Assessing ranking and effectiveness of evolutionary algorithm hyperparameters using global sensitivity analysis methodologies

Swarm and Evolutionary Computation 74 (2022) 101130

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Assessing ranking and effectiveness of evolutionary algorithm hyperparameters using global sensitivity analysis methodologies

Varun Ojha [*,a], Jon Timmis [b], Giuseppe Nicosia [c,d]

[a] University of Reading, Reading, United Kingdom
[b] The University of Sunderland, Sunderland, United Kingdom
[c] University of Catania, Catania, Italy
[d] University of Cambridge, Cambridge, United Kingdom

ARTICLE INFO

Keywords:
Hyperparameter optimization
Evolutionary algorithms
Global sensitivity analysis
Algorithm design
Algorithm stability analysis

ABSTRACT

We present a comprehensive global sensitivity analysis of two single-objective and two multi-objective state-of-the-art global optimization evolutionary algorithms as an *algorithm configuration problem*. That is, we investigate the *quality of influence* hyperparameters have on the performance of algorithms in terms of their *direct effect* and *interaction effect* with other hyperparameters. Using three sensitivity analysis methods, Morris LHS, Morris, and Sobol, to systematically analyze tunable hyperparameters of covariance matrix adaptation evolutionary strategy, differential evolution, non-dominated sorting genetic algorithm III, and multi-objective evolutionary algorithm based on decomposition, the framework reveals the behaviors of hyperparameters to sampling methods and performance metrics. That is, it answers questions like what are hyperparameters influence patterns, how they interact, how much they interact, and how much their direct influence is. Consequently, the ranking of hyperparameters suggests their order of tuning, and the pattern of influence reveals the *stability of the algorithms*.

1. Introduction

Optimization is at the core of advancement in machine learning and problem-solving. Effective optimization plays a vital role in solving problems, whether single-objective or multi-objective problems. For example, be it a simple neural network or deep learning, or a simple linear or nonlinear function, optimizing the coefficients (e.g., weights of neural networks) is the most crucial aspect, which requires effective optimization algorithms. Evolutionary algorithms (EAs) are global optimization algorithms that iteratively guide a population towards a final population, solving various problems. EAs are widely used because of their agnostic nature to problems being solved [1]. However, their effectiveness relies on hyperparameters like population size and genetic operators [2]. Understanding the sensitivity of hyperparameters to an algorithm's performance can be formulated as an *algorithm configuration problem* (ACP) [3,4], where informing optimal hyperparameter selection is essential for solving various tasks such as the optimization of neural networks [5], deep learning [6], and bio-inspired algorithms [7,8]. More specifically, ACP can be described as a process or a framework that aims to find a particular configuration of parameters for a target algorithm. And it minimizes a cost metric incurred by the algorithm on a given problem [9].

Since hyperparameters tuning is crucial in achieving high-quality performance in solving optimization problems, methods such as manual tuning, grid search, and Bayesian search optimization are used. Bergstra and Bengio [10] have shown the importance of *random search* instead of a *grid search* in sampling hyperparameter values. In addition, *manual tuning* without proper knowledge of hyperparameters can lead to too many trial-and-errors, and grid search and Bayesian search optimization are computationally expensive approaches that are often infeasible for such population-based optimization algorithms. Thus, Bergstra and Bengio [10] suggest that tuning some hyperparameters is more necessary than the others. Hence, our objective in this research is to assess the ranking and effectiveness of hyperparameters of four well-known EAs: covariance matrix adaptation evolutionary strategy (CMA-ES) [11], differential evolution (DE) [12], non-dominated sorting genetic algorithm III (NSGA-III) [13], and multi-objective evolutionary algorithm based on decomposition (MOEA/D) [14].

We select these algorithms as they are state of the art algorithms in single-objective and multi-objective optimization. They are the highly cited algorithms not only within the scientific community of bio-inspired computation but also in other scientific disciplinary areas

* Corresponding author.
E-mail address: v.k.ojha@reading.ac.uk (V. Ojha).

https://doi.org/10.1016/j.swevo.2022.101130
Received 14 October 2021; Received in revised form 11 April 2022; Accepted 11 July 2022
Available online 23 July 2022

such as operations research, applied mathematics, electrical engineering, civil engineering and many other research areas [15]. These algorithms are widely used in multiple multidisciplinary/interdisciplinary problems and are widely used to address real-world problems and open problems in a wide variety of research areas.

Moreover, researchers have massively investigated these algorithms to improve their performance. For example, a number of improvements to DE have been provided, including success-history based adaptive DE versions [16,17], mutation operator improvement [7,18–22] and scaling factor in mutation for accelerating convergence [23]. Similarly, an improved step size (mutation) strategy for the CMA-ES algorithm is investigated by Voß et al. [24], improved decomposition strategy like normal boundary intersection-style Tchebycheff approach, adaptive replacement strategies to assign a new solution to a sub-problem, and adaptive weight vector adjustment strategy for sub-problems, respectively proposed by Zhang et al. [25], Wang et al. [26], Qi et al. [27] for MOEA/D algorithm. Similarly, for NSGA-III performance enhancement, Cui et al. [28] designed an operator to balance the convergence and diversity of the population.

Such usefulness makes these algorithms suitable candidates to be the example of algorithms that can be used as test-beds for the sensitivity analysis methodology presented in our research work. Obviously, our methodology applies to all optimization algorithms with parameters (e.g., evolutionary, randomized, hybrid, constrained [29], dynamic [15], etc.). The algorithms used in the paper are only a good sample of single- and multi-objective optimization algorithms. We think the optimization research community and interdisciplinary research community mentioned above will benefit as many more optimization algorithms, including many-objective optimization algorithms [30–32], that can be studied using the methodology proposed in this paper.

In this work, we develop a framework for comprehensive sensitivity analysis of hyperparameters of these algorithms using global sensitivity analysis methodologies: elementary effects [33] and variance-based sensitivity analysis [34]. Using these methodologies, we assess the effectiveness of EA hyperparameters. Such an analysis investigates a model's parameters (or an algorithm's hyperparameters) influence on its output [35,36], leading to the minimization of the number of critical tunable hyperparameters to improve a model's performance [37,38].

In our ACP framework, the performance of single-objective EAs was assessed as per the *best solution*, while the performance of multi-objective EAs was assessed using three metrics: *generational distance* [39,40], *inverse generational distance* [13], and *hyper-volume indicator index* [41]. To evaluate EAs, we use state-of-the-art optimization problems belonging to diverse families: for single-objective optimization, we use a set of 33 problems [42–44], and for multi-objective optimization, we use a set of 10 problems [13].

Our ACP framework assesses each algorithm on three sensitivity analysis methods: Morris Latin Hypercube sampling [33], Morris sampling [33], and Sobol [34]. For each sample drawn from a hyperparameter search space, we ran each algorithm on 30 independent runs (for some, it was 10 times) and presented results using elementary effects and Sobol indices. These indices inform about (i) the direct effect and (ii) the interaction effect of a hyperparameter with other hyperparameters. Moreover, these two effects form a comparative matrix of low effect to high effect, where the diagonal from low direct and low interaction effects to high direct and high interaction effects shows the *order and ranking* of the hyperparameters. We ran algorithms on a sufficiently large sample set. These experiments were computationally expensive as they, in total, had 19 014 600 000 function evaluations. Computation of these sensitivity analysis indices is expensive, but they are a one-time effort, and once the ranking is determined, results are informative to researchers for further analysis and solving optimization problems. The source code and results are available at https://github.com/vojha-code/SAofEAs.

Our results reveal the pattern and behavior of hyperparameters to different sampling methods and matrices used to evaluate the performance of the algorithm. These patterns show how hyperparameters interact with one another or how the influence of one hyperparameter overwhelms the other. Moreover, results reveal how an algorithm is susceptible to its various hyperparameters and sampling methods, highlighting the stability of an algorithm. Consequently, these experiments rank the hyperparameter importance for an algorithm. For example, mutation type was found to have the strongest influence on the performance of DE, and results suggest the high importance of population size followed by the initial step size, crossover probability, and mode of decomposition, respectively, in CMA-ES, NSGA-III, and MOEA/D.

Later in Section 2, we present related work. Then, Sections 3–5 respectively describe algorithms, methodology, and experiments. The results are discussed in Section 6, followed by conclusions in Section 7. Supplementary A and B offer statistical tests and clustering analysis.

2. Related work

Hyperparameter tuning is a crucial subject that has continuously been reported in the literature over the past decades [2]. This is because an appropriate hyperparameter setting is challenging since EA hyperparameters exhibit linear and nonlinear effects [45], meaning that they show various interactions among them [2,46,47]. Abundant literature is available on EA hyperparameters tuning [45,46,48]. The majority of which focus on the *static* or *dynamic* setting of the hyperparameters [49–51]. However, a systematic study of the EA hyperparameters influence is rare [52], and it is largely attributed to the computationally expansive nature of EAs and the empirical evaluation requirement for the tuning of their hyperparameters [53]. For example, a package *Irace* experimentally evaluates optimal hyperparameters for an optimization algorithm [3]. Therefore, De Jong [2] posed questions like (i) what EA hyperparameters are useful for improving performance, and (ii) how do changes in a hyperparameter affect the performance of an EA?

Sensitivity analysis answers questions like *how uncertainty in each of the hyperparameters influences the uncertainty in the output of a model* [54]. Hence, sensitivity analysis is useful in answering the questions of De Jong [2]. However, sensitivity analysis is a computationally expansive method since hyperparameters are sampled from a vast hyperparameter search space. Therefore, the sensitivity analysis of EAs has very high computational (time) as well as memory (space) overhead. This has resulted in very few reported works available in the literature, despite its advantages in suggesting a ranking of hyperparameter importance.

The dynamic tuning of hyperparameters requires hyperparameters to adapt during an EA run [55], while static tuning informs which hyperparameters to tune before an EA run [50]. A systematic approach, like sensitivity analysis, is a static hyperparameter tuning approach. Paul et al. [56] offered an introductory work on the usage of *local* and *global* sensitivity analysis. However, they used a simple test case, and they mainly performed a sensitivity analysis of EAs from a theoretical perspective. Pinel et al. [52] performed a comprehensive sensitivity analysis of a parallel asynchronous cellular genetic algorithm on a scheduling problem. They comprehensively evaluated EAs population size, mutation probability, crossover probability, and other cellular genetic algorithm-related hyperparameters using the Fourier amplitude sensitivity test (Fast99) [57]. Pinel et al. [52] reported a ranking of hyperparameters on scheduling problem instances. On this scheduling problem instance, the crossover probability was ranked first, and in another instance, it was ranked third.

Our work takes an experimental approach to systematically analyze the importance of hyperparameters of state-of-the-art EAs on a testbench of state-of-the-art problems by applying Morris [33] and Sobol [34] sensitivity analysis methodologies. Our methodology comprises both single-objective and multi-objective EAs. Our framework offers a ranking of hyperparameters and insights into their effectiveness on EA performance. Our methodology is an *Algorithm Configuration Problem* (ACP) framework as defined by Iommazzo et al. [4]. This approach is

contextually similar to the AutoML approaches [58], where the effort is to find the optimal configuration of algorithms and hyperparameters to solve machine learning tasks through automatic data preparation, feature engineering, hyperparameter optimization, and neural architecture search or even optimization of neural network components such as activation functions [59]. Table 1 is a summary of hyperparameter methods compared to sensitivity analysis methods.

In fact, the ACP scope covers a wider range of methodologies and frameworks that seek to automate the design of algorithm configuration, such as AutoMOEA [60], Auto Weka [61], Auto-sklearn [62], irace [3], and others for machine learning hyperparameter optimization [63]. The goal of these methodologies is to perform hyperparameter optimization and automatic design of new algorithms by assessing components and parameters that offer the best performance on a set of problem instances [4,61]. The critical issue in such categorization is whether one would consider, for instance, a new evolutionary operator design in an EA framework as a new algorithms design or hyperparameter optimization? In our work, we consider such a scenario as hyperparameter optimization. However, we considered the ACP framework for the analysis of the sensitivity and influence of the hyperparameters on the performance of an algorithm rather than the optimization (or tuning) of the hyperparameters. For this, the framework systematically searches hyperparameters and assesses the performance of an algorithm, which is contrary to finding specific optimal values for a hyperparameter as other hyperparameter tuning methods would do. Hence, the goal of our ACP framework is to inform the ranking of the effectiveness of hyperparameters for a set of EAs.

3. Evolutionary algorithms

EAs are population-based evolution-inspired algorithms. EAs iteratively find solutions to a problem by applying evolutionary operators to candidate solutions. Selection, recombination, and mutation are among evolutionary operators applied to candidate solutions that generate new solutions in each generation. Such a process guides a sequence of generations from an initial population of candidate solutions to a final population. Four different EAs are investigated in this research: two single-objective and two multi-objective algorithms. Each of these EAs has its own version of evolutionary operators. This Section briefly describes each of these EAs and their performance measure metrics.

3.1. Single-objective evolutionary algorithms

A single-objective optimization (SOO) algorithm (single solution-based or population-based) *minimizes an objective function* (a cost function or a problem) as

$$f: \begin{array}{l}\mathbb{R}^n \to \mathbb{R} \\ \mathbf{x} \mapsto f(\mathbf{x}),\end{array} \tag{1}$$

Table 1
Hyperparameter tuning methods and sensitivity analysis (our framework). Hyperparameter tuning methods are search techniques for optimal hyperparameter values whereas our framework finds ordering of their significance and their sensitivity of influence on the algorithm.

Method	Tuning	Type	Use
Manual Tuning	static	requires intuitive guesses	trails and errors
Grid Search	static	systematic search	uninformed search
Bayesian Search	static	informed search	expansive and specific to instances
AutoMOEA	dynamic	systematic and informed	expansive and subjective
AutoML	dynamic	informed search	expansive and specific to problems
Our Framework	Static	ranking and analysis	expansive but one at a time

where $\mathbf{x} \in \mathbb{R}^n$ is a candidate solution (a search point in a solution space X), and we want $f(\mathbf{x})$ to be as minimum as possible. An SOO algorithm converges to a solution $\mathbf{x}^*$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x}),\ \forall \mathbf{x} \in X$. The solution $\mathbf{x}^*$, therefore, is a *global minimum* (global optimum). However, if for $f(\mathbf{x}^*) \leq f(\mathbf{x})$ there exists some $\delta > 0$ such that $|\mathbf{x} - \mathbf{x}^*| \leq \delta$ for any $\mathbf{x} \in X$, then the solution $\mathbf{x}^*$ is a *local minimum* (near-optimum).

We study two population-based single-objective global optimization algorithms: CMA-ES [11] and DE [12]. The basic steps and operators of CMA-ES and DE are as follows.

3.1.1. Covariance matrix adaptation evolution strategies (CMA-ES)

CMA-ES is a population-based *evolutionary strategy* optimization algorithm [11]. CMA-ES algorithm generates new candidate solutions during its search by sampling solutions from a *multivariate normal distribution*, $\mathcal{N}(\mathbf{m}, C)$, uniquely determined by its mean $\mathbf{m} \in \mathbb{R}^n$ and its symmetric positive definite covariance matrix $C \in \mathbb{R}^{n \times n}$. The initial population of λ candidate solutions at generation $g = 0$ is sampled as

$$\mathbf{x}_k^g \sim \mathbf{m}^g + \sigma^g \mathcal{N}(\mathbf{0}, C^g) \quad \text{for } k = 1, \ldots, \lambda, \tag{2}$$

where $\mathcal{N}(\mathbf{0}, C)$ is a multivariate normal distribution with zero mean and covariance matrix $C^g \in \mathbf{I}$, and $\sigma^g \in \mathbb{R}_{>0}$ is an initial step size.

For generation $g = 1, 2, \ldots$, multivariate normal distribution $\mathcal{N}(\mathbf{m}, C^{g+1})$ is generated (updated) with mean $\mathbf{m} \in \mathbb{R}^n$ and covariance matrix $C \in \mathbb{R}^{n \times n}$ updated with scalar factor $\sigma^g \in \mathbb{R}_{>0}$. Selection and recombination operations in CMA-ES are equivalent to computing moving mean m^{g+1}, a weighted average of selected points λ_{ratio} from generation g. Adding a random vector with zero-mean acts as a mutation in CMA-ES during the offspring generation step. The steps size control and covariance matrix adaptation (learning rate α_μ) are additional two necessary steps in a generation of CMA-ES [11].

3.1.2. Differential evolution (DE)

DE is a *gradient-free* EA, originally proposed by Storn and Price [12]. DE iteratively searches for a solution. For an initial population $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\lambda]$ of size λ, DE repeats its steps *selection, mutation*, and *recombination* until an optimum solution vector $\mathbf{x}^*$ is obtained, or until a maximum iteration is reached. At each generation $g = 1, 2, \ldots$, DE randomly selects three distinct candidate solutions $\mathbf{x}_{r1}^g$, $\mathbf{x}_{r2}^g$, and $\mathbf{x}_{r3}^g$ from X such that $\mathbf{x}_{r1}^g \neq \mathbf{x}_{r2}^g \neq \mathbf{x}_{r3}^g$. The selection of a base vector $\mathbf{x}_{r1}^g$ plays a crucial in DE.

A mutation operation is performed on a base vector $\mathbf{x}_{r1}^g$ to generate a donor vector $\mathbf{v}^{g+1}$, which is generated using a mutation method $\mathbf{b}_{\text{type}}$, a difference vector $(\mathbf{x}_{r2}^g \mathbf{x}_{r3}^g)$, and acceleration coefficient β. A mutation method $\mathbf{b}_{\text{type}}$ = "DE/rand/1" or similar mutation is performed as

$$\mathbf{v}^{g+1} = \mathbf{x}_{r1}^g + \beta(\mathbf{x}_{r2}^g \mathbf{x}_{r3}^g). \tag{3}$$

A crossover operation using a crossover method {bin, exp} is performed to generate a trial vector $\mathbf{u}^{g+1}$ which takes its elements from a donor vector $\mathbf{v}^{g+1}$ using a crossover probability P[X]. If the fitness $f(\mathbf{u}^{g+1})$ is better than the target vector $f(\mathbf{x}_t^{g+1})$, then the trial vector $\mathbf{u}^{g+1}$ replaces the target vector $\mathbf{x}_t^{g+1}$.

3.2. Multi-objective evolutionary algorithms

A multi-objective optimization (MOO) algorithm minimizes two or more objective functions *simultaneously* as

$$F(\mathbf{x}) \equiv (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})), \text{ i.e., } F: \mathbb{R}^n \to \mathbb{R}^k \text{ for } k \geq 2 \tag{4}$$

such that no one objective of the problem can be improved without a simultaneous detriment to at least one of the other objectives. Each $f_l(\mathbf{x})$, $l = 1, 2, \ldots, k$ is a scalar objective, and MOO optimizes the objective vector $F(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^n$ is its feasible solution. More specifically, a MOO algorithm produces a set of non-dominated solutions $\{\mathbf{x}_1, \mathbf{x}_2, \ldots,$

$\mathbf{x}_{\lambda'}\}$, also known as the Pareto-optimal solutions set [64].

A solution $\mathbf{x}_i$ dominates other solution $\mathbf{x}_j$ if for $j = 1,2,\ldots,\lambda, i \neq j$, and for all objectives $l = 1,2,\ldots,k$, $f_l(\mathbf{x}_i) \preccurlyeq f_l(\mathbf{x}_j)$ holds, where $\preccurlyeq$ should be read as "better off." On the contrary, a solution $\mathbf{x}_i$ is non-dominated if, for at least one objective l, $f_l(\mathbf{x}_i) \preccurlyeq f_l(\mathbf{x}_j)$ does not hold. For each $\mathbf{x}_i$, a set of such non-dominated solutions are called a Pareto-optimal set of solutions.

In this paper, we study the population-based multi-objective global optimization algorithms NSGA-III [13] and MOEA/D [14] and investigate their algorithmic hyperparameter setting in obtaining a better Pareto-optimal set of solutions.

3.2.1. Non-dominated sorting genetic algorithm–III (NSGA-III)

NSGA-III is a population-based MOO algorithm [13]. NSGA-III uses fast non-dominated sorting and niching operations to guide an initial population X of size λ candidate solutions through a predefined number of generations to a final population while simultaneously optimizing trade-offs of multiple objectives. In each step of NSGA-III, crossover, mutation, and non-dominated sorting is performed.

The fast non-dominated sorting sorts the λ candidate solutions into several sets (called Fronts) of non-dominated solutions: $F_1, F_2, \ldots, F_s$ such that the Front F_1 contains all the non-dominated candidate solutions of population X. That is, no one solution in F_1 is dominated by any other solutions. From all the remaining solutions (i.e., except the ones already in F_1), a new Front F_2 that contains all the next non-dominated solutions of X is determined. Similarly, Front F_3 and other Fronts are subsequently obtained using non-dominated sorting. Thus, it is possible to assign a rank to the candidate solutions such that those on the Front F_1 have rank 1, solutions in Front F_2 have rank 2, and so on.

NSGA-III performs *niching* as its selection operation on non-dominated sorting solutions. Niching takes advantage of a predefined set of reference points placed on a normalized hyperplane of a k-dimensional objective-space [65], where each individual $\mathbf{x} \in X$ in the population is associated with reference points [13]. The total number of reference points depends on the predefined number of divisions associated with each objective axis. NSGA-III repeats its operations selection, crossover, mutation, and recombination until a maximum iteration or a termination condition is reached. The performance of NSGA-III is measured in terms of the quality of solutions it produces in its iteration and in the final population.

3.2.2. Multi-objective evolutionary algorithm based on decomposition (MOEA/D)

MOEA/D solves a MOO problem by decomposing the MOO problem into many single (scalar) objective sub-problems [14]. Tchebycheff approach [66] or normal boundary interaction approach [65] are typically used approaches for decomposing a MOO problem into (say) N scalar sub-problems. A uniform spread of N weight vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$ and reference point $\mathbf{z}^* = (z_j^1, \ldots, z_j^k) = \min\{f_i(\mathbf{x}) | \mathbf{x} \in X\}$, for $i = 1, \ldots, k$ is used for computing $j = 1, 2, \ldots, N$ scalar objectives $y^{te}(\mathbf{x}|\mathbf{w}_j)$.

The scalar objective in Tchebycheff decomposition method is $y^{te}(\mathbf{x}|\mathbf{w}_j) = \max_{1 \leq i \leq k}\{\mathbf{w}_j^i | f_i(\mathbf{x}) - \mathbf{z}^*\})$, where the weight vector $\mathbf{w}_j = (w_j^1, \ldots, w_j^k)$. The optimal solution of $y^{te}(\mathbf{x}|\mathbf{w}_i)$ for weight vector $\mathbf{w}_i$ should be close to a solution $y^{te}(\mathbf{x}|\mathbf{w}_j)$ for weight vector $\mathbf{w}_j$. Hence, in MOEA/D, a neighborhood of weight vector $\mathbf{w}_i$ is defined with many closest points in $\{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$. The neighborhood may play a vital role in MOEA/D.

Moreover, each objective is optimized as a single (scalar) objective problem. That is, ith objective is optimized such that it minimizes its distance from a reference point on a k-objective space. Thus, all decomposed sub-problems move towards the reference point $\mathbf{z}^*$. MOEA/D maintains T closest solution vectors (Neighbor) for each candidate solution in successive steps. In each iteration, MOEA/D generates a new solution by selecting two solution vectors using genetic operators and evaluating them in order to update their neighborhood and the best solution $\mathbf{x}^*$. The details of the MOEA/D algorithm are available in Zhang and Li [14].

3.3. Performance metrics

3.3.1. Single objective metrics

A population-based EA applied to solve a single-objective problem offers the best solution in its final population. The best solution, $\mathbf{x}^*$ is the one that has the lowest $f(\mathbf{x})$ value among all solutions of all generations of a single-objective EA. Hence, the *Best Solution* obtained in fewer generations in a lesser wall clock time measures the quality of a single-objective EA.

3.3.2. Multi-objective metrics

Multi-objective EAs applied to a MOO problem typically offer a set of solutions that satisfy trade-offs between the objectives. This set of solutions is non-dominated solutions which are also known as a Pareto-front. A multi-objective EA, therefore, guides a population of candidate solutions from *current Pareto-front* **A** toward a *true Pareto-front* **Z**. In such a setting, three indicators are used to measure and compare the performance of EAs on MOO problems: generational distance, inverse generational distance, and hyper-volume indicator (Fig. 1):

Generational distance (GD) Generational distance GD_i at an iteration, i, measures the generational distance between *current Pareto-front* and *true Pareto-front* of a multi-objective problem [39,40]. Generational distance GD_i is a measure of error between *current Pareto-front* and *true Pareto-front* as

$$\mathrm{GD}_i \triangleq \frac{\sqrt{\sum_{i=1} d_i^2}}{n}, \tag{5}$$

where d_i^2 is the distance of the ith solution in *current Pareto-front* **A** from the *true Pareto-front* **Z** [40], and GD is typically the average distance of such n solutions (Fig. 1). Hence, GD is a minimization metric where a low value indicates a better solution.

Inverse generational distance (IGD) Inverse generational point provides combined information on the solutions' diversity and convergence quality. It makes use of a set of target reference points in k-dimensional objective space. Like GD, IGD compares solutions in the *current Pareto-front* **A** with *true Pareto-front* **Z**. However, IGD uses a single reference and computes the average Euclidean distance between all solutions that are nearest to the target reference points [13] as

$$IGD(\mathbf{A}, \mathbf{Z}) \triangleq \frac{1}{|\mathbf{Z}|} \sum_{i=1}^{|\mathbf{Z}|} \min_{j=1}^{|A|} d(\mathbf{z}_i, \mathbf{a}_j), \tag{6}$$

where $d(\mathbf{z}_i, \mathbf{a}_j) = \| \mathbf{z}_i, \mathbf{a}_j \|_2$. Similar to GD, IGD is a measure of error between the *current Pareto-front* and *true Pareto-front*. Hence, lower values of IGD indicate a better solution.

Hypervolume indicator (HV) Hyper-volume indicator, HV measures the dominance of Pareto-front solutions on a geometric space (e.g., area for a 2D objective space) framed by the k-dimensional objective-space with respect to a positive semi-axle r (see Fig. 1). Hence, HV measures the quality Pareto-optimal solutions set [67], and it is an indicator of the quality of the solutions obtained by two algorithms with respect to the same reference frame. The goal is to maximize the hyper-volume indicator index HV. A greater value indicates that the algorithm's overall performance is better with respect to another algorithm associated with a smaller hyper-volume value. Moreover, the greatest contributing point in a hyper-volume indicator analysis is the point covering the largest area, which can be considered the best solution [68].

4. Global sensitivity analysis

The goal of the *sensitivity analysis* is to study how the uncertainty of a model's output depends on the uncertainty of its inputs [69,70]. The *elementary effects* analysis, known as the "Morris method" [33], and

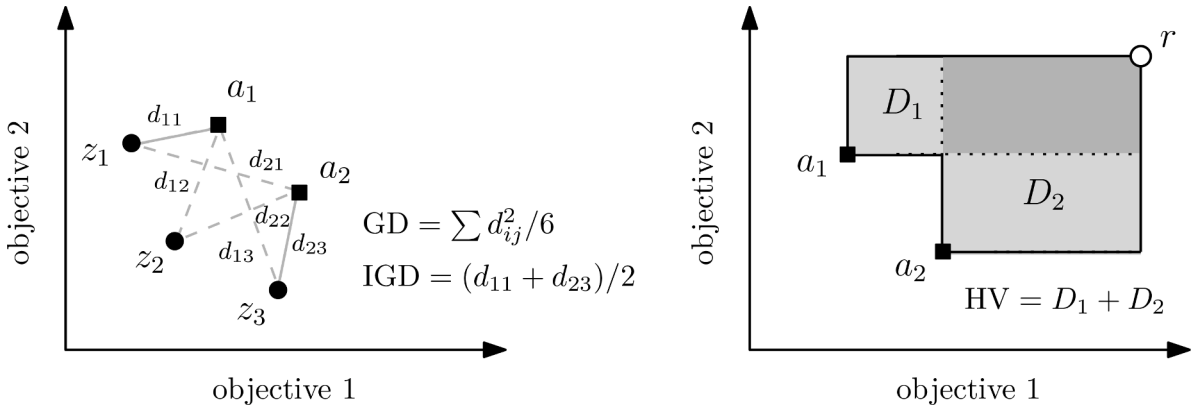

Fig. 1. Example of a 2D objective space and computation of GD, IGD, and HV metrics. The current Pareto front is $\mathbf{A} = \{a_1, a_2\}$ and true Pareto front is $\mathbf{Z} = \{z_1, z_2, z_2\}$, and optimum of two-objective is a reference point r. The distance between two points is d_{ij}, and the area framed by a point with the reference point, r, is the area D_i.

variance-based sensitivity analysis, known as the "Sobol method" [34], are used in this research for the global sensitivity analysis of the hyperparameters of four EAs. This framework of combining sensitivity analysis and EAs is an *algorithms configuration problem* that aims to inform algorithm performance to variations in hyperparameter on problem instances [4].

4.1. Elementary effects

The *elementary effects* (EE) technique, known as the "Morris method" as it was originally introduced by Morris [33], is an effective way to analyze the effects (sensitivity) of input variables on the outputs of a model or a system. In our case, the Morris method assesses the EE of the algorithmic hyperparameters on the performances of an EA. This is useful in analyzing the sensitivity of EA hyperparameters as the Morris method determines whether the effects of a hyperparameter on a model's outputs (EA performances on functions) are (a) insignificant and negligible, (b) linearly correlated, or (c) non-linearly correlated or involved in an interaction with other hyperparameters [70].

We briefly introduce the computation of EE as follows. Let us have $Y = f(\mathbf{X})$, or simply $Y(\mathbf{X})$ be the output of a model $f(\cdot)$ (an algorithm) that takes k hyperparameters $\mathbf{X} = \{X_1, X_2, \ldots, X_k\}$ from a hyperparameter space Ω of the *p-level* grid. Then we compute the *elementary effect* EE_i of ith hyperparameter X_i as

$$\mathrm{EE}_i = \frac{Y(X_1, \ldots, X_{i-1}, X_i + \Delta, \ldots, X_k) - Y(X_1, \ldots, X_{i-1}, X_i, \ldots, X_n)}{\Delta}, \tag{7}$$

where Δ is a value in $\left(\frac{1}{p-1}, \ldots, 1 - \frac{1}{p-1}\right)$ which is an incremental change in the values of hyperparameter X_i when X_i is sampled from *p-level* grid hyperparameter space Ω. In this scenario, for k hyperparameters and p discrete levels, $\Delta = p/2(p-1)$ indicates the distance (length) between two levels in the hyperspace Ω along ith axis. The total points in the hyperparameter space Ω, therefore, are $p^{k-1}[p - \Delta(p-1)]$ grid points, which increase exponentially as the number of hyperparameters k increases. However, we use a *one-at-a-time* (OAT) sampling technique for generating r sample points from this space to compute r EEs for each hyperparameter.

In the OAT sampling technique, hyperparameter X_i value is changed from a grid point $X_i^{(j)}$ to the adjacent grid point $X_i^{(j\pm1)}$ by a length of Δ while all other hyperparameters (say $X_{\sim i}$) remain as it is. Then the next hyperparameter X_{i+1} is chosen, whose value is changed while others remain fixed. This way of sampling is a *uniform, non-repeating random walk* through the grid of hyperspace Ω (we call it *Morris* [33]). Another way of sampling points (a set $\mathbf{X}$ of hyperparameters) from the hyperspace Ω is to use the Latin Hypercube Sample (LHS) based Morris method (*Morris LHS*) [71], which is a stratified sampling approach to cover all region of the hyperspace Ω. Here, we typically select r sample points for each hyperparameter X_i. Hence, both OAT-based Morris LHS and Morris sampling methods give us $r(k+1)$ sample points.

We measure two indices μ_i and σ_i indicate *mean* (central tendency) and *standard deviation* of EE_i of ith hyperparameter X_i. The measure

$$\mu_i = \frac{1}{r}\sum_{j=1}^{r} EE_i^j \tag{8}$$

indicates the overall influence of a hyperparameter X_i where a larger measure of μ_i means a larger *overall* individual ability to influence the outputs of an algorithm. We also measure the standard deviation σ_i of EE_i as

$$\sigma_i = \sqrt{\frac{1}{r-1}\sum_{j=1}^{r}\left(EE_i^j - \mu_i\right)^2}, \tag{9}$$

where a large measure of σ_i indicates that a hyperparameter has high interaction with other hyperparameters. The measure σ is an ensemble influence. That is, if σ_i has a high value, which means that the computed r elementary effects EE_i^r of ith hyperparameter X_i varied a lot because of the variation in the values of other hyperparameters as well. Whereas a low value of σ_i means small differences in the computed r elementary effects EE_i^r of the ith hyperparameter X_i. This indicates that the influence of a hyperparameter on a model's output is independent of the choice of other hyperparameters values. However, to understand the influence of a hyperparameter, both μ and σ measures need to be seen together (see Fig. 2). We normalized the values of μ_i and σ_i between 0 and 1 to effectively show results as per Fig. 2.

4.2. Variance-based sensitivity analysis

The *variance-based sensitivity analysis* is known as the "Sobol method" [34], and it shows how much variance of a model's output depends on its inputs. It is an in-depth sensitivity analysis method that uses two sensitivity indices: (a) *first-order effect* S_i to indicate a direct effect of a hyperparameter X_i on a model's output $Y = f(\mathbf{X})$ and (b) *total effect* ST_i to indicate a hyperparameter X_i interaction with its complementary parameters $X_{\sim i}$.

The direct effect S_i, irrespective of the hyperparameter interaction ST_i, indicates that, on average, how much the model's variance $V[Y(\mathbf{X})]$ could be reduced if the hyperparameter X_i is fixed to a value. Meaning a low value of S_i shows that the variance of the model's output $Y(\mathbf{X}|X_i = x_i^*)$ does not depend on X_i, and fixing X_i to a value does not have much impact on the model's output, while for a high value of S_i, it strongly does. Indeed, a low value of S_i indicates that ith hyperparameter's influence is negligible. Similarly, the interaction effect or total effect $ST_i = 0$ indicates that the model's output $Y(\mathbf{X}|X_i)$ does not depend on X_i, and it

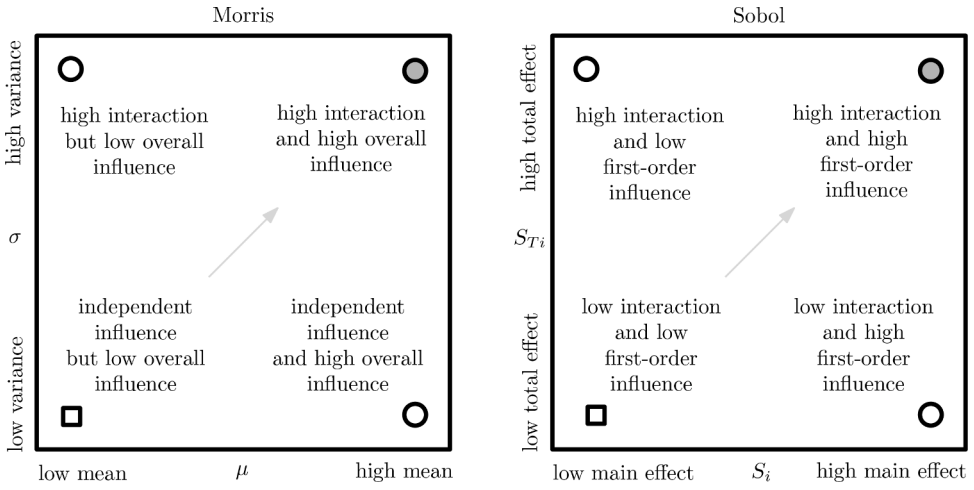

Fig. 2. Morris (*left*) and Sobol (*right*) indices interpretation. Top right corner circle in dark gray is the ideal case where a hyperparameter has high individual influence and high interaction (or total effect). Circles in white at the top left and bottom right corners are cases that have high importance in at least one direction. Bottom left square in white shows the least ideal case where hyperparameters are non-influential, and fixing them at any values within their defined domain will not influence the algorithm's performance. Arrow along the diagonal direction indicates the order of the hyperparameters' importance and influence.

is a non-influential parameter. The large values of interaction effect or total-effect ST_i show proportionally strong interactions between the hyperparameter X_i and its complementary parameter $X_{\sim i}$. The difference $ST_i - S_i \geq 0$, i.e., total interaction influence minus direct influence, shows how much ith hyperparameter is involved in interaction with other hyperparameters. We normalized the values of S_i and ST_i between 0 and 1 for lucid interpretation of their influence (see Fig. 2).

The first-order effect S_i and total effect ST_i of Sobol method are computed as

$$S_i = \frac{V(E(Y|X_i))}{V(Y)} = \frac{y_A \cdot y_{C_i} - f_0^2}{y_A \cdot y_A - f_0^2} = \frac{\frac{1}{N}\sum_{j=1}^{N} y_A^j y_{C_i}^j - f_0^2}{\frac{1}{N}\sum_{j=1}^{N} \left(y_A^j\right)^2 - f_0^2} \tag{10}$$

and

$$ST_i = 1 - \frac{V(E(Y|X_{\sim i}))}{V(Y)} = 1 - \frac{y_B \cdot y_{C_i} - f_0^2}{y_A \cdot y_A - f_0^2} = 1 - \frac{\frac{1}{N}\sum_{j=1}^{N} y_B^j y_{C_i}^j - f_0^2}{\frac{1}{N}\sum_{j=1}^{N} \left(y_A^j\right)^2 - f_0^2}, \tag{11}$$

where N is the number of random samples, $y_A = f(A)$, $y_B = f(B)$ and $y_{C_i} = f(C_i)$ are model output vectors on sample matrix A, B and C_i respectively; and the estimated mean f_0^2 is

$$f_0^2 = \left(\frac{1}{N}\sum_{j=1}^{N} y_A^j\right)^2. \tag{12}$$

Matrices $A_{N\times k}$ and $B_{N\times(k-2k)}$ are random sample points (hyperparameter values), and each matrix C_i is formed by taking all columns of matrix B except ith column, which is taken from ith column of matrix A. Such a sampling is similar to OAT sampling, except its rows are not sorted in any specific order, and all elements in a row differ from the other elements in the row.

5. Experiments

Our sensitivity analysis framework has four essential structural components:

1. setup of EAs tunable hyperparameters and optimization problems
2. sampling of hyperparameters from hyperparameter space of respective sensitivity analysis methods for respective algorithms
3. evaluation of EAs on optimization problem (testbench) for all sampled hyperparameter points and for each hyperparameter sample, the evaluation of respective EAs over a number of independent instances to obtain stable results and to observe expected (average-case) performance of algorithms over performance measures
4. computation of Morris and Sobol indices

In the experiment, all EAs start with a population of initial candidate solutions (uniformly randomly drawn from $\mathbb{R}^n$, n being dimensionality of the problem). Other commonalities among EAs are evolutionary operators like "selection," "mutation," and/or "crossover" for generating new (offspring) population and their evaluation. EAs repeat this process for a number of generations until a *termination condition* is met. We set the *termination condition* to be the desired number of *function evaluations*, and we set this to a value of 10000 for all four algorithms for all problems. The other hyperparameters setting for our experiments were as follows:

5.1. Single-objective algorithm hyperparameters

We analyzed two single-objective EAs over 33 optimization problems: 23 problems from testbench introduced in Yao et al. [42], and we took 10 optimization problems regarding shifted problems from CEC2014 (shifted Sphere, Ellipsoid, Ackley, and Griewank; and shifted and rotated Rosenbrock, and Rastrigin) and CEC 2015 (shifted and rotated Weierstrass, Schwefel, Katsuura, HappyCat) [43,44,72]. An EA needs to find a single optimal solution for an SOO problem in a few generations at the expanse of some wall-clock time. Hence, the *Best Solution* was used for SOO evaluation. Table 2 lists the hyperparameter tuning space of CMA-ES [11] and DE [12] algorithms.

The sensitivity analysis method setup for single-objective optimization was as follows. We used $p = 10$ grid levels to form the hyperparameter space Ω for respective single objective EAs. From this hyperparameter space, we select $r = 50$ sample points for each hyperparameter of CMA-ES and DE in the cases of *Morris LHS* and *Morris* methods (see Eqs. (8) and (9)). This gave us 300 and 400 sample points in total for CMA-ES and DE algorithms, respectively. The Sobol analysis is $2 + k$ times more expensive than Morris methods since it evaluates hyperparameter matrices A, B, and C_i, $i = 1, 2, \ldots, k$. For Sobol, we use $N = 100$, which gave us 700 and 900 sample points in total for CMA-ES and DE algorithms, respectively.

5.2. Multi-objective algorithm hyperparameters

We analyzed multi-objective EAs over a testbench consisting of four families of optimization problems: (i) DTLZ1, DTLZ2, DTLZ3, and DTLZ4 [73]; (ii) IDTLZ1 and IDTLZ2 [73]; (iii) CDTLZ2 [13]; and (iv) WFG3, WFG6, and WFG7 [74]. EAs were evaluated and analyzed for each listed MOO problem for 3 objectives, and each problem was solved as a 10-dimensional problem. This setting was chosen based on the computation effort required for these MOO problems.

Since the goal of the multi-objective EAs is to obtain a set of solutions where no one objective dominates over the other objectives [14,64], we

use GD (minimization), IGD (minimization), and HV (maximization) as the measures of EA performances (see Section 3.3.2). These metrics result in higher values for a large population size λ compared to a small population size λ. Thus, for *population-fair* performance analysis, the metrics were calculated from a union of populations of all generations of EAs and from not only the population of the last generation of the EAs. Moreover, the values were averaged over 30 independent runs for each sampled set of hyperparameters.

NSGA-III and MOEA/D have a few common tunable hyperparameters in addition to their subjective tunable hyperparameters. Table 3 shows the domain setting of these common and subjective tunable hyperparameters of NSGA-III and MOEA/D.

The sensitivity analysis method setup for multi-objective optimization was as follows. We used $p = 10$ grid levels to form the hyperparameter space Ω for respective single objective EAs. From this hyperparameter space, we select $r = 20$ sample points for each hyperparameter of CMA-ES and DE in the cases of *Morris LHS* and *Morris* methods (see Eqs. (8) and (9)). This gave us 140 and 160 sample points in total for NSGA-III and MOEA/D algorithms, respectively. In the Sobol analysis, we used $N = 30$, and this gave us 240 and 270 sample points in total for NSGA-III and MOEA/D algorithms, respectively, for their matrices A and B from which C_i matrices were created. The number of sampling points in this work is sufficiently large for good sensitivity analysis [69–71].

All algorithms, methods, and sensitivity analysis experiments were performed in MATLAB, and implementations of individual components were taken from MATLAB libraries. We used a *safe toolbox* [75] to implement sensitivity analysis sampling methods, indices calculations, and workflows. Single objective algorithms were implemented using ypea library [76]. We used the implementation of multi-objective optimization problems and evaluation measure metrics related to optimization algorithms from PlatEMO library [77]. The entire workflow framework was synchronized with the help of inbuilt functions of MATLAB.

The whole experiment was expensive to run since the total number of function evaluations was 19,014,600,000. The breakdown of this function evaluation was as follows (each multiplied by 10,000 concerning termination condition): DE, 858,580; CMA-ES, 720,080; MOEA/D, 171,600; and NSGA-III, 151,200. For DE and CMA-ES, there were 33 objective functions, and each one was run at least 10 times for each combination of hyperparameter settings. Similarly, for MOEA/D and NSGA-III, there were 10 functions, and each was run 30 times for stable results for each set. The hyperparameter sets were sampled in three different ways for all algorithms: Morris LHS, Morris, and Sobol, as mentioned in Sections 5.1 and 5.2. Our implementation of this framework for sensitivity analysis of EAs and results are available in Ojha et al. [78].

Table 2
Hyperparameter domain range of CMA-ES [11] and DE [12]. For both algorithms, the termination condition was 10,000 function evaluations.

Algo	Params	Domain	Description
CMA-ES	λ	$[10, 1000]$	Population size
	α_μ	$[0, 4]$	Learning rate
	σ_0	$[0.1, 2]$	Initial step size
	$\sigma_{0-scale}$	{False, True}	Re-scaling of σ_0: convergence speed controller
	$\mu\lambda_{\text{ratio}}$	$[0.1, 1]$	Percentage of population's elements usage in co-variance matrix estimation and update
DE	λ	$[10, 1000]$	Population size
	X	{bin, exp	Crossover methods: Binomial and Exponential
	P[X]	$[0, 1]$	Crossover probability
	$\beta_{\min}$	$[0, 1]$	Minimum Acceleration coefficient
	$\beta_{\max}$	$[0, 2]$	Maximum Acceleration coefficient, $\beta_{\max} = \beta_{\min} + \beta_{\max}$
	$\mathbf{b}_{\text{type}}$	{"best," "target-to-best," "rand-to-best," "rand"}	Base vector selection methods (mutation type or DE algorithm version)
	$\mathbf{b}\lambda_{\text{ratio}}$	$[0.01, 0.5]$	Percentage of base vectors (solution) to be used for difference vectors computation

Table 3
Hyperparameter domain range of NSGA-III and MOEA/D and their shared (*Common*) hyperparameters domain. For both algorithms, the termination condition was 10,000 function evaluations.

Algo	Params	Domain	Description
Common	λ	$[10, 1000]$	Population size.
	P[X]	$[0, 1]$	Simulated binary crossover (SBX) probability
	X_{DI}	$[1, 200]$	SBX distribution index
	P[PM]	$[0, 1]$	Polynomial mutation (PM) probability
	PM_{DI}	$[1, 200]$	PM distribution index
NSGA-III	K	$[2, 10]$	Tournament size
	Selection	Tournament	Parents selection for offspring generation
MOEA/D	*Mode*	{"penalty based boundary intersection (PBI)," "Tchebycheff," "Tchebycheff with normalization," "modified Tchebycheff"}	Method for MOO decomposition into many SOO subproblems
	ϵ_N	$[0.05, 0.5]$	Neighbors: percentage of the population considered as neighbors for each subproblem generation

6. Results and discussion

The results of sensitivity analysis of each algorithm for their performances on testbench were collected and processed to produce three indicators: (i) sensitivity analysis indices matrix as per Fig. 2, (ii) ordered bar plot arranged from low to high normalized sensitivity analysis total indices values, and (iii) mean score (average performance) of each hyperparameter over select performance measures. Additionally, the statistical tests and clustering analyses results are presented in supplementary Sections A and B. This section describes hyperparameter *influence, ranking*, and *quality* through these three indicators.

Each sensitivity analysis method varies how they sample hyperparameter sets as they use strategies such as LHS, OAT based uniform random walk, and OAT based uniform sampling. Morris LHS and Sobol use the LHS strategy, which means they stratified the hyperspace to draw samples to cover most of the sample space. Morris uses uniform random walk sampling. In summary, each method may present its own ordering of hyperparameters that influence ranking and interpretation. Hence, we are also interested in the commonality of results among methods.

6.1. Single-objective EAs

6.1.1. CMA-ES analysis

CMA-ES results are shown in Figs. 3–5, where Fig. 3 is a scatter plot that presents sensitivity analysis indices as per Fig. 2. It shows the tendency of the *quality of influence* a hyperparameter has on CMA-ES performance on all 33 problems in the testbench. For instance, λ, the population size in CMA-ES has a high *overall* influence and high *interaction* influence in all three sensitivity analysis methods. Hence, λ is the most significant hyperparameter of the CMA-ES algorithm, and this must be the first hyperparameter one must select to tune for the performance improvement when CMA-ES is applied to solve a problem.

Population size λ Population size λ is the most influential factor in CMA-ES algorithms. Both Morris and Sobol methods show a strong overall influence and high interaction for λ. Morris LHS ranked it as a

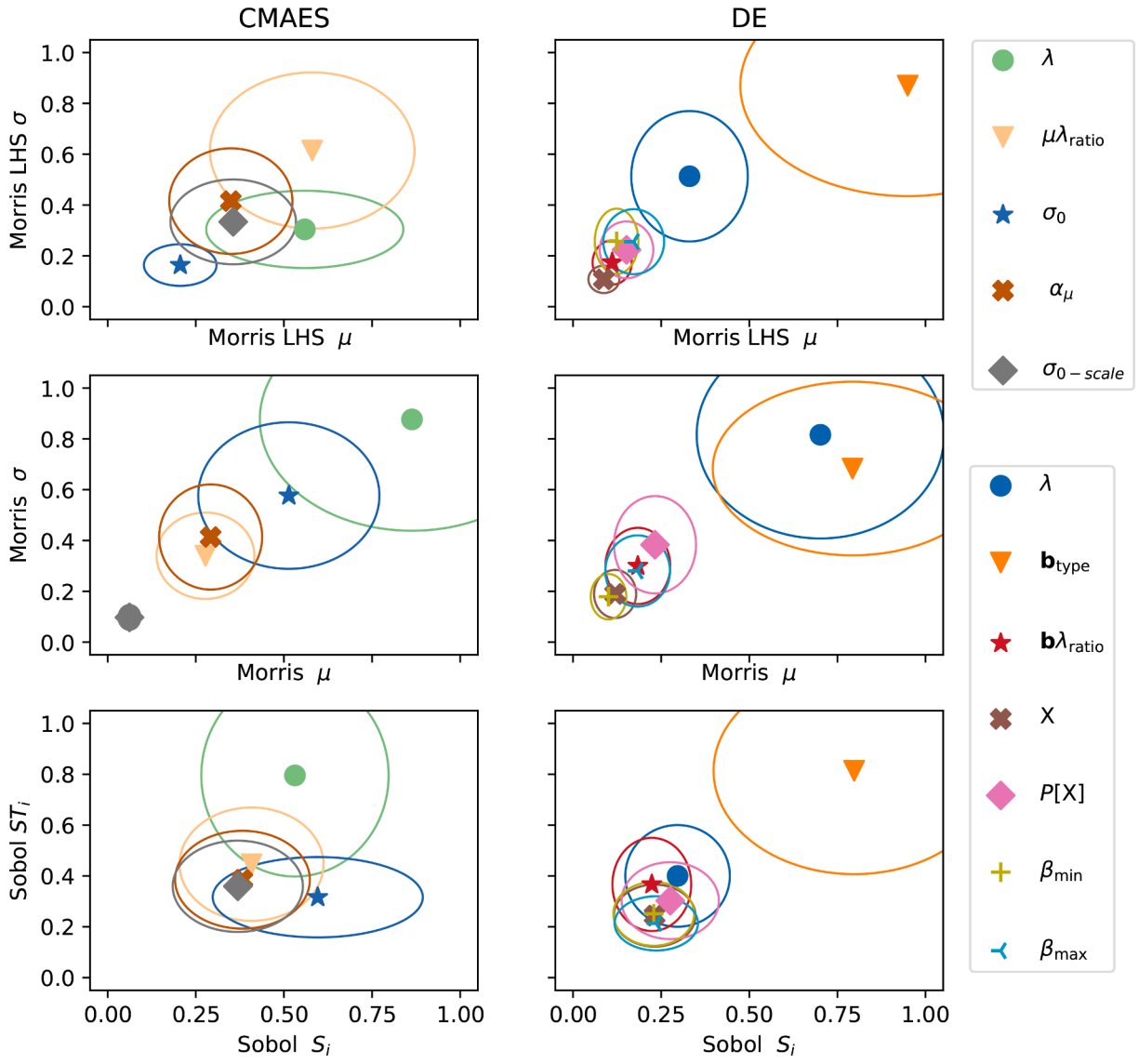

Fig. 3. Single objective algorithms sensitivity analysis. CMA-ES and DE hyperparameters sensitivity analysis are shown in column 1 and column 2, respectively. Rows 1, 2, and 3, respectively, indicate Morris LHS, Morris, and Sobol methods. The upper right legend belongs to CMA-ES and the lower right to DE. A symbol and a color represent each hyperparameter. An eclipse centered at a hyperparameter is the span of the standard deviation of the influence along with direct and interaction influences. A larger width of the eclipse of a hyperparameter in the *x*-axis direction means more variation in direct dominance of that hyperparameter, and a larger height in the *y*-axis direction means its variation in total (or interaction) influence. In each plot, the further apart a hyperparameter in the diagonal direction from the origin (0,0) is, the higher its importance to the algorithm. CMA-ES hyperparameter λ, $\mu\lambda_{ratio}$, σ_0, α_μ, and $\sigma_{0-scale}$ respectively are population size, percentage of the population for covariance matrix, initial step size, learning rate, and convergence speed controller. DE hyperparameters λ, $\mathbf{b}_{type}$, $\mathbf{b}\lambda_{ratio}$, X, $P[X]$, β_{min}, and β_{max} respectively are population size, base vector selection type (mutation type), percentage of the population for base vector selection, crossover-type, crossover probability, minimum acceleration coefficient, and maximum acceleration coefficient. Table 2 contains the hyperparameter name, definition and domain information. Supporting statistical tests [79] between direct and interaction effects and clustering analysis are provided in supplementary Sections A and B.

high direct influence but slightly lower interaction influence than covariance matrix size controller $\mu\lambda_{ratio}$ that has the highest interaction and direct influence in the Morris LHS method. Since MOEA/D decomposes problems into several single-objective problems, unsurprisingly, the size of the population and related hyperparameters are the most influential. This corroborates the fact that they offer exploration capabilities to population-based algorithms, allowing them to search a huge part of the search space concurrently. Figs. 4 and 5 confirm the significance of λ in CMA-ES. Fig. 5 also suggests that variation in CMA-ES performance is very high due to this interaction of population size with other hyperparameters as we observe a highly fluctuating performance of CMA-EA for varied λ values.

Covariance matrix size controller $\mu\lambda_{ratio}$ Hyperparameter $\mu\lambda_{ratio}$, which controls the percentage of population λ to be used for the covariance matrix estimation and update, has high interaction and direct influence on CMA-ES performance. The $\mu\lambda_{ratio}$ is the second most influential hyperparameter across all three methods (see Fig. 4). The significance of $\mu\lambda_{ratio}$ is evident as its values and the choice of λ are closely linked, and the choice of this ratio will increase or decrease the size of the covariance matrix that is at the core of the CMA-ES algorithm functioning. Similar to the performance of λ, $\mu\lambda_{ratio}$ performance is largely variable for its values (see Fig. 5).

Initial step size σ_0 Fig. 4 confirms the significance of σ_0 (initial step size) influence as this emerged as the next best hyperparameter in Morris and Sobol methods. Morris LHS, which is a stratified sampling method that covers the most hyperspace region, suggests that σ_0 is taken from most regions of its possible values and the CMA-ES performance had varied because of such sampling. However, the scores remain relatively

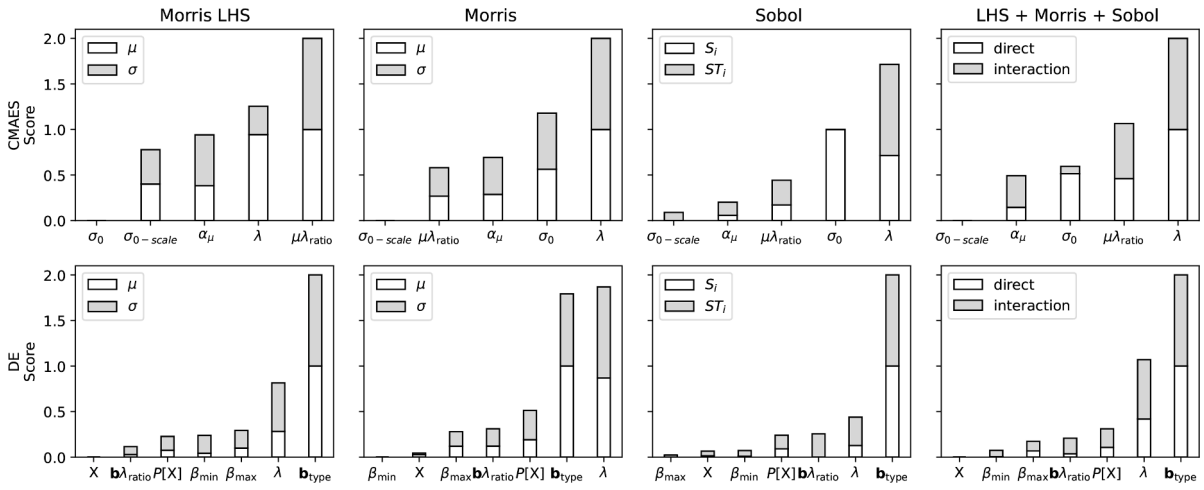

Fig. 4. Ordering (small to larger) of the sum of sensitivity analysis indices of single objective algorithms. CMA-ES (row 1) and DE (row 2) algorithms hyperparameters performance across all problems (functions). Columns 1, 2, and 3 respectively show performance evaluated using Morris LHS, Morris, and Sobol methods. The white color portion of a bar is the direct influence normalized value in $[0, 1]$ and gray color portion is interaction (total) influence value in $[0,1]$. Larger height bar implies a higher influence. Table 2 contains the hyperparameter name, definition and domain information.

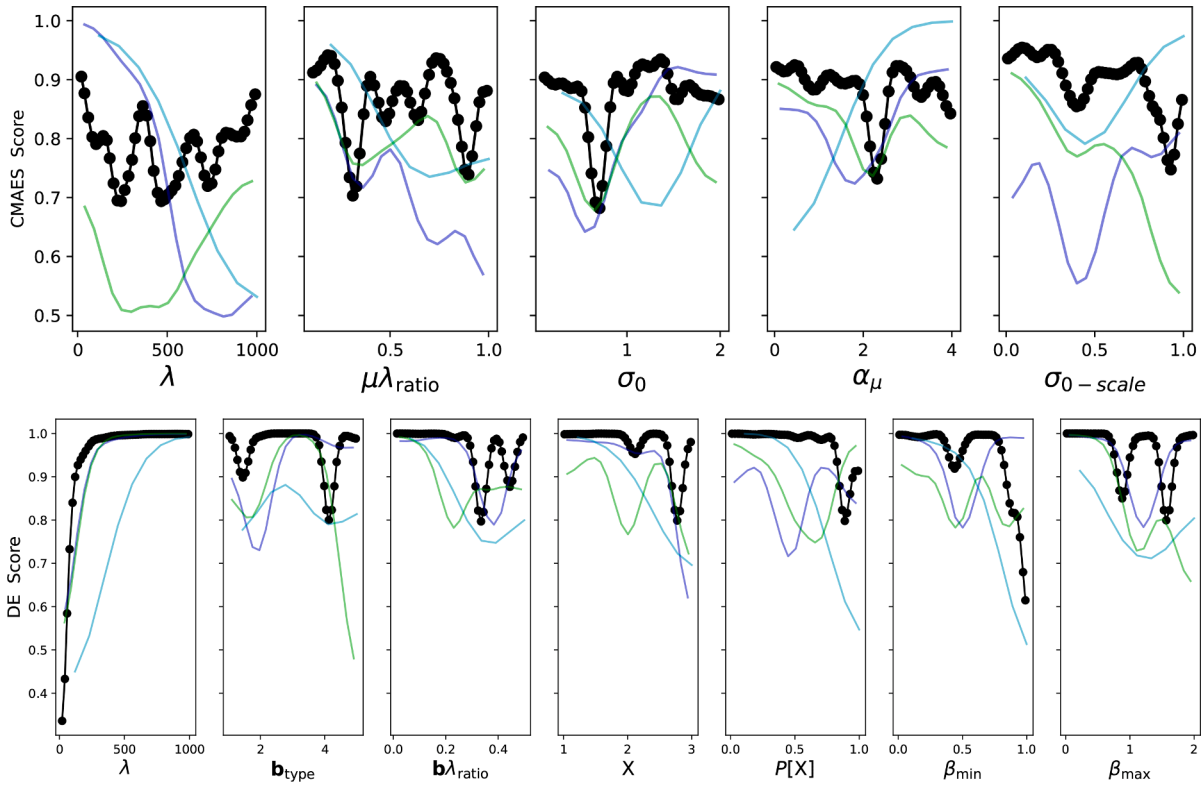

Fig. 5. CMA-ES and DE algorithms average performance on 10 runs (for 72 cases, 30 runs) of each hyperparameter set. CMA-ES and DE algorithms had 2740 and 2,980 hyperparameter sample sets evaluated in total (black dots) by Morris LHS (blue lines), Morris (cyan lines), Sobol (green lines) methods. Each dot in a subplot is a mean performance of a bin of total samples. Along *x*-axis there are 50 bins from lower to higher values which are plotted against each hyperparameter normalized score filtered (using Gaussian filter with sigma 2) in the *y*-axis. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

high (see Fig. 5). The performance σ_0 in Morris LHS is also impacted by the fact that for almost half of the time, its re-scaling was switched off by $\sigma_{0-scale}$. Accordingly, $\sigma_{0-scale}$ should have a higher influence on Morris LHS than σ_0, which indeed is the case (see Fig. 4). Examining Fig. 5, we may observe that for range [1,2] of σ_0 values, CMA-ES mean performances were largely consistent (or above certain high scores). More precisely, a range $[0.8, 1.5)$ σ_0 produces the best performance.

Learning-rate α_μ Learning-rate α_μ was found to be non-influential. However, since the performance of CMA-ES was consistent with its chosen values across all three methods, the learning-rate α_μ was better than re-scaling $\sigma_{0-scale}$. Moreover, the learning-rate α_μ shows more interaction with other hyperparameters than the convergence speed controller $\sigma_{0-scale}$. This is also evident as the gray bars are larger than the white bars in Fig. 4 and drop in performance for only a very small range of values around 2 in Fig. 5).

Convergence controller $\sigma_{0-scale}$ CMA-ES convergence controller

hyperparameter $\sigma_{0-scale}$, a hyperparameter meant for re-scaling of initial step size σ_0 on and off, is the least influential in both Morris and Sobol methods (see Fig. 3). This result is supported by both Figs. 4 and 5. However, it is an influential hyperparameter in the sense that it has a very high influence on σ_0, which is the third most influential hyperparameter. From Fig. 5, it is evident that no re-scaling of σ_0 performs better than re-scaled σ_0. This is the reason why for Morris LHS, σ_0 is the least influence hyperparameter.

CMA-ES hyperparameters ranking In summary, we may provide a *ranking of hyperparameters* for CMA-ES from the most to least influential hyperparameter as λ, $\mu\lambda_{\text{ratio}}$, σ_0, α_μ, and $\sigma_{0-scale}$. One may ignore tuning α_μ and $\sigma_{0-scale}$ completely as setting a sufficiently large function evaluation number would neutralize their importance in the CMA-ES algorithm.

6.1.2. DE analysis

DE versions $\mathbf{b}_{\text{type}}$ DE results are shown in Figs. 3–5, where Fig. 3 shows the quality of influence each hyperparameter has on DE performance over 33 problems. As per the result of the three sensitivity analysis methods, it is clear that the type of DE base vector selection method (mutation type) $\mathbf{b}_{\text{type}}$ (DE version) is by far the most significant hyperparameter. Examining Fig. 5, we observe that the type of DE mutation strategy *best* and *rand* have lower scores, and they fluctuate. In comparison, the base vector selection methods *target-to-best* and *best-to-best* performed more consistently with high scores. Therefore, the average performance of DE over the testbench was highly sensitive to the selection of $\mathbf{b}_{\text{type}}$. We also observe that $\mathbf{b}_{\text{type}}$ in Fig. 3 remains at (1,1) corner of the plot, meaning it had both a high *direct effect* and high *interaction effect*.

Population size λ Overall population size λ is the second most influential hyperparameter in DE (cf. Figs. 3 and 4). Comparatively, it had produced better scores for larger population sizes than the smaller population sizes (see Fig. 5). However, the size of the population of DE was a distant second influential hyperparameter. This indicates that except for small population size (less than 200), DE's performance was invariable when the population size was increased from 200 to 1000 (Fig. 5). This was when the number of function evaluations was the same for each population size, i.e., the number of function evaluations was 10,000 for each population size.

Crossover-type X *and crossover probability* $P[\text{X}]$ The crossover related hyperparameters are the type of crossover X and the probability of crossover $P[\text{X}]$. Between these, $P[\text{X}]$ plays a vital role in DE's performance, and the type of DE $\mathbf{b}_{\text{type}}$ was the least influential (cf. Figs. 3 and 4). For crossover-type *binomial* offered better scores than the crossover-type *exponential* (see Fig. 5). The crossover probability $P[\text{X}]$ has its usage only for binomial crossover. Hence, it was an influential hyperparameter in this setting.

Base vectors selection pool $\mathbf{b}\lambda_{\text{ratio}}$ The hyperparameter $\mathbf{b}\lambda_{\text{ratio}}$ defines the percentage of the population used for the selection of base vectors for DE. We found that $\mathbf{b}\lambda_{\text{ratio}}$ has a negligible influence on the performance of DE (cf. Figs. 3 and 4).

Acceleration coefficients $\beta_{\min}$ *and* $\beta_{\max}$ Similar to $\mathbf{b}\lambda_{\text{ratio}}$, acceleration coefficients hyperparameters $\beta_{\min}$ and $\beta_{\max}$ are least significant in DE. However, setting an appropriate range is vital for DE performance, as we observed in Fig. 5. This is evident because $\beta_{\min}$ and $\beta_{\max}$ acquire a relatively moderate influence in Morris LHS methods (see Fig. 4). Since the Morris LHS method uses a stratified sampling approach, it forced the selection of $\beta_{\min}$ and $\beta_{\max}$ values across their whole range and the performance of DE is impacted negatively by the higher values of $\beta_{\min}$ and $\beta_{\max}$. Examining Fig. 5, we observed that $\beta_{\min}$ scores for values in $[0.0, 0.5]$ performed consistently with better scores than the values in $(0.5, 1.0]$. However, Morris and Sobol had a uniform distribution and show that the influence of this hyperparameter is non-influential; therefore, setting these values somewhere in $[0.0, 0.5]$ will suffice, and one may not need to exhaustively tune this hyperparameter.

Similarly, $\beta_{\max}$ was found sensitive to its range selection. Fig. 5 offers us the ways to investigate which range had a positive influence and which had a negative. We observe that the lower values had higher scores than the larger values of $\beta_{\max}$ (see Fig. 5). Investigating closely, we found that scores in $[0.2, 0.9]$ are by far better than the scores for other values. This means tuning $\beta_{\max}$ values within range $[0.2, 0.9]$ for a problem is an appropriate strategy.

DE hyperparameters ranking In summary, we *rank* DE hyperparameters from the most significant to least significant as $\mathbf{b}_{\text{type}}$, λ, $P[\text{X}]$, $\mathbf{b}\lambda_{\text{ratio}}$, $\beta_{\max}$, $\beta_{\min}$, and X. That one would safely use DE with binomial crossover and set appropriate values (discussed above) of $\beta_{\max}$, $\beta_{\min}$.

6.1.3. Remarks on SOO hyperparameter rankings and algorithms

We evaluated two single objective optimization algorithms and presented rankings of their hyperparameter influence based on a combined assessment of three sensitivity analysis methods. Each method, as mentioned, has its own way of drawing samples from the hyperparameter space and thus has produced its own ranking. However, the results reveal some obvious signs of influence based on direct and interaction effects.

Supplementary A provides rich information on statistical tests among hyperparameters that one can thoroughly examine to reach the presented ranking and may find more information should one is interested in studying specific hyperparameters. For instance, the interaction effect of population size in CMA-ES is more significant than its direct effect (see Supplementary A), which confirms the analysis presented in Fig. 5. Additionally, clustering analysis of hyperparameters and objective function provides information behaviors of the algorithm on the class of problems they solve (see supplementary B). For example, the type of mutation in DE forms a distinct cluster of its performance characteristics, and all other hyperparameters are grouped together in one cluster (see supplementary B).

As a consequence of this analysis and the results presented in Section 6.1, it is clear that *DE is a more stable algorithm than CMA-ES*. See variation in scores of the hyperparameters of the CMA-ES algorithm compared to DE's hyperparameters in Fig. 5 and high interaction among CMA-ES's hyperparameters. In contrast, DE has a clear ranking of hyperparameters. Additionally, during the experiments, CMA-ES failed to solve some classes of problems for some combination of hyperparameters (see results in [78]).

6.2. Multi-objective EAs

6.2.1. NSGA-III analysis

Population size λ Results of NSGA-III are presented in Figs. 6–8. In Fig. 6, we present results of three measures GD, IGD, and HV; see columns in Fig. 6, and along rows in Fig. 6, we present Morris LHS, Morris, and Sobol sensitivity methods. For NSGA-III, we clearly observe that the population size λ is a significant hyperparameter, and the probability of crossover is the second most significant hyperparameter. Population size influence has approximately equal high direct influence and high interaction influence. That is, although population size is the most significant hyperparameter, NSGA-III performance varied because of the variation of the other hyperparameters as well (see NSGA-III has a monotonous line for λ in Fig. 8 that indicates a more liner influence on NSGA-III). This fact was found true across all methods and all measures as the eclipse of its influence centered around coroner (1,1) in Fig. 6, and the white and gray bars have comparable lengths in Fig. 8.

An examination of scores of the population size shows that population size does not fluctuate much for the HV metric after a certain population size, but for GD and IGD metrics, the scores keep increasing for increasing population size (see Fig. 8). However, this is monotonous, and one would expect such performance for GD and IGD metrics. The probability of crossover shows more fluctuations in all three metrics. Therefore, the variations in the performance of NSGA-III after a sufficiently large population size (in this case, 200) come from the variations of other hyperparameters, including crossover probability.

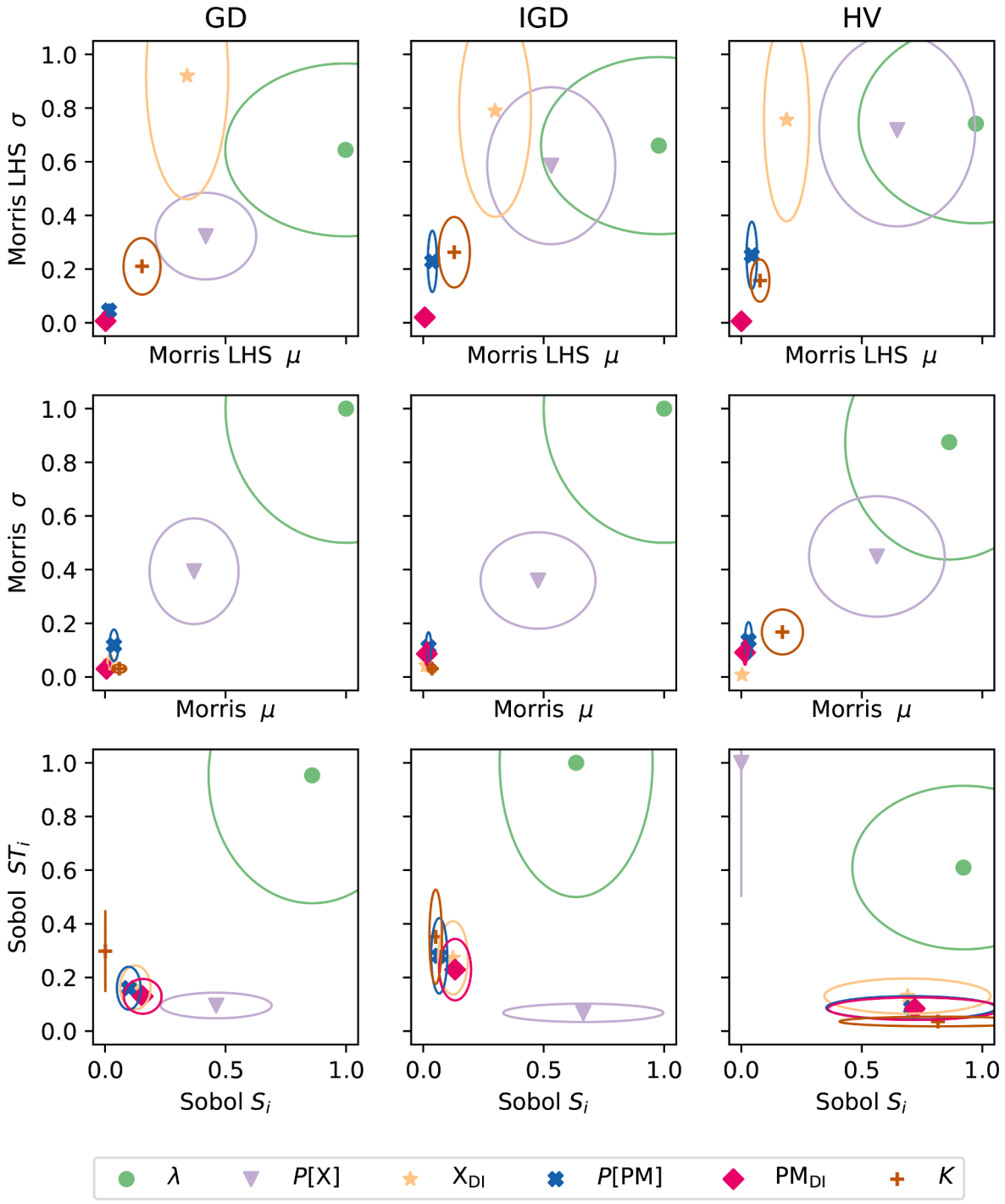

Fig. 6. NSGA-III hyperparameters sensitivity analysis. Columns 1, 2, and 3 respectively indicate performance metrics GD, IGD, and HV. Rows 1, 2, and 3, respectively, indicate Morris LHS, Morris, and Sobol methods. Legends of hyperparameter are shown at the bottom. Each hyperparameter is represented by a symbol and a color. An eclipse centered at a hyperparameter is the standard deviation of its influence and direction of its influence. The further apart a hyperparameter in the diagonal direction from the origin (0,0) is, the higher its importance to the algorithm. A larger width of the eclipse of a hyperparameter in the *x*-axis direction means more variation in the direct influence of a hyperparameter, and a larger height in the *y*-axis direction means variation in total (or interaction) influence. Supporting statistical tests and clustering analysis are provided in supplementary Sections A and B.

Crossover and mutation hyperparameters The probability of crossover *P*[X] shows a more linear relationship between its values and NSGA-III performance measures GD, IGD, and HV. For increasing values of crossover probability, we see decreasing GD and IGD scores (signs of better performance) and increasing scores of HV for some values (see Fig. 8). A crossover rate of around 0.6 leads to better solutions along the problem's objective dimensions, i.e., increasing scores of HV and lower scores of GD and IGD. This fact is supported by the strong direct and interaction influence of crossover *P*[X] for IGD and HV metrics and relatively direct influence on GD. The Sobol method on *P*[X] does show a very strong total influence compared to direct influence on all metrics. In summary, the *P*[X] performance has a behavior of monotonous increase and is one of the most influential hyperparameters in NSGA-III.

For crossover related hyperparameter crossover distribution indices

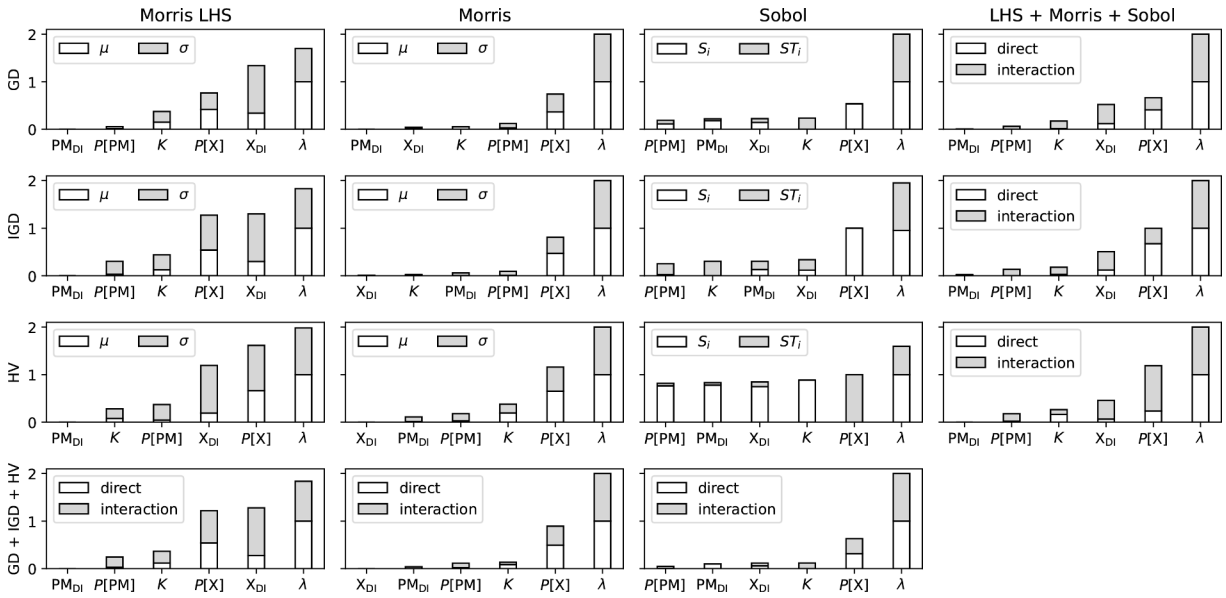

Fig. 7. NSGA-III algorithm's hyperparameters performance across all problems (functions). Rows 1, 2, and 3 respectively, show performance evaluated using GD, IGD, and HV metrics. Columns 1, 2, and 3, respectively indicate Morris LHS, Morris, and Sobol methods. The white color portion of a bar is direct influence normalized value in $[0,1]$ and gray color portion is interaction (total) influence value in $[0,1]$. A larger height bar implies a higher influence, and hyperparameters in each subplot are arranged from low to high influence.

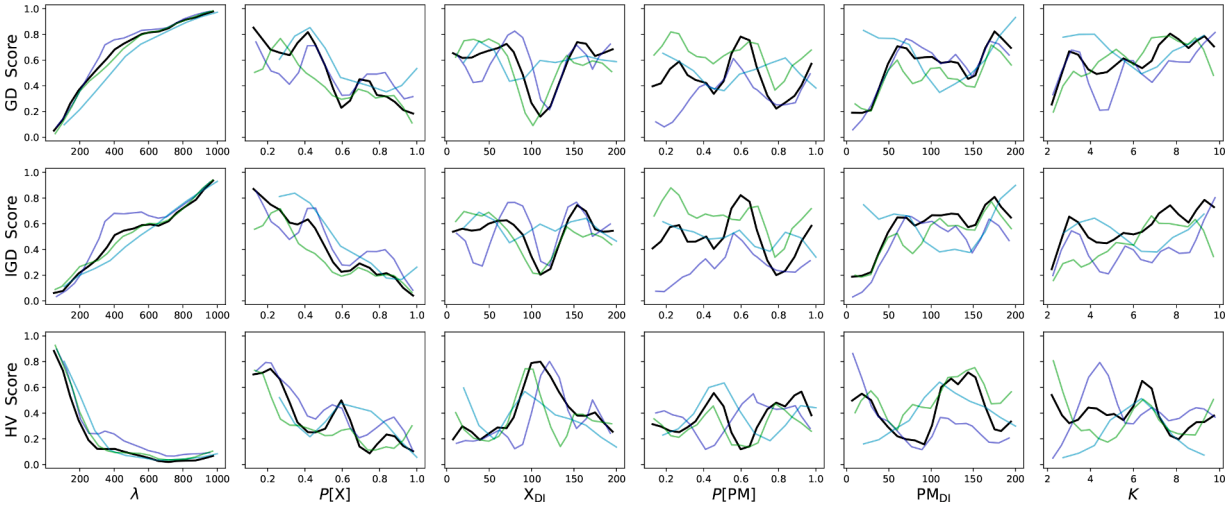

Fig. 8. NSGA-III algorithm average performance on 30 runs of each set of hyperparameters. NSGA-III hyperparameter (x-axis) against the mean metric value (y-axis). Rows 1, 2, and 3, respectively, are GD, IGD, and HV metrics. The scores are normalized between 0 and 1 and smooth out using a Gaussian 1D filter with sigma 0.99. The y-axis is GD, IGD, and HV metrics values normalized between a score of 0 and 1, where 0 is the best score for GD and IGD, and 1 is the best score for HV. A total of 520 samples were evaluated for the NSGA-III algorithm jointly by Morris LHS (blue lines), Morris (cyan lines), and Sobol (green lines) methods. The hyperparameter values are arranged in 20 bins (lower values to higher values) across the x-axis. Each line in each plot connects the mean values of 20 bins of such samples. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

X_{DI}, the performance remains consistent and largely non-influential (cf. Figs. 6 and 8) as only for a certain range of its value (a small range around 100), it shows a spike in the performance of NSGA-III. Similarly, the mutation distribution index PM_{DI}, the performance of NSGA-III is better for a certain range (around 100–150 or low values of PM_{DI}, see Fig. 8). For both X_{DI} and PM_{DI}, this phenomenon occurred roughly around a value of 100 of these indices, which aligned with the range for these hyperparameters suggested in Deb and Deb [80], Deb et al. [81].

Similar to the probability of crossover P[X], the probability of mutation P[PM] shows a sudden change in performance around a value of 0.6, but in a complementary direction (see a drop in HV and spike in GD and IGD metric in Fig. 8). The direct and interaction influence of mutation related hyperparameters P[PM] and PM_{DI} is low for NSGA-III (cf. Figs. 6 and 7).

Tournament size K Tournament size K, probability of polynomial mutation P[PM], polynomial mutation distribution index PM_{DI} and simulated binary crossover (SBX) distribution index X_{DI} have comparable significance. However, they differ in different methods and metrics. Among these hyperparameters, tournament size K clearly shows a high influence on NSGA-III performance. Tournament size K shows more interaction influence than direct influence, except for the HV metric of the Sobol method. The high score of K in Fig. 8 with clear fluctuation is the evidence of its interaction with other hyperparameters, but the scores (especially in GD and IGD scores) show an upward trend, indicating it has comparatively less influence on guiding the population towards true Pareto-front than hyperparameters P[PM], PM_{DI} and X_{DI}.

We may also confirm that the lower value of K is more influential than its higher values.

NSGA-III hyperparameters ranking Considering the hyperparameters' performance influence, we rank them from most influential to least influential hyperparameters as λ, $P[X]$, X_{DI}, K, $P[PM]$, and PM_{DI}. Here, λ is effective up to a certain population size, and then λ saturates. The tuning of crossover $P[X]$ linearly influences NSGA-III, and X_{DI}, $P[PM]$, and PM_{DI} require setting a fixed value, but their influence fluctuates, i.e., they are affected by the setting of values of other hyperparameters a lot.

6.2.2. MOEA/D analysis

Population size λ MOEA/D results are shown in Figs. 9–11. In Fig. 9, the results of three sensitivity analysis methods for three metrics of MOEA/D performance are presented. Unlike NSGA-III results, population size λ is not a clear most significant hyperparameter for MOEA/D multi-objective algorithm. Rather, MOEA/D's hyperparameters *Mode*, the MOO decomposition method, is also among the influential hyperparameters. Morris LHS method shows that the *Mode* is the most significant hyperparameter overall on three metrics. Fig. 11 also confirms this fact as for the population size values, the GD, IGD, and HV metrics

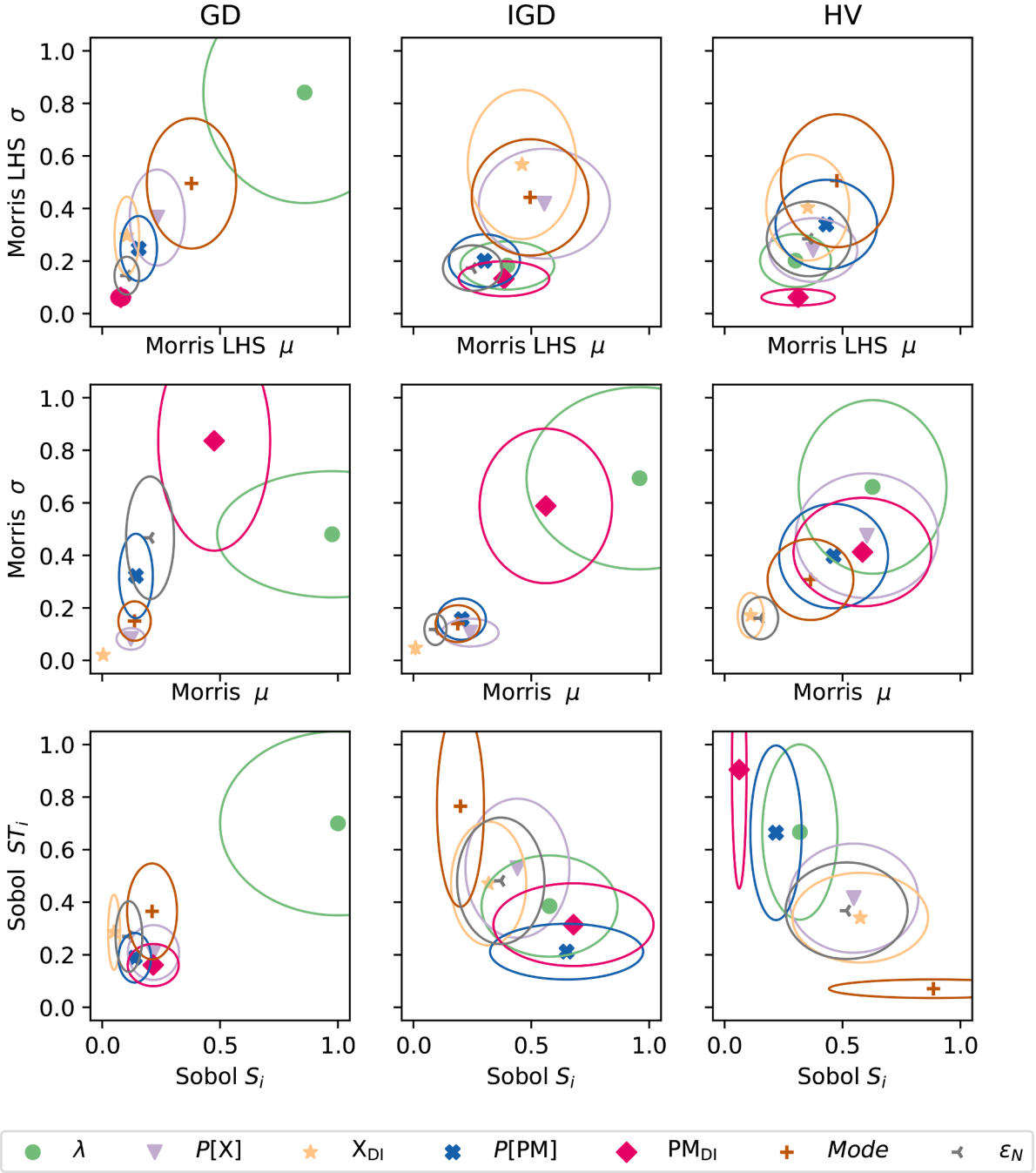

Fig. 9. MOEA/D hyperparameters sensitivity analysis. Columns 1, 2, and 3 respectively indicate performance metrics GD, IGD, and HV. Rows 1, 2, and 3, respectively, indicate Morris LHS, Morris, and Sobol methods. Legends of hyperparameter are shown at the bottom. Each hyperparameter is represented by a symbol and a color. An eclipse centered at a hyperparameter is the standard deviation of its influence and direction of its influence. Further apart a hyperparameter in the diagonal direction from the origin (0,0) is, the higher its importance to the algorithm. A larger width of the eclipse of a hyperparameter in the x-axis direction means more variation in the direct influence of a hyperparameter, and a larger height in the y-axis direction means variation in total (or interaction) influence. Supporting statistical tests and clustering analysis are provided in supplementary Sections A and B.

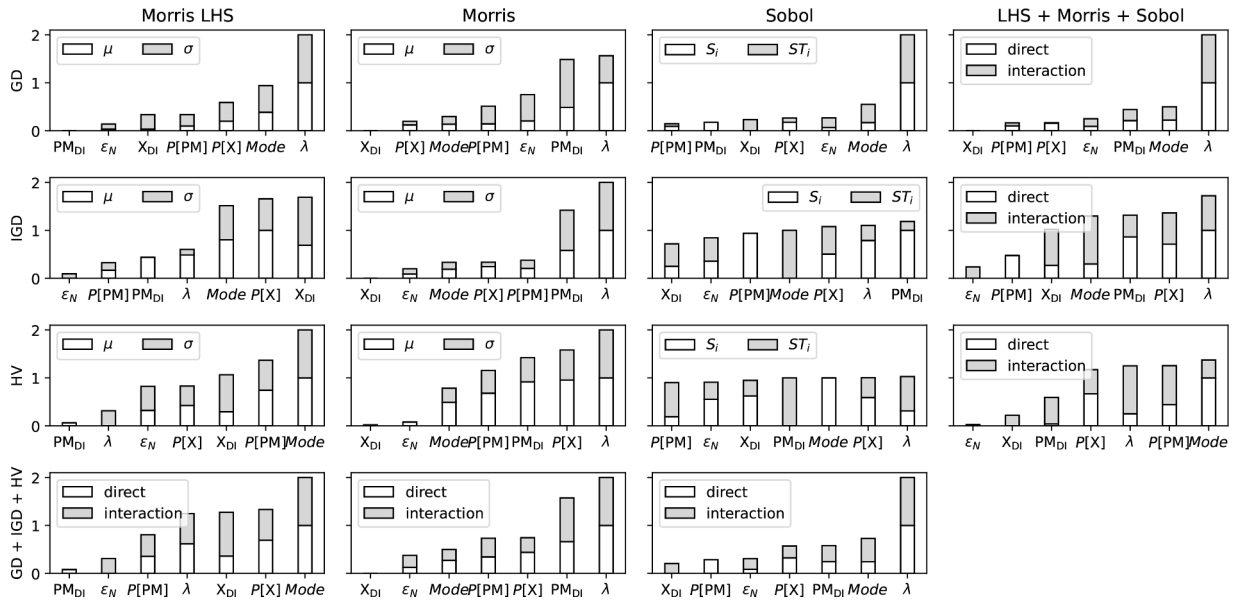

Fig. 10. MOEA/D algorithm's hyperparameters performance across all problems (functions). Rows 1, 2, and 3, respectively, show performance evaluated using GD, IGD, and HV metrics. Columns 1, 2, and 3 respectively indicate metric Morris LHS, Morris, and Sobol methods. The white color portion of a bar is direct influence normalized value in $[0, 1]$ and gray color portion is interaction (total) influence value in $[0, 1]$. A larger height bar implies a higher influence, and hyperparameters in each subplot are arranged from low to high influence.

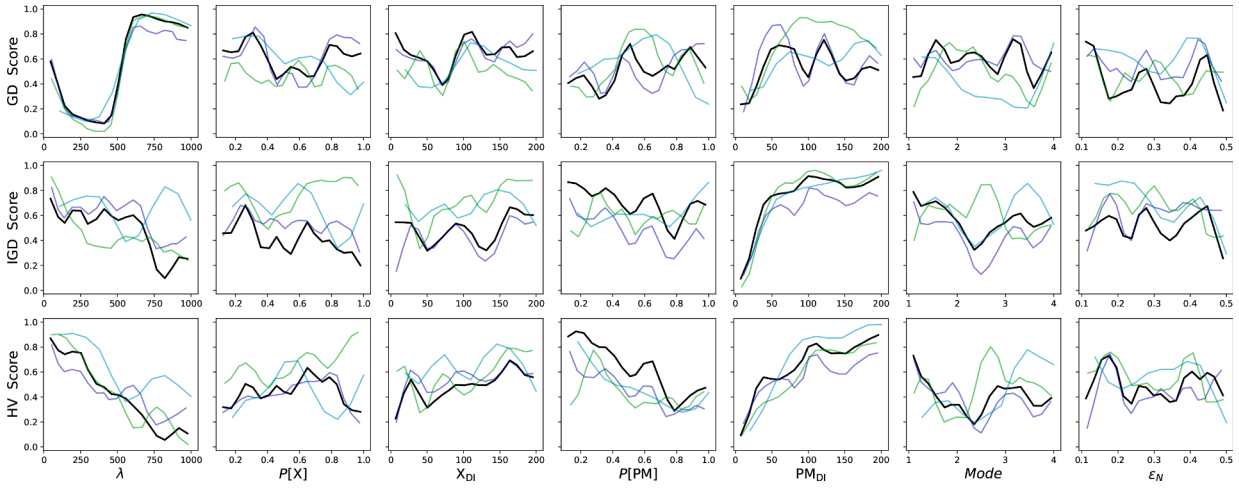

Fig. 11. MOEA/D algorithm average performance on 30 runs of each set of hyperparameters. MOEA/D hyperparameter (x-axis) against the mean metric value (y-axis). Rows 1, 2, and 3, respectively, are GD, IGD, and HV metrics. The scores are normalized between 0 and 1 and smooth out using a Gaussian 1D filter with sigma 0.99. The y-axis is GD, IGD, and HV metrics values normalized between a score of 0 and 1, where 0 is the best score for GD and IGD, and 1 is the best score for HV. A total of 590 samples were evaluated for the MOEA/D algorithm jointly by Morris LHS (blue lines), Morris (cyan lines), Sobol (green lines) methods. The hyperparameter values are arranged in 20 bins (lower values to higher values) across the x-axis. Each line in each plot connects the mean values of 20 bins of such samples. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

show a strong relation.

For example, the HV metric in Fig. 11 shows a linear trend, but it has clear fluctuations in scores. This is because population size has high interaction with other hyperparameters, and tuning population size alone cannot compensate for the role of the other hyperparameters in the performance of MOEA/D on the GD metric. However, for the IGD metric, population size improves the performance of MOEA/D. This shows a highly fluctuating behavior of population size in MOEA/D for varied metrics, i.e., MOEA/D performance has a nonlinear relationship with the population size. This means population size is rather highly involved with interaction with other hyperparameters as the variation in other hyperparameters also influences the performance of MOEA/D.

MOO decomposition type Mode The next set of hyperparameters that we observe as highly influential is *Mode*, as it shows high interaction and high overall influence in Morris LHS, Morris, and Sobol for GD and HV metrics. HV metric for Sobol placed the hyperparameters on the direct influence to high total influence diagonal (see Fig. 9), which suggests that the hyperparameters either have a good high interaction or good overall influence. Hence, the sum of these, presented in Fig. 10, differs only marginally. Sobol rank *Mode* is second in the GD metric as both high interaction and high overall influence and third in the HV metric as it has a high direct influence.

Examining the performance of *Mode* in Fig. 11, we confirm that the type of MOO decomposition "Tchebycheff with normalization" had the best performance, followed by "penalty based boundary intersection (PBI)" and "Tchebycheff" has significantly poor performance and

"modified Tchebycheff," decomposition mode had the worse scores among MOO decomposition methods. MOEA/D hyperparameter ϵ_N refers to the number of neighbors for selecting the percentage of the population for sub-problems selection MOEA/D has an equivalent influence as the probability of mutation distribution index PM_{DI}. However, ϵ_N value less than 0.2 show a sharp improvement in MOEA/D performance.

Crossover and mutation hyperparameters Genetic operator related hyperparameters $P[X]$, X_{DI}, $P[PM]$ and PM_{DI} show varied significance on different metrics on different sensitivity methods. For example, the probability of mutation distribution index PM_{DI} has a high influence on HV metrics (pink diamond and eclipse in Fig. 9) and a high total influence on HV metrics in the Sobol method. The probability of mutation $P[PM]$ is second to PM_{DI} in total influence on HV as per the Sobol method. This suggests that *mutation has a high influence in diversifying the population* in MOEA/D, helping it produces a better Pareto-front. We also observe that $P[PM]$ and PM_{DI} have mirror image like performance (see Fig. 11), which suggests that values of $P[PM]$ around 0.8 and higher values of PM_{DI} are more effective in MOEA/D performance. The probability of crossover $P[X]$ has competing performance in the MOEA/D, and it is similar to performances of mutation related hyperparameters. That is, unlike NSGA-III, the probability of crossover does not outshine the crossover and mutation related hyperparameters.

MOEA/D hyperparameters ranking In summary, the *ranking of hyperparameters* of MOEA/D from the most influential to least influential hyperparameters is λ, *Mode*, PM_{DI}, $P[PM]$, $P[X]$, ϵ_N, and X_{DI}.

6.2.3. Remarks on MOO hyperparameter rankings and algorithms

Providing ranking to hyperparameters for MOO is more challenging than SOO since it uses three distinct sensitivity analysis methods and uses three distinct performance metrics. However, we look for potential agreement between these distinct measures. We observe that the population size λ clearly emerged as the most influential hyperparameter in all three analyses and metrics for NSGA-III, and the probability of crossover was the second most influential. These two hyperparameters significantly dominate all other hyperparameters in NSGA-III. Whereas for MOEA/D, the population size λ dominates only for the GD metric and for Morris analysis. For HV and IGD metric and Morris LHS and Sobol analysis, *Mode* and mutation probability are dominant factors. Unlike NSGA-III, there is no clear, significantly dominant hyperparameter in MOEA/D. Therefore, considering hyperparameters' strong variability and dependency on the type of hyperparameter sampling methods and type of performance metrics, we may confirm that NSGA-III is a more stable algorithm than MOEA/D.

7. Conclusions

We present a framework for systematic and methodological analysis of the effectiveness of the evolutionary algorithm hyperparameters. This analysis results in (i) identifying the pattern of influence each hyperparameter has on the algorithm, (ii) recommending rankings of hyperparameter influence, and (iii) analyzing the stability of algorithms related to hyperparameter sampling and performance metrics. We apply our methodology to state-of-the-art evolutionary algorithms: two single-objective algorithms and two multi-objective algorithms. The single-objective algorithms used are covariance matrix adaptation evolutionary strategy (CMA-ES), differential evolution (DE), and multi-objective algorithms used are non-dominated sorting genetic algorithm III (NSGA-III), and multi-objective evolutionary algorithm based on decomposition (MOEA/D). Our methodology involves two global sensitivity analysis methods, Morris and Sobol. This methodology is computationally heavy, but it produces widely usable and effective recommendations on hyperparameters ranking, being the order in which one can tune EA hyperparameters to achieve high performance. For example, the initial step size, base vector selection type (mutation), probability of crossover, and mode multi-objective problem decomposition were among the most influential hyperparameters of CMA-ES, DE, NSGA-III, and MOEA/D algorithms, respectively. The results show how the hyperparameters interact with one another when they are sampled differently, and different performance measures are used. This framework can further analyze the sensitivity and influence of adaptive and dynamically tuneable hyperparameters for future work. Furthermore, since different hyperparameters sampling methods showed varied ranking, this work can further study the influence of the sampling method or sensitivity of an algorithm or its hyperparameters towards a particular type of sampling.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.swevo.2022.101130

References

[1] K. De Jong, Evolutionary computation: a unified approach. Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, 2016, pp. 185–199.
[2] K. De Jong, Parameter setting in EAs: a 30 year perspective, Stud. Comput. Intell. (SCI) 54 (2007) 1–18.
[3] M. López-Ibáñez, J. Dubois-Lacoste, L.P. Cáceres, M. Birattari, T. Stützle, The irace package: iterated racing for automatic algorithm configuration, Oper. Res. Perspect. 3 (2016) 43–58.
[4] G. Iommazzo, C. d'Ambrosio, A. Frangioni, L. Liberti, Algorithmic configuration by learning and optimization. Cologne-Twente Workshop on Graphs and Combinatorial Optimization, 2019.
[5] M. Crossley, A. Nisbet, M. Amos, Quantifying the impact of parameter tuning on nature-inspired algorithms. The 12th European Conference on Artificial Life, MIT Press, 2013, pp. 925–932.
[6] R. Taylor, V. Ojha, I. Martino, G. Nicosia, Sensitivity analysis for deep learning: ranking hyper-parameter influence. 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2021, pp. 512–516.
[7] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Trans. Evol. Comput. 13 (3) (2009) 526–553.
[8] V.K. Ojha, A. Abraham, V. Snášel, ACO for continuous function optimization: aperformance analysis. 2014 14th International Conference on Intelligent Systems Design and Applications, IEEE, 2014, pp. 145–150.
[9] K. Eggensperger, M. Lindauer, F. Hutter, Pitfalls and best practices in algorithm configuration, J. Artif. Intell. Res. 64 (2019) 861–893.
[10] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2012) 281–305.
[11] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. Proceedings of IEEE International Conference on Evolutionary Computation, 1996, pp. 312–317.
[12] R. Storn, K. Price, Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359, https://doi.org/10.1023/A:1008202821328.
[13] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: solving problems with box constraints, IEEE Trans. Evol. Comput. 18 (4) (2013) 577–601.
[14] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
[15] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, X. Yao, A survey of evolutionary continuous dynamic optimization over two decades—Part B, IEEE Trans. Evol. Comput. 25 (4) (2021) 630–650.
[16] A. Viktorin, R. Senkerik, M. Pluhacek, T. Kadavy, A. Zamuda, Distance based parameter adaptation for success-history based differential evolution, Swarm Evol. Comput. 50 (2019) 100462.
[17] A.P. Piotrowski, J.J. Napiorkowski, Step-by-step improvement of jade and shade-based algorithms: success or failure? Swarm Evol. Comput. 43 (2018) 88–108.
[18] J. Cheng, Z. Pan, H. Liang, Z. Gao, J. Gao, Differential evolution algorithm with fitness and diversity ranking-based mutation operator, Swarm Evol. Comput. 61 (2021) 100816.
[19] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2010) 4–31.
[20] S.M. Islam, S. Das, S. Ghosh, S. Roy, P.N. Suganthan, An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization, IEEE Trans. Syst., Man, Cybern., Part B (Cybernetics) 42 (2) (2011) 482–500.

[21] A. Biswas, S. Das, A. Abraham, S. Dasgupta, Design of fractional-order PIλDμ controllers with an improved differential evolution, Eng. Appl. Artif. Intell. 22 (2) (2009) 343–350.

[22] S. Das, A. Abraham, A. Konar, Automatic clustering using an improved differential evolution algorithm, IEEE Trans. Syst., Man, Cybern. - Part A 38 (1) (2007) 218–237.

[23] S. Das, A. Konar, U.K. Chakraborty, Two improved differential evolution schemes for faster global search. Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, 2005, pp. 991–998.

[24] T. Voß, N. Hansen, C. Igel, Improved step size adaptation for the MO-CMA-ES. Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, ACM, 2010, pp. 487–494.

[25] Q. Zhang, H. Li, D. Maringer, E. Tsang, MOEA/D with NBI-style Tchebycheff approach for portfolio management. IEEE Congress on Evolutionary Computation, IEEE, 2010, pp. 1–8.

[26] Z. Wang, Q. Zhang, A. Zhou, M. Gong, L. Jiao, Adaptive replacement strategies for MOEA/D, IEEE Trans. Cybern. 46 (2) (2015) 474–486.

[27] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, J. Wu, MOEA/D with adaptive weight adjustment, Evol. Comput. 22 (2) (2014) 231–264.

[28] Z. Cui, Y. Chang, J. Zhang, X. Cai, W. Zhang, Improved NSGA-III with selection-and-elimination operator, Swarm Evol. Comput. 49 (2019) 23–33.

[29] J. Yuan, H.-L. Liu, Z. He, A constrained multi-objective evolutionary algorithm using valuable infeasible solutions, Swarm Evol. Comput. 68 (2022) 101020.

[30] J. Liang, K. Qiao, C. Yue, K. Yu, B. Qu, R. Xu, Z. Li, Y. Hu, A clustering-based differential evolution algorithm for solving multimodal multi-objective optimization problems, Swarm Evol. Comput. 60 (2021) 100788.

[31] D. Han, W. Du, X. Wang, W. Du, A surrogate-assisted evolutionary algorithm for expensive many-objective optimization in the refining process, Swarm Evol. Comput. 69 (2022) 100988.

[32] G. Rivera, C.A.C. Coello, L. Cruz-Reyes, E.R. Fernandez, C. Gomez-Santillan, N. Rangel-Valdez, Preference incorporation into many-objective optimization: an ant colony algorithm based on interval outranking, Swarm Evol. Comput. 69 (2022) 101024.

[33] M.D. Morris, Factorial sampling plans for preliminary computational experiments, Technometrics 33 (2) (1991) 161–174.

[34] I.M. Sobol, S. Kucherenko, Global sensitivity indices for nonlinear mathematical models, review, Wilmott Mag. 2005 (2005) 56–61, https://doi.org/10.1002/wilm.42820050114.

[35] B. Iooss, A. Saltelli, Introduction to sensitivity analysis, in: R. Ghanem, D. Higdon, H. Owhadi (Eds.), Handbook of Uncertainty Quantification, Springer, 2016, pp. 1–20, https://doi.org/10.1007/978-3-319-11259-6_31-1.

[36] R. Brooks, M. Semenov, P. Jamieson, Simplifying sirius: sensitivity analysis and development of a meta-model for wheat yield prediction, Eur. J. Agron. 14 (2001) 43–60, https://doi.org/10.1016/S1161-0301(00)00089-7.

[37] P. Conca, G. Stracquadanio, G. Nicosia, Automatic tuning of algorithms through sensitivity minimization, in: P. Pardalos, M. Pavone, G.M. Farinella, V. Cutello (Eds.), Machine Learning, Optimization, and Big Data, Springer, 2015, pp. 14–25.

[38] M.C. Hill, D. Kavetski, M. Clark, M. Ye, M. Arabi, D. Lu, L. Foglia, S. Mehl, Practical use of computationally frugal model analysis methods, Groundwater 54 (2) (2016) 159–170.

[39] D.A.V. Veldhuizen, G.B. Lamont, Evolutionary computation and convergence to a pareto front. Late Breaking Papers at the 1998 Genetic Programming Conference, Stanford University, 1998, pp. 221–228.

[40] D.A.V. Veldhuizen. Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1999. Ph.D. thesis.

[41] E. Zitzler, L. Thiele, Multiobjective optimization using evolutionary algorithmsa comparative case study. International Conference on Parallel Problem Solving from Nature, Springer, 1998, pp. 292–301.

[42] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (2) (1999) 82–102.

[43] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. Technical Report, Zhengzhou University, Zhengzhou China and Nanyang Technological University, Singapore, 2013.

[44] J. Liang, B. Qu, P. Suganthan, Q. Chen, Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Learning-Based Real-Parameter Single Objective optimization. Technical Report, Zhengzhou University, Zhengzhou China and Nanyang Technological University, Singapore, 2014.

[45] C.F. Lima, F.G. Lobo, Parameter-less optimization with the extended compact genetic algorithm and iterated local search. Genetic and Evolutionary Computation Conference, Springer, 2004, pp. 1328–1339.

[46] T. Jansen, K.A.D. Jong, I. Wegener, On the choice of the offspring population size in evolutionary algorithms, Evol. Comput. 13 (4) (2005) 413–440.

[47] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol. Comput. 9 (2) (2001) 159–195.

[48] A. Greco, S.D. Riccio, J. Timmis, G. Nicosia, Assessing algorithm parameter importance using global sensitivity analysis, in: I. Kotsireas, P. Pardalos, K. E. Parsopoulos, D. Souravlias, A. Tsokas (Eds.), Analysis of Experimental Algorithms, Springer, 2019, pp. 392–407, https://doi.org/10.1007/978-3-030-34029-2_26.

[49] A.E. Eiben, Z. Michalewicz, M. Schoenauer, J.E. Smith, Parameter control in evolutionary algorithms. Parameter setting in evolutionary algorithms, Springer, 2007, pp. 19–46.

[50] O. Kramer, Evolutionary self-adaptation: a survey of operators and strategy parameters, Evol. Intell. 3 (2) (2010) 51–65.

[51] P.L. Iglesias, D. Mora, F.J. Martinez, V.S. Fuertes, Study of sensitivity of the parameters of a genetic algorithm for design of water distribution networks, J. Urban Environ. Eng. 1 (2) (2007) 61–69.

[52] F. Pinel, G. Danoy, P. Bouvry, Evolutionary algorithm parameter tuning with sensitivity analysis, in: P. Bouvry, M.A. Kłopotek, F. Leprévost, M. Marciniak, A. Mykowiecka, H. Rybiński (Eds.), Security and Intelligent Information Systems, Springer, 2012, pp. 204–216.

[53] J. Maturana, F. Lardeux, F. Saubion, Autonomous operator management for evolutionary algorithms, J. Heuristics 16 (6) (2010) 881–909.

[54] A. Saltelli, S. Tarantola, F. Campolongo, M. Ratto, Sensitivity Analysis in Practice: A guide To Assessing scientific Models vol. 1, John Wiley & Sons, 2004.

[55] Y. Lou, S.Y. Yuen, G. Chen, Non-revisiting stochastic search revisited: results, perspectives, and future directions, Swarm Evol. Comput. 61 (2021) 100828.

[56] G. Paul, C.L. Müller, I.F. Sbalzarini, Sensitivity analysis from evolutionary algorithm search paths. EVOLVE - A bridge between Probability, Set Oriented Numerics and Evolutionary Computation, in: Studies in Computational Intelligence, Springer, 2011.

[57] A. Saltelli, S. Tarantola, K.-S. Chan, A quantitative model-independent method for global sensitivity analysis of model output, Technometrics 41 (1) (1999) 39–56.

[58] X. He, K. Zhao, X. Chu, AutoML: a survey of the state-of-the-art, Knowledge-Based Syst. 212 (2021) 106622.

[59] V.K. Ojha, A. Abraham, V. Snášel, Simultaneous optimization of neural network weights and active nodes using metaheuristics. 2014 14th International Conference on Hybrid Intelligent Systems, IEEE, 2014, pp. 248–253.

[60] L.C. Bezerra, M. López-Ibánez, T. Stützle, Comparing decomposition-based and automatically component-wise designed multi-objective evolutionary algorithms. International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2015, pp. 396–410.

[61] C. Thornton, F. Hutter, H.H. Hoos, K. Leyton-Brown, Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013, pp. 847–855.

[62] M. Feurer, K. Eggensperger, S. Falkner, M. Lindauer, F. Hutter, Auto-sklearn 2.0: Hands-free AutoML via meta-learning, arXiv:2007.04074(2020).

[63] M. Feurer, F. Hutter, Hyperparameter optimization. Automated Machine Learning, Springer, Cham, 2019, pp. 3–33.

[64] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.

[65] I. Das, J.E. Dennis, Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems, SIAM J. Optim. 8 (3) (1998) 631–657.

[66] K. Miettinen, Nonlinear Multiobjective Optimization vol. 12, Springer, 2012.

[67] C.M. Fonseca, L. Paquete, M. López-Ibánez, An improved dimension-sweep algorithm for the hypervolume indicator. IEEE International Conference on Evolutionary Computation, IEEE, 2006, pp. 1157–1163.

[68] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, V.G. Da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Trans. Evol. Comput. 7 (2) (2003) 117–132.

[69] A. Saltelli, Sensitivity analysis for importance assessment, Risk Anal. 22 (3) (2002) 579–590.

[70] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, S. Tarantola, Global Sensitivity Analysis: The Primer, John Wiley & Sons, 2008.

[71] F. Campolongo, A. Saltelli, J. Cariboni, An effective screening design for sensitivity analysis of large models, Environ. Model. Softw. 22 (2007) 1509–1518.

[72] J.J. Liang, S. Baskar, P.N. Suganthan, A.K. Qin, Performance evaluation of multiagent genetic algorithm, Nat. Comput. 5 (1) (2006) 83–96.

[73] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable multi-objective optimization test problems. Proceedings of 2002 IEEE Congress on Evolutionary Computation vol. 1, 2002, pp. 825–830, https://doi.org/10.1109/CEC.2002.1007032.

[74] S. Huband, P. Hingston, L. Barone, L. While, A review of multiobjective test problems and a scalable test problem toolkit, IEEE Trans. Evol. Comput. 10 (5) (2006) 477–506.

[75] F. Pianosi, F. Sarrazin, T. Wagener, An effective screening design for sensitivity analysis of large models, Environ. Model. Softw. 70 (2015) 80–85.

[76] S. Heris, YPEA: Yarpiz evolutionary algorithms, 2019. https://github.com/smkalami/ypea. Accessed on 22 September 2021.

[77] Y. Tian, R. Cheng, X. Zhang, Y. Jin, PlatEMO: a MATLAB platform for evolutionary multi-objective optimization, IEEE Comput. Intell. Mag. 12 (4) (2017) 73–87.

[78] V. Ojha, J. Timmis, G. Nicosia, Sensitivity analysis evolutionary algorithms, 2022. https://github.com/vojha-code/SAofEAs. Accessed on 10 February 2022.

[79] D. Kalpić, N. Hlupić, M. Lovrić, Student's t-Tests, Springer, 2011, pp. 1559–1563.

[80] K. Deb, D. Deb, Analysing mutation schemes for real-parameter genetic algorithms, Int. J. Artif. Intell. Soft Comput. 4 (1) (2014) 1–28.

[81] K. Deb, R.B. Agrawal, et al., Simulated binary crossover for continuous search space, Complex Syst. 9 (2) (1995) 115–148.