cec.2002.1007032

Deb K, Sindhya K, Okabe T (2007) Self-adaptive simulated binary crossover for real-parameter optimization. In: Proceedings of the 9th annual conference on genetic and evolutionary computation. Association for computing machinery, London, GECCO '07, p 1187-1194, https://doi.org/10.1145/1276958.1277190

Doerr B, Doerr C (2020) 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. Springer International Publishing, Cham, pp 271–321, https://doi.org/10.1007/978-3-030-29414-4_6

Durillo JJ, Nebro AJ (2011) jMetal: a Java framework for multi-objective optimization. In: Advances in Engineering Software, vol 42, no. 10. Elsevier, Oxford, pp 760–771, https://doi.org/10.1016/j.advengsoft.2011.05.014

Eiben A, Hinterding R, Michalewicz Z (1999) Parameter control in evolutionary algorithms. IEEE Trans Evol Comput 3(2):124–141. https://doi.org/10.1109/4235.771166

Eiben AE, Horvath M, Kowalczyk W, Schut MC (2007) Reinforcement learning for online control of evolutionary algorithms. In: Engineering self-organising systems, 4th international workshop, ESOA 2006. Springer, Hakodate, Japan, pp 151–160, https://doi.org/10.1007/978-3-540-69868-5_10

Garg H (2019) A hybrid GSA-GA algorithm for constrained optimization problems. Inf Sci 478:499–523. https://doi.org/10.1016/j.ins.2018.11.041

Grefenstette JJ (1986) Optimization of control parameters for genetic algorithms. IEEE Trans Syst Man Cybern 16(1):122–128. https://doi.org/10.1109/tsmc.1986.289288

Hamdan MM (2012) The distribution index in polynomial mutation for evolutionary multiobjective optimisation algorithms: an experimental study. In: Proceedings of international conference on electronics computer technology

Hamdan MM (2014) Revisiting the distribution index in simulated binary crossover operator for evolutionary multiobjective optimisation algorithms. In: 2014 fourth international conference on digital information and communication technology and its applications (DICTAP) pp 37–41. https://doi.org/10.1109/dictap.2014.6821653

Hansen MP, Jaszkiewicz A (1998) Evaluating the quality of approximations to the non-dominated set. Technical University of Denmark, Technical Report IMM-REP-1998-7, Denmark

Hassanat A, Almohammadi K, Alkafaween E, Abunawas E, Hammouri A, Prasath VBS (2019) Choosing mutation and crossover ratios for genetic algorithms-a review with a new dynamic approach. Information 10(12):390. https://doi.org/10.3390/info10120390

Hinterding R, Michalewicz Z, Eiben AE (1997) Adaptation in evolutionary computation: a survey. In: Proceedings of 1997 IEEE international conference on evolutionary computation (ICEC '97). IEEE, Indianapolis

Huang C, Li Y, Yao X (2020) A survey of automatic parameter tuning methods for metaheuristics. IEEE Trans Evol Comput 24(2):201–216. https://doi.org/10.1109/tevc.2019.2921598

Huang C, Bai H, Yao X (2022) Online algorithm configuration for differential evolution algorithm. Appl Intell. https://doi.org/10.1007/s10489-021-02752-1

Huband S, Barone L, While L, Hingston P (2005) A scalable multi-objective test problem toolkit. In: Evolutionary multi-criterion optimization, third international conference, EMO 2005, lecture notes in computer science, vol 3410. Springer, Guanajuato, México, pp 280–295, https://doi.org/10.1007/978-3-540-31880-4_20

Ishibuchi H, Yoshida T, Murata T (2003) Balance between genetic search and localsearch in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Trans Evol Comput 7(2):204–223. https://doi.org/10.1109/tevc.2003.810752

Ishibuchi H, Masuda H, Tanigaki Y, Nojima Y (2015) Modified distance calculation in generational distance and inverted generational distance. In: Proceedings of 8th international conference on evolutionary multi-criterion optimization. Springer, Guimaraes, Portugal, pp 110–125, https://doi.org/10.1007/978-3-319-15892-1_8

Ishibuchi H, Imada R, Setoguchi Y, Nojima Y (2018) How to specify a reference point in hypervolume calculation for fair performance comparison. Evol Comput 26(3):411–440. https://doi.org/10.1162/evco_a_00226

Jiang S, Ong YS, Zhang J, Feng L (2014) Consistencies and contradictions of performance metrics in multiobjective optimization. IEEE Trans Cybern 44(12):2391–2404. https://doi.org/10.1109/tcyb.2014.2307319

Jong KAD (1975) Analysis of the beavior of a class of genetic adaptive systems. PhD thesis, Computer and Communication Sciences Department, University of Michigan

Karafotias G, Hoogendoorn M, Eiben AE (2015) Parameter control in evolutionary algorithms: trends and challenges. IEEE Trans Evol Comput 19(2):167–187. https://doi.org/10.1109/tevc.2014.2308294

Knowles J, Corne D (2002) On metrics for comparing nondominated sets. In: Proceedings of the 2002 congress on evolutionary computation. CEC'02, vol 1. IEEE, Honolulu, pp 711–716, https://doi.org/10.1109/cec.2002.1007013

Knowles JD (2002) Local-search and hybrid evolutionary algorithms for pareto optimization. PhD thesis, Department of Computer Science, University of Reading

Kochenderfer MJ, Wheeler TA (2019) Algorithms for optimization. The MIT Press, Cambridge and London

Korejo I, Yang S, Li C (2009) A comparative study of adaptive mutation operators for genetic algorithms. In: The VIII metaheuristic international conference, Hamburg, Germany

Kundu T, Garg H (2022) A hybrid ITLHHO algorithm for numerical and engineering optimization problems. Int J Intell Syst 37(7):3900–3980. https://doi.org/10.1002/int.22707

Kundu T, Garg H (2022) LSMA-TLBO: a hybrid SMA-TLBO algorithm with lévy flight based mutation for numerical optimization and engineering design problems. Adv Eng Software. https://doi.org/10.1016/j.advengsoft.2022.103185

Kursawe F (1991) A variant of evolution strategies for vector optimization. In: Schwefel HP, Männer R (eds) Parallel problem solving from nature. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 193–197, https://doi.org/10.1007/BFb0029752

Lacerda MGPd, Pessoa LFdA, Neto FBdL, Ludermir TB, Kuchen H (2021) A systematic literature review on general parameter control for evolutionary and swarm-based algorithms. Swarm Evol Comput. https://doi.org/10.1016/j.swevo.2020.100777

Lee CY, Yao X (2004) Evolutionary programming using mutations based on the Lévy probability distribution. IEEE Trans Evol Comput 8(1):1–13. https://doi.org/10.1109/tevc.2003.816583

Li M, Yao X (2019) Quality evaluation of solution sets in multiobjective optimisation: a survey. ACM Comput Surv (CSUR) 52(2):26. https://doi.org/10.1145/3300148

Liu Z, Chen G, Ong C, Yao Z, Li X, Deng J, Cui F (2023) Multi-objective design optimization of stent-grafts for the aortic arch. Mater Des. https://doi.org/10.1016/j.matdes.2023.111748

Lobo FG, Lima CF, Michalewicz Z (2007) Parameter setting in evolutionary algorithms, studies in computation intelligence, vol 54. Springer, Berlin, Heidelberg,. https://doi.org/10.1007/978-3-540-69432-8

Long Q, Li G, Jiang L (2022) A novel solver for multi-objective optimization: dynamic non-dominated sorting genetic algorithm (DNSGA). Soft Comput 26(2):725–747. https://doi.org/10.1007/s00500-021-06223-0

Lopez EM, A C, Coello C (2016) IGD+ -EMOA: a multi-objective evolutionary algorithm based on IGD+. In: 2016 IEEE congress on evolutionary computation (CEC), pp 999–1006, https://doi.org/10.1109/cec.2016.7743898

López-Ibánez M, Dubois-Lacoste J, Cáceres LP, Birattari M, Stutzle T (2016) The irace package: iterated racing for automatic algorithm configuration. Oper Res Perspect 3:43–58. https://doi.org/10.1016/j.orp.2016.09.002

Mezura-Montes E, Palomeque-Ortiz AG (2009) Self-adaptive and deterministic parameter control in differential evolution for constrained optimization. In: Constraint-handling in evolutionary optimization, pp 95–120, https://doi.org/10.1007/978-3-642-00619-7_5

Mohamed A, Oliva D, Suganthan P (2022) Handbook of nature-inspired optimization algorithms: the state of the art: Volume II: solving constrained single objective real-parameter optimization problems. Studies in systems, decision and control, Springer International Publishing https://doi.org/10.1007/978-3-031-07516-2

Nama S, Sharma S, Saha AK, Gandomi AH (2022) A quantum mutation-based backtracking search algorithm. Artif Intell Rev 55(4):3019–3073. https://doi.org/10.1007/s10462-021-10078-0

Nebro AJ, Luna F, Alba E, Dorronsoro B, Durillo JJ, Beham A (2008) AbYSS: adapting scatter search to multiobjective optimization. IEEE Trans Evol Comput 12(4):439–457. https://doi.org/10.1109/tevc.2007.913109

Okabe T, Jin Y, Sendhoff B (2003) A critical survey of performance indices for multi-objective optimisation. In: The 2003 congress on evolutionary computation, 2003. CEC '03, vol 2. IEEE, Canberra, Australia, pp 878–885, https://doi.org/10.1109/cec.2003.1299759

Ozcelikkan N, Tuzkaya G, Alabas-Uslu C, Sennaroglu B (2022) A multi-objective agile project planning model and a comparative meta-heuristic approach. Inf Softw Technol. https://doi.org/10.1016/j.infsof.2022.107023

Papa G, (2021) Applications of dynamic parameter control in evolutionary computation. In, (2021) Genetic and evolutionary computation conference companion (GECCO '21 Companion). ACM, Lille, France, proceedings of the genetic and evolutionary computation conference companion, DOI 10(1145/3449726):3461435

Parpinelli RS, Plichoski GF, Silva RSD, Narloch PH (2019) A review of techniques for online control of parameters in swarm intelligence and evolutionary computation algorithms. Int J Bio-Inspired Comput 13(1):1. https://doi.org/10.1504/ijbic.2019.097731

Rahimi I, Gandomi AH, Deb K, Chen F, Nikoo MR (2022) Scheduling by NSGA-II: review and bibliometric analysis. Processes 10(1):98. https://doi.org/10.3390/pr10010098

Rajabi A, Witt C (2020) Self-adjusting evolutionary algorithms for multimodal optimization. In: Proceedings of GECCO '20. ACM Press, Cancun, Mexico, pp 1314–1322, https://doi.org/10.1007/s00453-022-00933-z

Rechenberg I (1971) Evolutionsstrategie; optimierung technischer systeme nach prinzipien der biologischen evolution. PhD thesis, Department of Process Engineering, Technical University of Berlin

Riquelme N, Lucken CV, Barán B, (2015) Performance metrics in multi-objective optimization. In: 2015 Latin American computing conference (CLEI). IEEE, Arequipa, Perú,. https://doi.org/10.1109/clei.2015.7360024

Schaffer JD (1985) Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the 1st international conference on genetic algorithms. L. Erlbaum Associates Inc., Sheffield, UK, pp 93–100

Schott JR (1995) Fault tolerant design using single and multicriteria genetic algorithm optimization. PhD thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology

Sharma S, Khodadadi N, Saha AK, Gharehchopogh FS, Mirjalili S (2023) Non-dominated sorting advanced butterfly optimization algorithm for multi-objective problems. J Bionic Eng 20(2):819–843. https://doi.org/10.1007/s42235-022-00288-9

Smith J, Fogarty T (1996) Self adaptation of mutation rates in a steady state genetic algorithm. In: Proceedings of 1996 IEEE international conference on evolutionary computation. IEEE, Nagoya, Japan, pp 318–323, https://doi.org/10.1109/icec.1996.542382

Smith JE, Fogarty TC (1997) Operator and parameter adaptation in genetic algorithms. Soft Comput 1(2):81–87. https://doi.org/10.1007/s005000050009

Srinivas N, Deb K (1994) Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput 2(3):221–248. https://doi.org/10.1162/evco.1994.2.3.221

Storn R, Price K (1997) Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359. https://doi.org/10.1023/a:1008202821328

Tan K, Chiam S, Mamun A, Goh C (2009) Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. Eur J Oper Res 197(2):701–713. https://doi.org/10.1016/j.ejor.2008.07.025

Tanabe R, Ishibuchi H (2020) An Analysis of Quality Indicators Using Approximated Optimal Distributions in a 3-D Objective Space. IEEE Trans Evol Comput 24(5):853–867

Tanaka M, Watanabe H, Furukawa Y, Tanino T (1995) GA-based decision support system for multicriteria optimization. In: 1995 IEEE international conference on systems, man and cybernetics. Intelligent systems for the 21st century, vol 2. IEEE, Vancouver, British Columbia, Canada, pp 1556–1561, https://doi.org/10.1109/icsmc.1995.537993

Tinós R, Yang S (2007) Self-adaptation of mutation distribution in evolutionary algorithms. In: 2007 IEEE congress on evolutionary computation. IEEE, Singapore, pp 79–86, https://doi.org/10.1109/cec.2007.4424457

Veldhuizen DAV, Lamont GB (1998) Evolutionary computation and convergence to a pareto front. Late-breaking papers book at the genetic programming 1998 conference (GP-98). Stanford University Bookstore, Winsconsin, pp 221–228

Wang J, Liu Y, Ren S, Wang C, Ma S (2023) Edge computing-based real-time scheduling for digital twin flexible job shop with variable time window. Robot Comput Integr Manuf. https://doi.org/10.1016/j.rcim.2022.102435

Wang S, Ali S, Yue T, Li Y, Liaaen M (2016) 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, Austin, https://doi.org/10.1145/2884781.2884880

Yang S, Uyar S (2006) Adaptive mutation with fitness and allele distribution correlation for genetic algorithms. In: Proceedings of the 2006 ACM symposium on Applied computing - SAC '06. ACM, Dijon, France, pp 940–944, https://doi.org/10.1145/1141277.1141499

Zeng F, Low MYH, Decraene J, Zhou S, Cai W (2010) Self-adaptive mechanism for multi-objective evolutionary algorithms. In: Proceedings of the international multiconference of engineers and computer scientists pp. 7–12

Zhang J, Chen WN, Zhan ZH, Yu WJ, Li YL, Chen N, Zhou Q (2012) A survey on algorithm adaptation in evolutionary computation. Front Electr Electr Eng 7(1):16–31. https://doi.org/10.1007/s11460-012-0192-0

Zhao Z, Liu B, Zhang C, Liu H (2019) An improved adaptive NSGA-II with multi-population algorithm. Appl Intell 49(2):569–580. https://doi.org/10.1007/s10489-018-1263-6

Zhou A, Jin Y, Zhang Q, Sendhoff B, Tsang E (2006) Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In: 2006 IEEE international conference on evolutionary computation. IEEE, Vancouver, pp 892–899, https://doi.org/10.1109/cec.2006.1688406

Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans Evol Comput 3(4):257–271. https://doi.org/10.1109/4235.797969

Zitzler E, Deb K, Thiele L (1999) Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 8(2):173–195. https://doi.org/10.1162/106365600568202

Zitzler E, Thiele L, Laumanns M, Fonseca CM, Fonseca VGd (2003) Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans Evol Comput 7(13):117–132. https://doi.org/10.1109/tevc.2003.810758