Contents lists available at ScienceDirect

Applied Soft Computing

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

On the diversity and robustness of parameterised multi-objective test suites

Estefania Yap [*], Mario Andrés Muñoz, Kate Smith-Miles

School of Mathematics and Statistics, The University of Melbourne, Parkville, VIC 3010, Australia

ARTICLE INFO

Article history:
Received 6 July 2020
Received in revised form 26 May 2021
Accepted 8 June 2021
Available online 21 June 2021

Keywords:
Multi-objective optimisation
Benchmark suites
Experimental evaluation
Exploratory landscape analysis
Algorithm ranking

ABSTRACT

The development of optimisation algorithms heavily relies on comparing performance across benchmark problem suites. In continuous unconstrained multi-objective optimisation, the most popular suites are ZDT, DTLZ and WFG. For each problem in these suites, there are construction parameters that control characteristics such as the degree of multimodality or deceptiveness. Despite encouragement from the suites' authors to do otherwise, experiments are largely performed using only the original values of these parameters. It is important to understand the robustness of these test problems, and their potential to create a diversity of challenging problem landscapes to guide future algorithm testing and development. In this paper we propose a methodology for evaluating robustness of the benchmark test problems by strategically varying construction parameters and exploring how problem difficulty and landscape characteristics are affected. Our methodology adopts both Latin Hyper-cube Sampling and a design and analysis of experiments model to construct more diverse problem instances within the benchmark problem classes. These problem variants are evaluated for eight diverse multi-objective optimisation algorithms to contribute to our understanding of problem robustness. We measure robustness of problems indirectly in terms of impacts on algorithm performance and rankings, and directly in terms of Exploratory landscape Analysis (ELA) metrics that are used to establish problem robustness from a landscape characteristics perspective. Our results show that only eleven of the 21 benchmark problems are robust for algorithms in absolute terms, nine in relative terms, and seven which provide evidence of both types of algorithm robustness. There are also nine problems which satisfy requirements for landscape robustness. Of these, only four of the 21 benchmark problem classes are robust across all measures. These results highlight the importance of diversity in selecting benchmark problems, as the majority of the test suite problems, if only default construction parameters are considered, do not support robust conclusions to be drawn in general about how algorithms perform in the presence of various constructed characteristics intended to challenge algorithms. The existing benchmark test problems are currently insufficient for understanding algorithm performance, certainly with the popularly used default parameters, and more efforts in generating diverse problem instances would serve the research community well.

1. Introduction

Multi-objective optimisation problems (MOOP) are an important class within optimisation due to their applicability to model real world problems. In MOOPs, two or more conflicting goals are simultaneously minimised and/or maximised, with trade-offs between these objectives being observed. A classic example within the economics domain involves the minimisation of risk alongside maximisation of profit.

In practice, to solve a MOOP it is necessary to select and apply a well-performing algorithm. However, it is established by the No Free Lunch Theorems [1,2] that no single algorithm can be expected to be best across the whole set of problems. Instead, an algorithm may only display superior performance over a subset of problems which exhibit certain *characteristics* that the algorithm exploits. This results in the challenge of identifying the relationship between different problem characteristics and algorithm performance. Suites of synthetic benchmark problems exist for the purpose of exploring this relationship and comparing performance. Synthetic problems are able to support the development and testing of algorithms since their characteristics and optimal solutions are conveniently known. A diverse range of difficulties and problem types can be artificially constructed and algorithms systematically tested. In contrast, real-world problems are typically solved by completing experiments. Therefore, the

* Corresponding author.
E-mail address: estefaniay@student.unimelb.edu.au (E. Yap).

https://doi.org/10.1016/j.asoc.2021.107613
1568-4946/© 2021 Elsevier B.V. All rights reserved.

time required, financial cost and constraints on experimental resources often result in the inability to replicate experiments. Consequently, this limits the volume of real-world problems available to study. By using synthetic problems to achieve a better understanding of algorithm behaviours across the diverse set of problems, performance on real-world ones with similar characteristics can be inferred.

As a consequence, benchmarks play an essential role in determining an algorithm's performance against different problem characteristics. The motivation for Deb's initial paper introducing a MOOP generating toolkit was to systematically test algorithms on problems with controlled difficulty [3]. Deb asserted that the true efficiencies of an algorithm are only revealed when evaluated over challenging problems, hence the need for controllable difficulty. As such, its test suite had tuneable parameters that affected the landscape of the resulting MOOP. This approach was followed by the authors guiding the development of the ZDT [3,4] and WFG [5] suites, who encourage the development of different types of problem classes and instances using their provided toolkits. However, over one decade later the original test problems with default parameters are still commonly used for evaluating algorithm performance [6–14] without exploring the impact of construction parameters on the difficulty and conclusions drawn. The few exceptions of variations in parameters are where the number of decision variables and objectives are changed [15,16]. As such, the effect that tuning the construction parameters has on the difficulty of the landscape, and its consequential impact on algorithm performance, is rarely explored. However, it is likely that each one of these test suites is capable of generating a much greater diversity of landscapes than is typically studied when default parameters are assumed. This limits our ability to gain further insights into the true efficiencies of an algorithm. Moreover, benchmark problems are treated as the representative problem of their combination of constructed characteristics, but this may not hold true and should be confirmed with testing. Liao, Molina and Stützle conclude that the ranking of algorithms is dependent on the chosen benchmark set [17], and thus more care should be taken in their selection to ensure efficient information gain for each algorithm.

Therefore, one aim should be to understand the potential of each test suite problem to generate a *diversity* of landscapes, and to explore their *robustness* in terms of landscape difficulty and impact on algorithm performance. This is relevant because an algorithm may perform differently on problems with different levels of specification of a characteristic (e.g. from a few to hundreds of local optima). This information would be useful for the construction of new benchmarks, and guiding algorithm development and tuning by understanding their strengths and weaknesses. The concept of robustness is commonly associated with algorithms, rather than problems, where an algorithm is considered robust if similar performance is obtained despite algorithm parameter tuning [18]. Alternatively, a robust problem is one whose landscape is resistant to change despite parameter variations, such that the performance of a group of algorithms in both relative and absolute terms, does not change. In other words, the characteristics present using default parameters will persist despite tuning. Similarly, non-robust problems produce a greater variety of landscape characteristics after tuning, leading to changes on performance and a better understanding of the strengths and weaknesses of an algorithm. Understanding which test problems are robust, and which have potential to generate a greater diversity of landscapes, is important to ensure that the available test suites are fully understood for their potential. The learned information can provide guidance to the next generation of test suite constructions.

Given the similarity of concepts between robustness of algorithms and problems in the face of parameter tuning, in this paper we propose an extension of the framework posed by Eiben and Smit [18] for analysing parameter tuning robustness of algorithms. In contrast to their definition of algorithm robustness, we evaluate problem robustness considering the landscape difficulty using two approaches: directly by calculating landscape metrics and exploring the change in such metrics over the parameter space; and indirectly, using algorithm performance as a proxy measure of difficulty, considering no changes in absolute performance metric[1] for each algorithm as an indication of robustness of difficulty, as well as relative robustness using algorithm ranking statistics. We assert and will later demonstrate that tuning of construction parameters results in controllable difficulty not only between different problem classes, but also within different instances of a problem class, enabling a much greater diversity of useful problems to be studied from the test suites than those typically reported on using only the default parameters. As such, our work complements ongoing research into the construction of benchmarks that will challenge MOOP algorithms [5,13,14, 20–25,25–28], as well as the abundant research into the development and refinement of algorithms [21,29–31] that perform well for certain landscape difficulties, by providing information on whether algorithm performance can be generalised [32] to problems that may differ from those studied.

With this framework in mind, we investigate the effects of strategic variations of the construction parameters for continuous, unconstrained multi-objective benchmark test problems. A portfolio of eight state-of-the-art Evolutionary Algorithms (EA) is initially evaluated on three popular benchmark suites, i.e., ZDT, DLTZ and WFG, which have as an advantage their clear construction method. Then, variations in construction parameters are introduced in two phases, and the same algorithms are evaluated again on these new instances of each problem. In the first phase the variation is introduced by Latin Hyper-cube Sampling (LHS) of the parameter space to generate diverse samples of the construction parameters using a space filling design. In the next phase, a Design and Analysis of Experiments (DACE) model is used to strategically explore the relationship between construction parameters and problem difficulty for each algorithm across the DTLZ and WFG suites. We then utilise the model to search for construction parameter combinations that generate harder problems for each algorithm. These variants of problem instances – attempting to push the diversity and difficulty of the test suite problems beyond their default parameter settings – form the basis for our analysis of robustness. It is imperative to note that the contribution of this paper is not to explore new techniques to evolve harder problems as has previously been done [33,34]. However, we do generate new difficult test instances using a simple DACE model in order to demonstrate the importance of diverse problem instances for exploring the full potential of the test suites, and challenging conclusions about algorithms on test problems that may only be valid for particular parameter settings.

This paper is organised as follows. We describe the test suites and problems used in Section 2. In Section 3 we present our method of generating problem variants in a suite by strategically adapting construction parameters using a DACE model. Furthermore, we define the concept of problem robustness, and how it is measured. Empirical results in Section 4 demonstrate that there is potential for most test problems to be made harder, and that algorithm performance changes across instances within the parameter domain. We further extend our analysis to include two suites that are more recent. Section 5. Section 6 concludes the paper with a discussion of the implications of this study and the future direction of this research.

[1] We use a modified version of the Inverted Generational Distance (IGD^+) [19] metric named IGD^+_M as our performance metric, but this choice could be replaced with any other common measure capable of comparing performance across different problems.

2. Test suites

Test problems are constructed to contain characteristics that can challenge optimisation algorithms. A collection of these problems forms a test suite. Deb [3] first introduced a method by which benchmarks for multi-objective problems could be constructed in order to address the lack of such problems in MOOPs. Deb suggested constructing bi-objective test problems using three functions, with f_1 controlling the difficulty of algorithms along the Pareto-optimal front (PF), g controlling problem characteristics, and h defining the shape of the PF. This toolkit was utilised by Zitzler et al. [4] to develop the ZDT test suite. The ZDT test suite has received criticism [5,35,36] due to the lack of difficulty, and although it is no longer popularly used, we study it due to its recognition within the community. Deb et al. created the DTLZ problems [35] to address this, using a different method of construction for scalable test problems that were no longer limited to bi-objective problems and had desirable properties such as known PFs. However, despite the flexibility in the number of objectives, the DTLZ suite lacked the flexibility of Deb's initial toolkit in the construction and inclusion of problem characteristics.

Upon review of the existing multi-objective test suites, Huband et al. introduced the WFG toolkit [37] that could control desired characteristics by using a series of transformations that determined their presence. The toolkit resulted in the WFG test suite which included problems with characteristics not included in prior suites.

Test suites are designed to allow for conclusions to be drawn by testing algorithms across a variety of characteristics. However, it is impractical for test suites to contain all possible combinations of characteristics. Moreover, it is too computationally expensive to run each algorithm on every combination. Instead, they are constructed alongside guidelines that suggest the types of problems to be included [5]. Problems that are chosen to represent a characteristic or combination of characteristics within a test suite have fixed construction parameters which are selected by their designers.

The choice of test problems in a suite is crucial since the conclusion of a superior algorithm is supported by its stronger performance over the test suite. If the benchmarks with fixed construction parameters are not stable for algorithm performance when construction parameters are varied, the conclusions drawn may be accidentally misleading. We investigate these concepts of problem robustness on the popular ZDT, DTLZ and WFG test suites after briefly reviewing their construction.

2.1. ZDT test suite

The ZDT test suite was built using Deb's toolkit which used three functions to construct problems:

$$\begin{aligned} &\text{Minimise } \mathcal{F}(\mathbf{x}) = (f_1(x_1), f_2(\mathbf{x})) \\ &\quad \text{s.t. } f_2(\mathbf{x}) = g(x_2, \dots, x_m) h(f_1(x_1), g(x_2, \dots, x_m), \alpha, q) \end{aligned} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^m$, and m is the number of decision variables. The function f_1 is only a function of the first decision variable x_1, g is a function of the remaining $m-1$ variables and h is a function of both f_1 and g. As a result, f_2 is a function of all m decision variables, with $g(x_2, \dots, x_m)$ defining the problem characteristic. The construction parameters α and q for each problem (where applicable) are defined within h. The shape of the PF is convex when $\alpha < 1$, concave when $\alpha > 1$ and linear when $\alpha = 1$, and q controls the number of discontinuous PFs.

Although one of the most popular multi-objective benchmark test suites, the ZDT test suite [4] has been criticised for being simplistic and restricted to bi-objective problems [5,38], and thus not difficult for algorithms. There are only five unconstrained, continuous test problems (ZDT1-ZDT4, ZDT6) within this test suite as specified, which are chosen to represent a variety of characteristics. Due to the problem construction, all PFs are formed with $g(\mathbf{x}) = 1$ and are used to measure algorithm performance.

2.2. DTLZ test suite

The seven unconstrained DTLZ test suite problems (DTLZ1-DTLZ7) are developed using a bottom-up approach where the PF is defined first, and subsequently the search space is constructed by adding surfaces parallel to the PF. For M objective functions, the decision variable vector can be partitioned into M non-overlapping groups

$$x \equiv (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{M-1}, \mathbf{x}_M) \tag{2}$$

and the problem can be defined as:

$$\begin{aligned} &\text{Minimise } f_1(\mathbf{x}_1), \\ &\text{Minimise } f_2(\mathbf{x}_2), \\ &\qquad\quad \vdots \\ &\text{Minimise } f_{M-1}(\mathbf{x}), \\ &\text{Minimise } f_M(\mathbf{x}) = g(\mathbf{x}_M) h\big(f_1(\mathbf{x}_1), f_2(\mathbf{x}_2), \dots, \\ &\qquad\qquad f_{M-1}(\mathbf{x}_{M-1}), g(\mathbf{x}_M)\big), \\ &\qquad \text{s.t. } \mathbf{x}_i \in \mathbb{R}^{|\mathbf{x}_i|}, \text{ for } i = 1, 2, \dots, M \end{aligned} \tag{3}$$

Similarly to the ZDT problems, the Mth objective is formed by two functions, g and h. The PF is described by solutions which are the global minimum of $g(\mathbf{x}_M)$ (at g^*):

$$f_M = g^* h(f_1(\mathbf{x}_1), f_2(\mathbf{x}_2), \dots, f_{M-1}(\mathbf{x}_{M-1}), g^*) \tag{4}$$

where g controls the problem characteristics and h controls the shape of the PF. Although the DTLZ problems are scalable to any number of objectives and decision variables, their method of construction results in many restrictions for the characteristics which can be introduced [5]. Therefore, this suite lacks variety when testing algorithms. The standard problems are tri-objective.

2.3. WFG test suite

The construction of the WFG toolkit uses a different approach compared to ZDT and DTLZ. Starting with an initial vector of parameters[2] $\mathbf{x}$ the presence of each problem characteristic is introduced sequentially by applying a series of transition vectors, $\mathbf{t}^i$ where $i = 1, \dots, p$ represents the ith characteristic. The inclusion of each additional transition vector provides added complexity until finally a vector $\mathbf{w}$ is derived, which is associated with the objective functions. EAs are only able to indirectly manipulate $\mathbf{w}$ through the manipulation of $\mathbf{x}$. The format of the WFG toolkit is defined as:

$$\begin{aligned} \text{Given } \mathbf{x} &= \{x_1, \dots, x_n\} \\ \text{Minimise } f_{m=1:M}(\mathbf{w}) &= Dw_M + S_m h_m(w_1, \dots, w_{M-1}) \\ \text{where } \mathbf{w} &= \{w_1, \dots, w_M\} \\ &= \{max(t_M^p, A_1)(t_1^p - 0.5) + 0.5, \dots, \\ &\quad max(t_M^p, A_{M-1})(t_{M-1}^p - 0.5) + 0.5, t_M^p\} \\ \mathbf{t}^p &= \{t_1^p, \dots, t_M^p\} \longleftarrow t^{p-1} \longleftarrow \dots \\ &\quad \longleftarrow t^1 \longleftarrow \mathbf{x}_{[0,1]} \\ \mathbf{x}_{[0,1]} &= \{x_{1,[0,1]}, \dots, x_{n,[0,1]}\} \\ &= \{x_1/x_{1,max}, \dots, x_n/x_{n,max}\} \end{aligned} \tag{5}$$

[2] In the original paper of Huband et al. [5] the vector of parameters is referred to as $\mathbf{z}$, not $\mathbf{x}$. We use $\mathbf{x}$ for decision parameters to keep the nomenclature consistent in this paper.

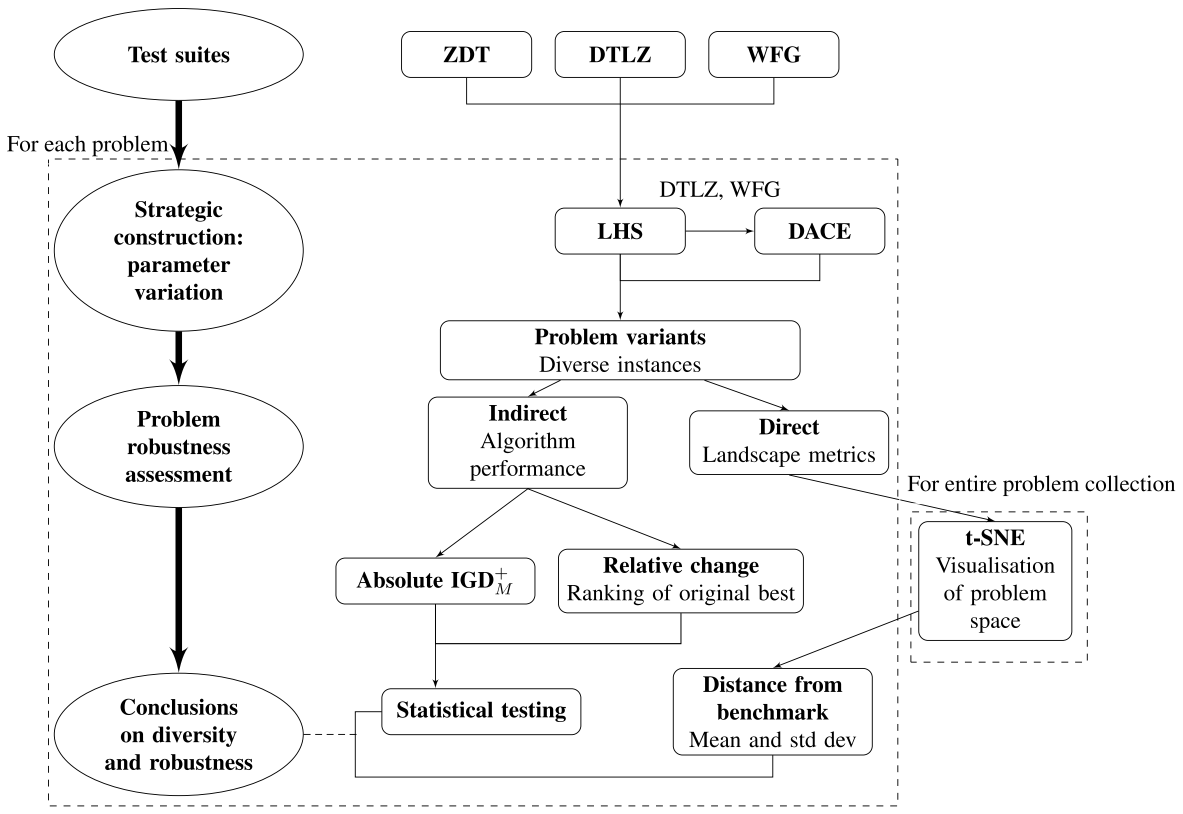


Fig. 1. Flowchart of relationships and process of assessing problem robustness.

where M is the number of objectives, $\mathbf{w}$ is a vector of the M underlying parameters, $\mathbf{x}$ the set of decision variables , $D > 0$ is the distance scaling constant, $A_{1:M-1} \in \{0, 1\}$ are the degeneracy constants where each $A_i = 0$ reduces the dimensionality of the PF by one, $h_{1:m}$ are shape functions determining the shape of the PF, $S_{1:M} > 0$ are scaling constants, $\mathbf{t}^{1:p}$ are transition vectors with "$\longleftarrow$" indicating each sequential transition.

For further information regarding the composition of the WFG toolkit and problems, the reader is referred to the original paper [5].

3. Methodology

Fig. 1 illustrates our methodology to evaluate problem robustness. Starting with a problem suite, such as ZDT, DTLZ or WFG, a set of algorithms and a performance metric to evaluate them, we generate an initial set of problem instances using Latin Hyper-cube Sampling (LHS) of the parameter space. After collecting performance data, a Design and Analysis of Computer Experiments (DACE) model is used to select parameter combinations that create more difficult test problems. Once enough new and diverse test instances have been constructed, we evaluate problem robustness indirectly through observing changes in algorithm performance as the parameters of a test problem are varied, as well as directly where the landscape changes. Impact on algorithm performance across the parameter space is measured in both absolute terms of the performance measure, and in relative terms by measuring whether the best algorithm and ranking statistics have changed from the original problem ranking of algorithms. As a more direct measure of whether parameter variation has effectively changed the nature of the intended problem characteristics established by the default parameter setting, we use exploratory landscape analysis (ELA) metrics to summarise test problem characteristics. These ELA metrics are used as feature vectors on the entire problem collective to generate a t-distributed stochastic neighbour embedding (t-SNE) visualisation of the problem space containing all variants. The mean and standard deviation of the distances between the original problem and the problem variants is then calculated to measure the magnitude of any landscape change. A conclusion of problem robustness is supported when statistical testing and exploration provides both indirect and direct evidence that problem characteristics persist despite variation in the default parameter settings. A lack of robustness of a test problem means that the intended characteristics may not persist if the non-default parameter setting is used, but a greater landscape diversity can be explored if beneficial to gain further insights into algorithm strengths and weaknesses. Further details within each step are now presented in the remainder of this section.

3.1. Algorithms and performance metric

Experiments were run on the existing benchmark test problems and all generated instances in the bi-objective ZDT, and tri-objective DTLZ and WFG test suites using notable algorithms. The focus of this paper is not on the strengths and weaknesses of particular *algorithm instances* obtained through tuning a generic algorithm. Instead, it is about the insights into *problem instance* robustness and diversity evident from considering a portfolio of diverse algorithms reflecting a wide distribution of performances. To ensure this, state-of-the-art algorithms were chosen to cover a diverse range of elitism and niching strategies [39,40]. Of course, other algorithms could be chosen, and the methodology is repeatable. The chosen algorithms fall into four different classes [41]:

- Pareto dominance: NSGA-II [42], NSGA-III [15] and SPEA2 [43]

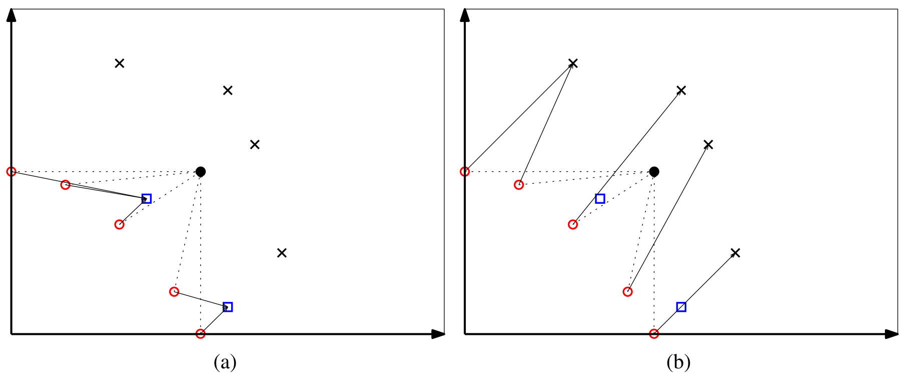


Fig. 2. The difference in the rescaled IGD$^+$ calculation for approximate fronts below and above the nadir point (filled circle). The total distance of the solid lines are scaled by the total distance of the dotted lines. (a) The case where the approximate front (square) is below the nadir point, based on the Pareto Front (open circle). Calculating the IGD^+_N will result in a value bounded between [0, 1]. (b) When the approximate front (cross) is above the nadir point, IGD^+_N metric will be above 1.

Table 1
Problem variation design for the ZDT and DTLZ test suites. The original parameter values used for each benchmark problem are included.

Problem	Parameter Design	Original
ZDT1	$\alpha \in \{1, 2, \dots, 30\}$	$\alpha = 0.5$
ZDT2	as ZDT1	$\alpha = 2$
ZDT3	$\{\alpha, q\} \in [0, 30]^2$	$\alpha = 0.5$
		$q = 5$
ZDT4	$\alpha \in \{1, 2, \dots, 30\}$	$\alpha = 0.5$
ZDT6	$\alpha \in \{0.5, 1, 2, \dots, 30\}$	$\alpha = 2$
	$p_1 \in \{1, 2, \dots, 100\}$	$p_1 = 10$
DTLZ1	$p_2 \in [0, 1]$	$p_2 = 0.5$
	$p_3 \in [0, 100]$	$p_3 = 100$
DTLZ2	$p_1 \in [0, 1]$	$p_1 = 0.5$
	$p_1 \in \{1, 2, \dots, 100\}$	$p_1 = 10$
DTLZ3	$p_2 \in [0, 1]$	$p_2 = 0.5$
	$p_3 \in [0, 100]$	$p_3 = 100$
DTLZ4	$p_1 \in [0, 100]$	$p_1 = 10$
	$p_2 \in [0, 1]$	$p_2 = 0.5$
DTLZ5	$p_1 \in [0, 1]$	$p_1 = 0.5$
DTLZ6	$p_1 \in [0, 1]$	$p_1 = 0.1$
DTLZ7	$p_1 \in [0, 100]$	$p_1 = 3$
	$p_2 \in [0, 10]$	$p_2 = 1$

- Decomposition: MOEA/D (Tchebycheff decomposition) [44] and RVEA [36]
- Grid-based: GrEA [45]
- Indicator-based: HypE [46] and IBEA [47].

We use the PlatEMO toolbox [21] for algorithms and performance indicators, run with a budget of 10^4 evaluations using default algorithm settings in PlatEMO. Our reasoning for using only default parameters for the algorithm implementations is three-fold: firstly, algorithm parameter tuning requires function evaluations and we do not have sufficient budget (10^4 evaluations) to perform both problem solving and algorithm parameter tuning (to optimality). Secondly, we consider the default parameters to be representative of the most likely performance of the algorithm in practice, since users are often content to use default parameter settings when trialling algorithms. Thirdly, the purpose of our study is to establish the robustness of problem instances, and the performance metrics of an arbitrary set of algorithms and their implementation is used as a proxy measure for robustness to ascertain the stability of the performance of the algorithms amidst problem construction parameter variation. The choice of the algorithms and their parameters settings is therefore less critical as it is only a means to observe stability and robustness. If different algorithm parameter setting were chosen for an algorithm implementation, we would consider that to be a different algorithm for the purposes of this study.

Our next step is to choose a performance metric. There is no single metric for MOOPs that can identify the best solutions for every aspect of convergence, diversity and spread, and number of solutions [48]. This is due to the existence of the PF and the multiple solutions required to be generated by an EA. Therefore, we consider only metrics which take into account convergence and diversity. Both IGD$^+$ and Hypervolume (HV) satisfy this requirement. The relationship between the two metrics is consistent on convex PFs and can therefore be jointly used to assess solution set optimality [49]. However, on concave PFs, they are not consistent [48] and HV may be a misleading measure of performance [50]. Additionally, HV generally prefers knee and boundary points rather than well distributed ones [51]. It is for these reasons that we choose IGD$^+$ as our comparative metric. The original IGD metric is calculated as:

$$IGD(Z, A) = \frac{\left(\sum_{i=1}^{|Z|} d_i^2\right)^{1/2}}{|Z|} \tag{6}$$

where $d_i = \min_{\vec{s} \in A} |F(\vec{z}_i) - F(\vec{s})|, \vec{z}_i \in Z$, and d_i is the smallest distance from each solution in the PF $\vec{z}_i \in Z$ to the closest solutions in the approximated front $\vec{s} \in A$. The distance from each $\vec{z}_i$ to the solution $\vec{s}$ is calculated as $\sqrt{\sum_{i=1}^{|Z|} (\vec{s}_i - \vec{z}_i)^2}$. While IGD does not factor in the dominance relation [19], IGD$^+$ does so by modifying the distance calculation to $\sqrt{\sum_{i=1}^{|Z|} max\,(\vec{z}_i - \vec{s}_i, 0)^2}$.

However, since we are making comparisons across problems and potentially different PFs, it is important that we select a measure that allows for such comparison. We propose a modified version of IGD$^+$ to allow this. Firstly, we normalise IGD$^+$ by dividing by the mean distance between nadir point r, and the PF as our reference set, resulting in:

$$IGD^+_N(Z, A) = \frac{(\sum_{i=1}^{|Z|} d_i^2)^{1/2}}{(\sum_{i=1}^{|Z|} n_i^2)^{1/2}} \tag{7}$$

where n_i is the smallest distance from every $\vec{z}_i \in Z$ to r. The solutions between or equivalent to the true PF and r will result in IGD$^+_N \in [0, 1]$. However, since obtained solutions may be worse than the r as in Fig. 2, it is possible for IGD$^+_N \to \infty$ as performance worsens. Since the extent of poor performance beyond r is not important, we propose another modification to truncate these

Table 2
Problem variation design for the WFG test suites. The original parameter values used for each benchmark problem are included.

Problem	Parameter Design	Original
WFG1	$p_1 \in (0, 1)$	$p_1 = 0.35$
	$p_2 \in (0, 1)$	$p_2 = 0.8$
	$p_3 \in (0, 1)$	$p_3 = 0.75$
	$p_4 \in (0, 1)$	$p_4 = 0.85$
	$p_5 \in (0, 10]$	$p_5 = 0.02$
	$p_6 \in (0, 10]$	$p_6 = 1$
	$p_7 \in \{1, 2, \dots, 10\}$	$p_7 = 5$
WFG2	$p_1 \in (0, 1)$	$p_1 = 0.35$
	$p_2 \in (0, 10]$	$p_2 = 1$
	$p_3 \in (0, 10]$	$p_3 = 1$
	$p_4 \in \{1, 2, \dots, 10\}$	$p_4 = 5$
WFG3	$p_1 \in (0, 1)$	$p_1 = 0.35$
WFG4	$p_1 \in (0, 1)$	$p_1 = 0.35$
	$p_2 \in \{1, 2, \dots, 100\}$	$p_2 = 30$
	$p_3 \in [0, 100]$	$p_3 = 10$
WFG5	$p_1 \in (0, 1)$	$p_1 = 0.35$
	$p_2 \in (0, 0.1]$	$p_2 = 0.001$
	$p_3 \in (0, 0.1]$	$p_3 = 0.05$
WFG6	$p_1 \in (0, 1)$	$p_1 = 0.35$
WFG7	$p_1 \in (0, 1)$	$p_1 = 0.35$
	$p_2 \in (0, 1)$	$p_2 = 0.98/49.98$
	$p_3 \in (0, 1]$	$p_3 = 0.02$
	$p_4 \in (0, 100]$	$p_4 = 50$
WFG8	$p_1 \in (0, 1)$	$p_1 = 0.35$
	$p_2 \in (0, 1)$	$p_2 = 0.98/49.98$
	$p_3 \in (0, 1]$	$p_3 = 0.02$
	$p_4 \in (0, 100]$	$p_4 = 50$
WFG9	$p_1 \in (0, 1)$	$p_1 = 0.35$
	$p_2 \in (0, 1)$	$p_2 = 0.98/49.98$
	$p_3 \in (0, 100]$	$p_3 = 0.02$
	$p_4 \in (0, 100]$	$p_4 = 50$
	$p_5 \in (0, 0.1]$	$p_5 = 0.001$
	$p_6 \in (0, 0.1]$	$p_6 = 0.05$
	$p_7 \in \{1, 2, \dots, 100\}$	$p_7 = 30$
	$p_8 \in [0, 100]$	$p_8 = 95$

results that show poor performance:

$$IGD^+_M = \begin{cases} 1 - IGD^+_N & 0 \le IGD^+_N \le 1 \\ \frac{1}{IGD^+_N} - 1 & 1 < IGD^+_N \end{cases} \quad (8)$$

This results in the convenient format where $\text{IGD}^+_M = 1$ occurs when the obtained solutions are equivalent to the PF, and problems become harder (as indicated by worse performance) as $\text{IGD}^+_M \to -1$. Consequently, we can consider IGD^+_M as a goodness metric.

3.2. Strategic construction of problem variants

Instance generation is used to test the robustness of algorithm performance by perturbing construction parameters. We use the MATSuMoTo surrogate model toolbox [52] for problem instance generation. Problem dimensions are not altered and are instead held constant to best demonstrate how construction parameters affect algorithm performance. With the exception of ZDT3, all other ZDT problems have only one parameter that is varied on evenly-spaced increments. The remainder problems are varied in three stages: (a) generating an initial set of 30 diverse problem instances; (b) fitting a model to predict the expected difficulty of the combination of different construction parameters for each algorithm; and (c) iteratively generating up to 5 new instances for each algorithm by maximising the expected improvement (EI) [53] of the fitted model. Tables 1 and 2 show the construction parameter domains we explore for the ZDT, DTLZ and WFG[3] test suites. Since the α parameter in the ZDT suite is used to control the extent of the convexity/concavity of the Pareto front, for problems with only this single parameter (all but ZDT3), we explore only integer values because the shape will be convex when $\alpha < 0$, linear when $\alpha = 0$ and concave when $\alpha = 1$. Here we are less concerned with minor degrees of convexity and as such, the original benchmark of ZDT1 and ZDT4 cover the single convex versions among all the instances. We also do not look at problem robustness of growing dimensionality for problems, and instead hold the number of decision and objective variables constant, while only altering construction parameters.

[3] Parameters for the WFG problems are subject to constraints. The reader is directed to the paper by Huband et al. [5] for further information.

3.2.1. Initial problem generation via latin hyper-cube sampling

Our design of experiments uses LHS design to generate diverse samples from multi-dimensional distributions, consisting of 30 instances within each problem class. These are also later referred to as "LHS perturbations" in this paper. Within the WFG test suite, there are several restrictions on the construction parameters for the shape and transformation functions. LHS is used for each separate characteristic that is then compiled together for each problem. For the transformations with constraints on relationships such as the multi-modal shift requiring $(4p_2 + 2)\pi \geq 4p_3$, we generated 100 LHS combinations and retain only the first 30 that satisfy the constraints.

3.2.2. Design and analysis of computer experiments (DACE) model

New problem instances are generated by fitting a model to the initial sample of the previously described 30 LHS instances, and iteratively searching for potentially more difficult variants. The DACE model [54] is a surrogate modelling technique that identifies the approximate relationship between inputs and outputs. An advantage of a DACE model is that it focuses on maximising difficulty (smaller IGD_M^+) while minimising uncertainty. As such, while difficulty is used as the output for the DACE search, its purpose is not only to generate harder problems, but also balancing this with finding inputs which are expected to iteratively increase model accuracy. As such, we obtain problems which are not only harder, but also ones which are potentially more diverse. Assume we have a model of a stochastic process in the form:

$$y(\mathbf{c}^{(i)}) = \mu + \epsilon(\mathbf{c}^{(i)}) \tag{9}$$

where μ is the mean response, $\epsilon(\mathbf{c}^{(i)}) \sim N(0, \sigma^2)$ is the error, $\mathbf{c}^{(i)}$ is the sampled input and $y(\mathbf{c}^{(i)})$ the corresponding estimated output.

Unlike in linear regression where the error terms are independent, the DACE approach assumes that if $\mathbf{c}_i$ and $\mathbf{c}_j$ are close, their errors $\epsilon(\mathbf{c}_i)$ and $\epsilon(\mathbf{c}_j)$ must also be close. To account for correlations differing across distances, a special weighted distance function is used:

$$d(\mathbf{c}^{(i)}, \mathbf{c}^{(j)}) = \sum_{h=i}^{k} \theta_h |c_h^{(i)} - c_h^{(j)}|^{Ph}, (\theta_h \geq 0, p_h \in [1, 2]) \tag{10}$$

With this function, the correlation between $\mathbf{c}^{(i)}$ and $\mathbf{c}^{(j)}$ is:

$$\text{Corr}\left[\epsilon(\mathbf{c}^{(i)}), \epsilon(\mathbf{c}^{(j)})\right] = \exp\left[-d(\mathbf{c}^{(i)}, \mathbf{c}^{(i)})\right] \tag{11}$$

The DACE model has a total of $2k + 2$ parameters: $\left\{\mu, \sigma^2, \theta_1, \dots, \theta_k, p_1, \dots, p_k\right\}$ which are estimated by maximising the likelihood of the sample input–output observations given the model.

Let $\mathbf{y} = \left(y^{(1)}, \dots, y^{(n)}\right)'$ denote the n-vector of observed function values, $\mathbf{R}$ denote the $n \times n$ matrix whose (i,j)'s are $\text{Corr}\left[\epsilon(\mathbf{c}^{(i)}), \epsilon(\mathbf{c}^{(j)})\right]$, and $\mathbf{1}$ denote the n-vector of ones, which multiplies the scalar μ. We initialise the model with $\theta_1, \dots, \theta_k$ with the correlation function parameters, and $p_1, \dots, p_k$ with values of 2. The likelihood function is then written as:

$$\mathcal{L} = \frac{1}{(2\pi)^{n/2}(\sigma^2)^{n/2}|\mathbf{R}|^{\frac{1}{2}}} \exp\left[-\frac{(\mathbf{y} - \mathbf{1}\mu)'\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\mu)}{2\sigma^2}\right] \tag{12}$$

and maximised using calculus to estimate optimal μ and σ^2 as:

$$\hat{\mu} = \frac{\mathbf{1}'\mathbf{R}^{-1}\mathbf{y}}{\mathbf{1}'\mathbf{R}^{-1}\mathbf{1}}, \tag{13}$$

$$\hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{1}\hat{\mu})'\mathbf{R}^{-1}\mathbf{y} - \mathbf{1}\hat{\mu}}{n} \tag{14}$$

The values of $\hat{\mu}$ and $\hat{\sigma}^2$ are then substituted into Eq. (12) and this function is again maximised to obtain estimates for $\hat{\theta}_h$ and $\hat{p}_h$.

For the purpose of this study, we generate a surrogate model for each algorithm on each problem, with construction parameters as inputs and the respective IGD_M^+ as the outputs.

Since problems become easier as $\text{IGD}_M^+ \to 1$, The DACE model is then used to search for optimal construction parameter vectors $\mathbf{c}$ that generate new instances that minimise IGD_M^+ for each algorithm. The model is used to iteratively identify the next instance $\mathbf{c}$ whose improvement $I(\mathbf{c})$ in reducing IGD_M^+ as a function of parameters, is maximised. This expected improvement (EI) is defined as:

$$\text{E}\left[I(\mathbf{c})\right] = \text{E}\left[\max(f_{\min} - \mathbf{Y}, 0)\right] \tag{15}$$

Here f_{min} denotes the model's best obtained value of IGD_M^+ so far, and Y the random variable being modelled by $y(\mathbf{c})$. EI is large where the uncertainty is large or when there is a high likelihood that $y(\mathbf{c})$ will be smaller than f_{min}. As a result, EI is able to balance local and global search by modelling the uncertainty at $\mathbf{c}$ and identifies the next instance that it believes can lead to a more difficult problem.

Due to the stochastic nature of the algorithms' search mechanisms and dependence on the initial population, we evaluate each algorithm 30 times and obtain the mean IGD_M^+ as the target model output.

3.3. Evaluating problem robustness

In the existing literature, algorithm robustness in the context of parameter tuning is defined in relation to the variance of algorithm performance across three dimensions: the problem instances, tuneable algorithm parameter vectors, and random seeds [18]. Our study is related to understanding the impact of problem instance variation through the tuning of the problem construction parameters. Therefore, we define problem robustness in two ways: through directly observing the change in landscape, as well as indirectly by using algorithm performance as a proxy measure. While changes in landscape are considered a direct measure of diversity, in isolation this information is insufficient. If algorithms perform the same across diverse landscapes, we do not gain much new information about algorithms on these updated characteristics (although we could conclude the algorithm performs similarly on all instances of that problem class). As such, understanding how algorithms respond to these changes is important too. Conversely, with only indirect measures, information is lost on what changes are occurring in the landscape that are causing the algorithm performance variation. For this reason, we need to consider both types of measures, direct and indirect, in evaluating problem robustness.

3.3.1. Indirect measures of robustness

Indirectly, problem robustness is measured by the variation of algorithm performance on the problem instances generated when construction parameters are varied. Algorithm performance comparisons can be made using:

1. absolute performance as the difference in performance measure (IGD_M^+) values (irrespective of the portfolio), and
2. relative performance in terms of the final ranking of each algorithm (within a portfolio) based on a performance metric.

The importance of the two different measures depends on whether the motivation of a study is to understand the algorithm's performance in absolute or relative terms. As such, we treat these as separate measures of robustness to changes in problem specification and construction parameters. It is important to note that for indirect measures which use algorithms as

Table 3
Features calculated to characterise multi-objective landscapes, captured via a random walk.

Description	First autocorrelation	Average
Proportion of dominated solutions	#inf_r1_rws	#inf_avg_rws
Proportion of dominating solutions	#sup_r1_rws	#sup_avg_rws
Proportion of incomparable solutions	#inc_r1_rws	#inc_avg_rws
Proportion of non-dominated solutions	#lnd_r1_rws	#lnd_avg_rws

a proxy for evaluating problem difficulty, the algorithm's robustness, for a fixed setting of algorithm parameters, is evaluated over construction parameter variation in order to evaluate problem robustness.

For consistency, we adapt the definitions of algorithm robustness from Eiben and Smit [18] when indirectly measuring problem robustness through absolute algorithm performance. In the same way that a problem instance represents a given parameter setting for a problem, an EA instance represents an algorithm with a given parameter setting. We follow Eiben and Smit's definitions of robustness being applicable to individual EA instances and not all versions of the EA in general. In Eiben and Smit's framework, robustness is evaluated on the collection of benchmark problems $F_i \in \mathcal{F}$, and conclusions about algorithm robustness are drawn based on the overall performance on the entire collection, $\mathcal{F}$. In our case, we are interested in the robustness of each individual F_i, by considering a set of parameter variations $\bar{p}_k$. As such, we draw algorithm conclusions for each F_i based on the performance on the collection of all k instances. Eiben and Smit define six types of algorithm robustness, of which we will adopt the four of highest relevance: widely applicable, fallible, tuneable, and stable. Originally, an algorithm A, tuned with parameters $\bar{p}$ is considered robust if it demonstrates consistently strong absolute performance (above a threshold) across a large number of problems. It is the tuned algorithm $A(\bar{p})$ which is robust, not the problems $F_i \in \mathcal{F}$. Eiben and Smit define this robustness as *widely applicable*. Furthermore, if $A(\bar{p})$ shows a large range of absolute performance when evaluating across $\mathcal{F}$, $A(\bar{p})$ is deemed *fallible*. It is important to note that wide applicability and fallibility can co-exist, as they measure two different phenomenon. However, wide applicability implies that fallibility is limited to a small number of F_i. We adapt both terms for absolute performance by substituting F_i with $F_i(\bar{p})$ and $\mathcal{F}$ for F_i. In the case of relative performance, we adapt the term *stable* when the relative performance for the best algorithm (on the original benchmark) stays consistently best over new instances. In Eiben and Smit's framework, stability is defined as when the difference between the best and worst runs of $A(\bar{p})$ is small, when initialising on different random seeds. We consider these as similar measures, since they account for the change in/from best. Unlike for absolute performance, we do not measure the range of deviation from best. Lastly, we adapt the term *tuneable* to define problems where construction parameter variation leads to significant differences in landscape characteristics, rather than describing algorithms with a large performance variation induced by changes in algorithm parameter values as per the original definition [18].

We summarise our adaptation of the Eiben and Smit's framework of algorithm robustness [18] with the following definitions:

Definition 1. An algorithm is robust - in indirect absolute terms - if it is widely applicable and/or non-fallible across all construction parameter variations

Definition 2. An algorithm is robust - in indirect relative terms - if its ranking within a portfolio of algorithms is stable across all construction parameter variations

Definition 3. The problem landscape is robust if it is not tuneable, such that construction parameter variations do not significantly change the landscape characteristics

Definition 4. A problem F_i is robust if both indirect and direct measures demonstrate robustness, i.e. algorithm robustness and problem landscape robustness must both be satisfied

To confirm the validity of our results, Mann–Whitney U tests were conducted for both wide applicability and stability. In the former, we compared the individual runs for each algorithm on instances created by LHS perturbation (problems generated using LHS) and DACE, against runs on the original benchmark. An α level of 0.05 is used to reject the null hypothesis that there is no change in difficulty, hence a small p-value supports a claim of difficulty varying and simultaneously, wide applicability and fallibility. In the case of stability, a Mann–Whitney U test was used with the individual runs and their ranks are compared. An α level of 0.05 is used to reject the null hypothesis that there is no change in ranking of the original best algorithm, hence a small p-value supports a claim of algorithm instability.

3.3.2. Direct measures of robustness

In measuring robustness directly through the landscape, we consider the problem itself as tuneable to the construction parameters if there is variation in the landscape. Problem landscapes can be measured by extracting features which characterise their topology. These features can be used to quantify challenges that algorithms may encounter while solving the problem, but the impact of different landscapes on algorithms may differ. As such, we consider the absence of significant changes in the landscape in our definition of problem robustness. We define the landscape by measuring existing multi-objective features via random walks that are shown to correlate with characteristics of benchmarks [55]. While the selected features will impact the variation present between landscapes, we argue that the presence of any observed change in landscape, regardless of chosen features, demonstrates a lack of robustness.

Features were calculated during a random walk of length 10^3, with 10 neighbours selected for each sample. Thereafter, the proportion of dominated, dominating, incomparable and non-dominated in the neighbourhood were obtained. Measures were taken for the first autocorrelation across the 10^3 samples, as well as the average. Features used are summarised in Table 3. T-distributed stochastic neighbour embedding (t-SNE) separated by Euclidean distance is used for dimension reduction to measure the average deviance between instances, as well as to provide visualisations in 2d. Thereafter, the means and standard deviation between and original benchmark problem and the problem variants are calculated using their co-ordinates in the t-SNE problem space. Problems which are tuneable will demonstrate large mean and standard deviations between distances.

Understanding the robustness of suites can provide information to guide algorithm development. The concepts of wide applicability, non-fallibility, stability and tuneability establish the formal definition of problem robustness we use in this paper. The combination of the first three signify that conclusions drawn by algorithms on a single instance are likely to hold across the entire problem class, while problems that lack robustness and are also tuneable are valuable since they allow for more diversity in generating new problems for systematic testing.

Table 4
The obtained p-values when comparing the difference in distribution between the benchmark problem against the generated instances by perturbation and DACE (excluding ZDT for DACE). Significant differences at an α level of 0.05 are bold, denoting wide applicability.

Problem	GrEA		HypE		IBEA		MOEA/D		NSGA-II		NSGA-III		RVEA		SPEA2	
	Pert.	DACE	Pert.	DACE	Pert.	DACE	Pert.	DACE	Pert.	DACE	Pert.	DACE	Pert.	DACE	Pert.	DACE
ZDT1	.597	–	**.000**	–	.638	–	**.000**	–	.751	–	**.022**	–	**.001**	–	**.000**	–
ZDT2	**.000**	–	**.000**	–	**.000**	–	**.000**	–	**.000**	–	**.000**	–	**.000**	–	**.000**	–
ZDT3	.226	–	**.000**	–	**.011**	–	.262	–	.054	–	.450	–	.178	–	.069	–
ZDT4	.262	–	**.000**	–	**.003**	–	.072	–	.348	–	.101	–	.870	–	**.013**	–
ZDT6	.459	–	.478	–	**.026**	–	**.000**	–	**.029**	–	.067	–	.061	–	**.003**	–
DTLZ1	**.000**	**.000**	**.000**	**.000**	**.003**	**.000**	**.000**	**.000**	**.003**	**.000**	**.005**	**.000**	**.008**	**.000**	**.001**	**.000**
DTLZ2	.226	**.015**	**.000**	**.000**	.731	.817	.226	.567	**.047**	**.001**	.209	**.000**	.965	**.022**	.110	**.000**
DTLZ3	.054	**.000**	**.04**	**.000**	**.000**	.200	**.000**	**.000**	**.000**	**.001**	**.006**	**.000**	**.001**	**.000**	**.002**	**.001**
DTLZ4	.423	.856	**.001**	.060	.517	.365	.406	.409	**.047**	.390	.423	.150	.155	.559	**.000**	**.008**
DTLZ5	.900	.559	**.000**	**.000**	.389	.499	.670	**.016**	.638	**.000**	.690	.948	**.022**	**.000**	**.022**	**.000**
DTLZ6	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
DTLZ7	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.003**	**.000**	**.000**	**.000**	**.000**	**.000**	**.001**	**.000**
WFG1	**.000**	**.000**	–	–	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	–	–	–	–	–	–
WFG2	**.003**	**.000**	**.000**	**.000**	**.000**	**.000**	**.003**	**.000**	**.003**	**.012**	**.029**	**.000**	**.029**	**.002**	**.002**	**.000**
WFG3	.249	**.013**	.209	**.027**	.900	.121	.110	**.000**	.957	.789	.203	**.022**	.296	.153	.067	**.005**
WFG4	**.004**	**.000**	**.002**	**.000**	.147	**.000**	.198	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
WFG5	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
WFG6	**.044**	.079	.893	.972	.226	.261	.517	.318	**.011**	**.001**	.282	.371	.237	.075	.113	.067
WFG7	.110	**.002**	**.000**	**.000**	.469	.471	**.015**	**.000**	**.005**	**.000**	**.000**	**.000**	**.002**	**.000**	**.000**	**.000**
WFG8	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
WFG9	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**

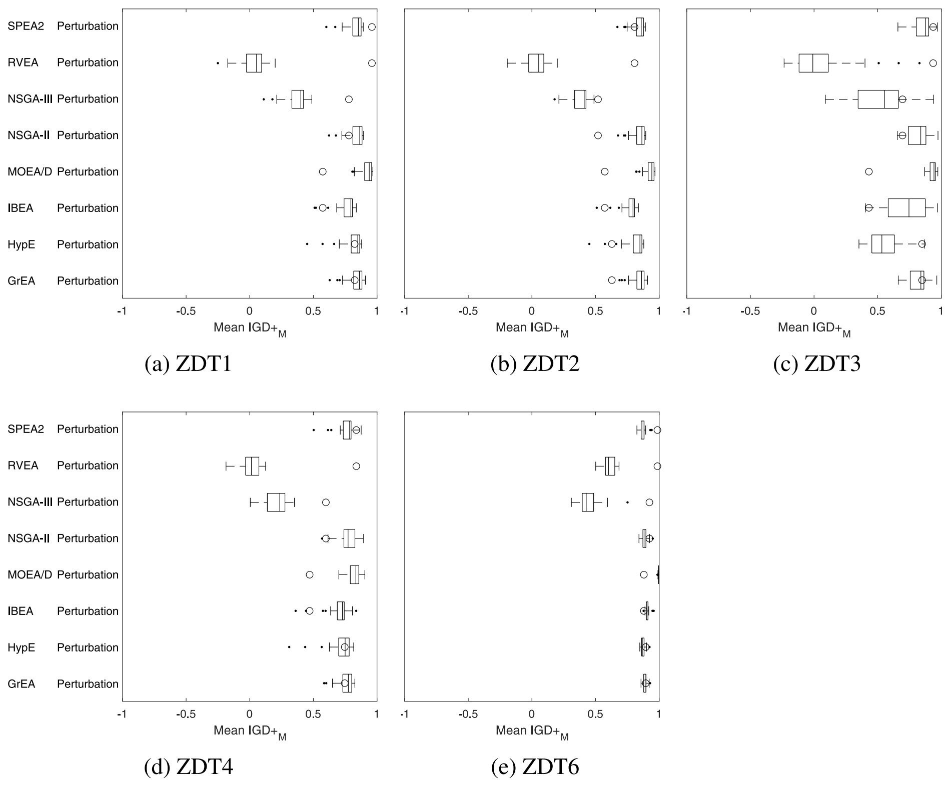


Fig. 3. Boxplots showing the potential impact of parameter variation on instance difficulty, measured by mean IGD^+_M for generated instances from the ZDT problem suite. The IGD^+_M values for the original problem are denoted by open circles. Outliers are denoted by dots.

4. Results

In this section we present the direct and indirect measures – in absolute and relative terms – of algorithm and problem robustness, including statistical testing for significance.

4.1. Algorithm robustness: Absolute performance

Algorithms are considered widely applicable if they show good performance on a large range of instances within a problem class, and fallible if their performance metric (IGD^+_M) varies greatly across these instances [18]. It is important to discuss the notion of problem difficulty or hardness for an algorithm, in order to establish if the absolute performance value reflects an algorithm finding a problem more or less challenging. It should be noted that while we refer to difficulty or hardness, we do not establish a threshold value, and instead refer to difficulty relative to performance on the original benchmark problem. The table of the p-values for each test suite can be found in Table 4 which can be

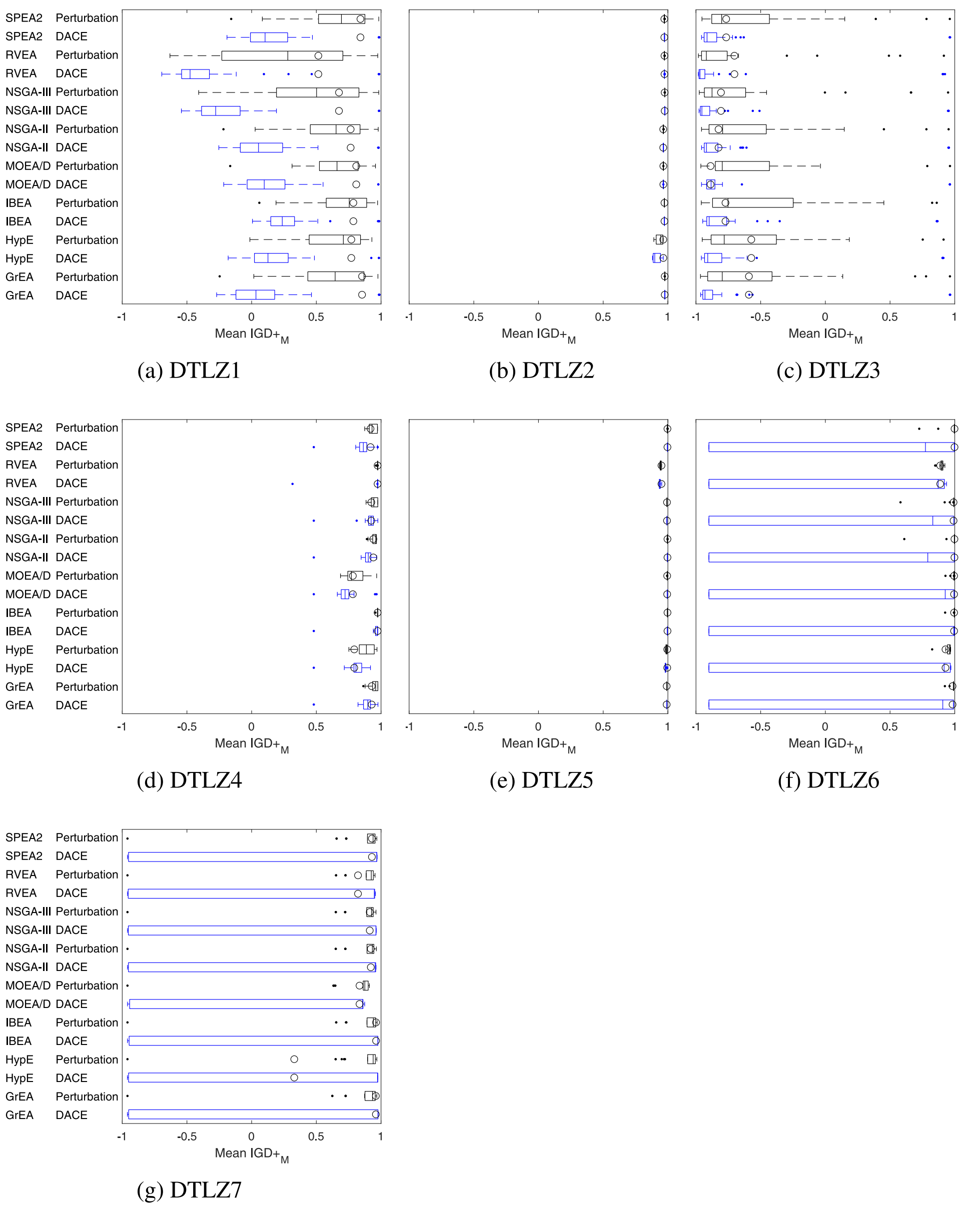


Fig. 4. Boxplots showing the potential impact of parameter variation on generating difficulty, measured by mean IGD^+_M for generated instances from the DTLZ problem suite. The IGD^+_M values for the original problem are denoted by open circles. Outliers are denoted by dots. The performance of algorithms on the DACE instances are shown in blue.

used as a simultaneous measure of wide applicability and fallibility by comparing the distributions of the original benchmark against the newly generated instances.

4.1.1. *ZDT suite*

The α parameter in the ZDT suite has a direct effect on the degree of convexity or concavity. The boxplots are shown in Fig. 3. For ZDT1, ZDT3 and ZDT6, the boxplots show that most algorithms find the original problem easier. In contrast, the benchmark ZDT2 is found to be more difficult than the alternative instances by all algorithms except NSGA-II and RVEA. ZDT4 is shown to easier or harder depending on algorithm. RVEA is a strong performer on the original benchmark problems, but shows poorer performance in any other perturbed instance.

Overall, the observed variation in IGD^+_M is small in only ZDT6, with larger variations for some algorithms (e.g RVEA and NSGA-III). As such, algorithms are considered fallible for all but ZDT6. Setting the IGD^+_M of the original benchmark as the threshold value, it is clear that most algorithms perform above this threshold for all problems except ZDT2 and ZDT3. Thus, for this suite, algorithms are widely applicable on all problems except ZDT2 and ZDT3. Table 4 confirms that ZDT4 and ZDT6 are widely applicable, with p-values above 0.05.

4.1.2. *DTLZ suite*

Within the original benchmarks of the DTLZ suite, all problems except DTLZ3 are easy, as shown in Fig. 4. Of these, Table 4 shows only DTLZ2, DTLZ4 and DTLZ5 are unchanging in absolute difficulty and therefore algorithms widely applicable on them, being easy problems for all algorithms with little variation in performance created by perturbations and DACE. Variation in DTLZ4 is able to introduce slightly more difficulty. In both DTLZ6 and DTLZ7, most perturbations result in similar performance. However, the DACE instances are able to introduce more difficulty and discrimination across performance. DTLZ1 and DTLZ3 are shown to be hard and discriminating but fallible for algorithms given the

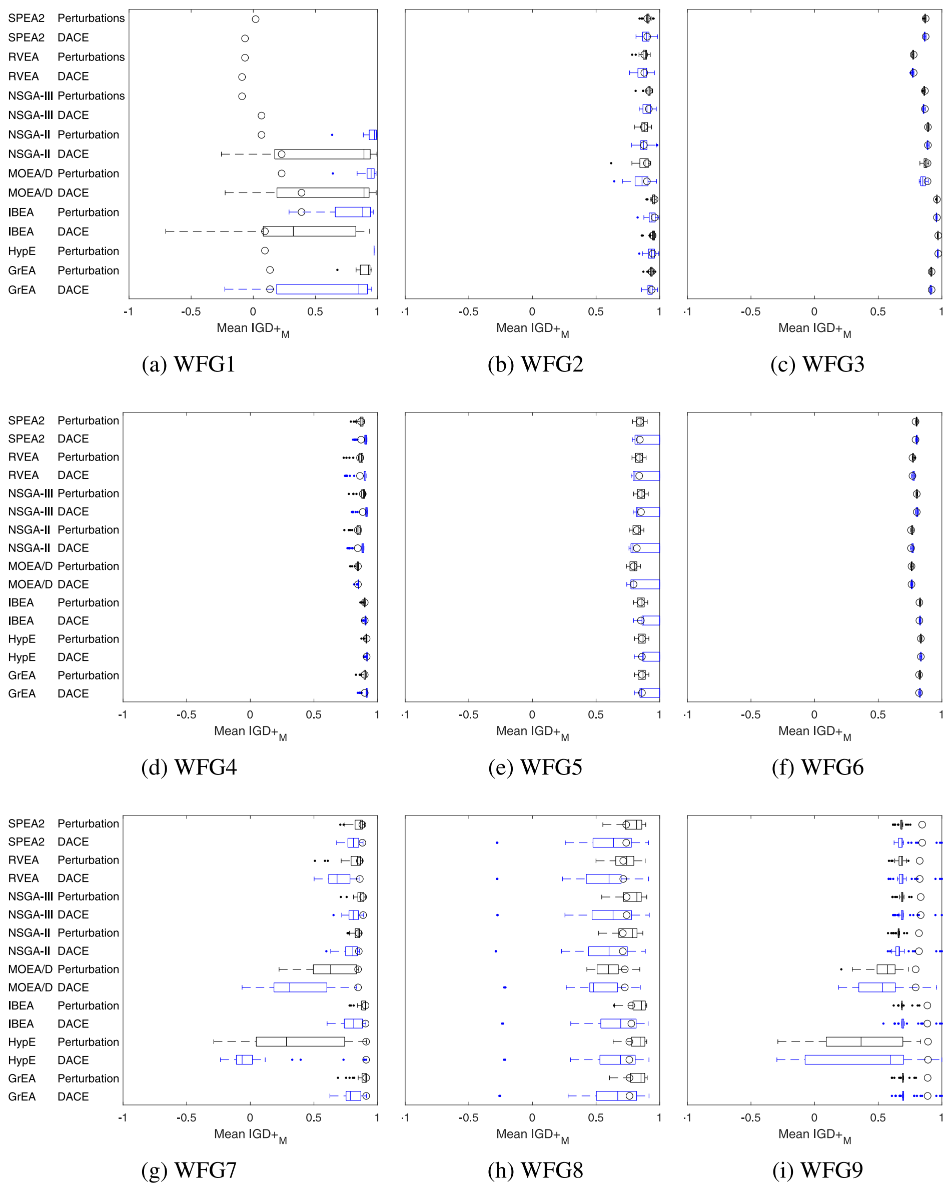


Fig. 5. Boxplots showing the potential impact of parameter variation on generating difficulty, measured by mean IGD_M^+ for generated instances from the WFG problem suite. The IGD_M^+ values for the original problem are denoted by open circles. Outliers are denoted by dots. The performance of algorithms on the DACE instances are shown in blue.

possibility of creating instances where algorithms perform worse than the threshold (the IGD_M^+ of the original benchmark). The performance of all algorithms on the benchmark DTLZ3 is originally similar. However, with the introduction of new instances, a larger range of discrimination between algorithms is observed.

The majority of the original DTLZ suite is easy for our portfolio of algorithms, but varies once the construction parameters are varied, and considerably harder instances can be found. Algorithms are not widely applicable on DTLZ1, DTLZ3, DTLZ6 and DTLZ7 since there are many problems performing worse than the threshold value. Given the range in IGD_M^+, they are also fallible. The newly generated problems provide evidence that strong algorithm performance on the benchmark may not reflect strong performance across all problem instances on this popular suite.

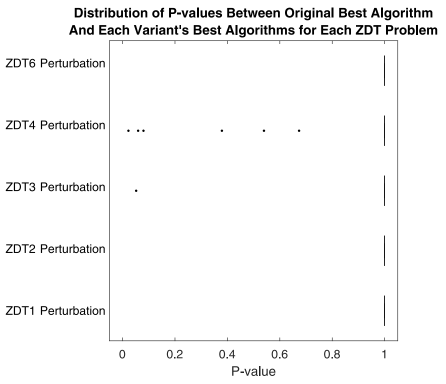
4.1.3. WFG suite

Fig. 5 shows the WFG suite has more stable construction than that of DTLZ, with WFG2-WFG6 being consistent in difficulty. Problem instances generated by perturbation on these are unable to generate more difficulty in many of the problems. WFG1, WFG7, WFG8 and WFG9 prove the existence of varying difficulty. However, we do observe the same pattern of the benchmark instances in the suite being easy in all but WFG1. HypE, NSGA-III, RVEA and SPEA2 were unable to run for WFG1 perturbations due to the algorithms being unable to solve the generated instances (complex numbers forming part of the solutions and guiding the algorithms into complex space).

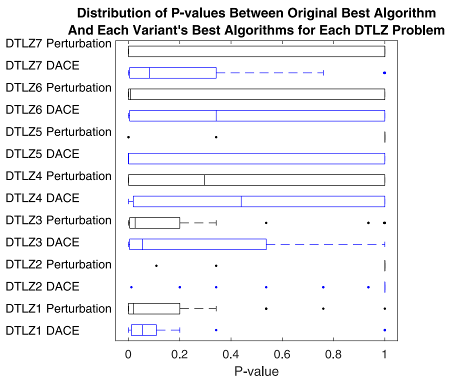
Interestingly, WFG3 and WFG6 both share the same single construction parameter p_1, and neither are able to generate more difficulty. This suggests that independently, p_1 has no relation to difficulty. Table 4 confirms that these two are the only problems which are non-fallible. Both WFG7 and WFG8 also share identical parameters, with only the order of transitions being different. Both show promise in generating difficulty, with WFG7 showing more distinction between algorithms. Algorithms are widely applicable on all but WFG1, WFG7, WFG8 and WFG9 which show many instances with performance below that of the original benchmark.

Table 5
Average feature vector distance and standard deviation after rescaling, between instances for each problem in 2*d* t-SNE projection.

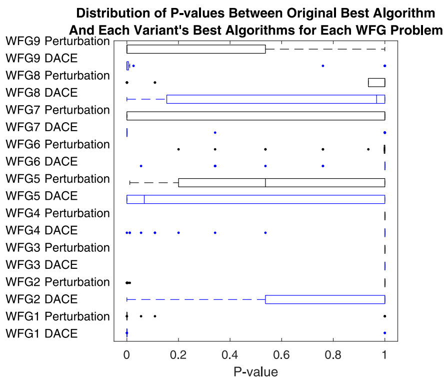
Problem	Mean	Std Dev.	Problem	Mean	Std Dev.	Problem	Mean	Std Dev.
ZDT1	0.103	0.013	DTLZ1	0.076	0.158	WFG1	0.741	0.246
ZDT3	0.403	0.476	DTLZ2	0.113	0.065	WFG2	0.220	0.230
ZDT4	0.044	0.013	DTLZ3	0.108	0.197	WFG3	0.145	0.091
ZDT6	0.155	0.625	DTLZ4	0.155	0.264	WFG4	0.270	0.076
			DTLZ5	0.161	0.097	WFG5	0.324	0.454
			DTLZ6	0.242	0.288	WFG6	0.143	0.066
			DTLZ7	0.220	0.147	WFG7	1.044	0.263
						WFG8	0.329	0.122
						WFG9	0.971	0.323



(a) ZDT



(b) DTLZ



(c) WFG

Fig. 6. The *p*-value distribution of the Mann–Whitney U test when comparing between the performance of 30 runs of the best algorithm on the original problem, and the best algorithm for each problem variant on: (a) ZDT problems. (b) DTLZ problems. (c) WFG problems.

Despite the WFG suite originally being composed to be more difficult, the performances of all algorithms within our portfolio are quite high, with WFG1 being the only original problem offering challenging difficulty. Additionally, very little discrimination is visible among the algorithms within the portfolio. With the introduction of variation, more diversity in problem difficulty is evident. This again shows algorithm weaknesses across different instances of the same problem.

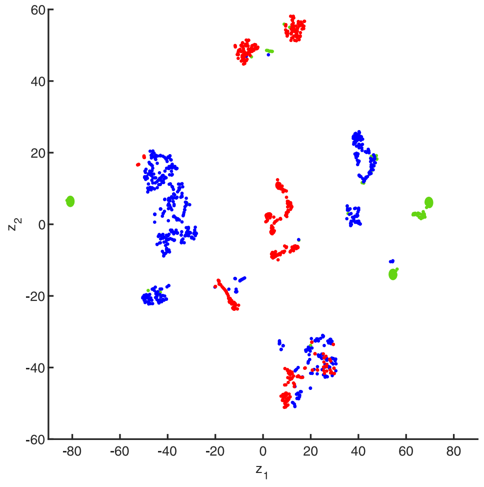
4.2. Algorithm robustness: Relative performance

Best-ranking algorithm stability is observed for problems where the best algorithm on the original problem instance (with default construction parameters) remains unchanged in ranking when new instances are introduced. We compare the best performing algorithm on the original problem with the best on each problem instance to evaluate ranking changes. The distribution of the *p*-values for each test suite can be found in Fig. 6.

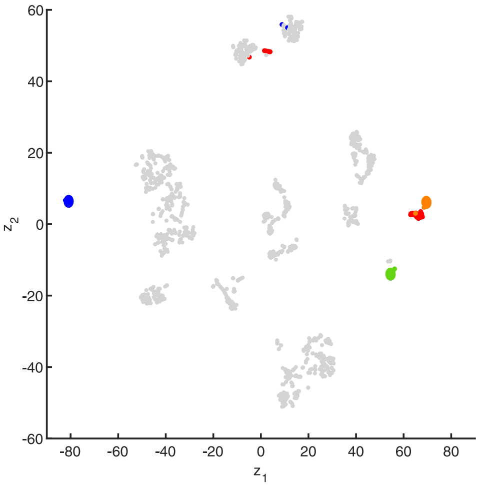
Fig. 6(a) of the ZDT suite shows algorithm stability with almost all variants sharing the same ranking for the best algorithm on the original problem. Change in ranking is only observed in few ZDT3 and ZDT4 instances.

For the DTLZ suite, most problems are not consistent for relative difficulty, and thus unstable for algorithm performance. Only DTLZ2 shows the best algorithm remaining the same across almost all variants. For DTLZ5 there are changing ranks observed in the DACE variants. At least 25% of instances for all other problems show ranking changes, but are consistent in the majority of instances.

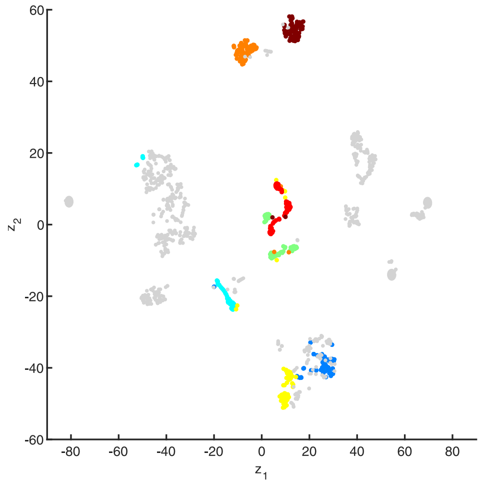
Similarly to ZDT and DTLZ, the majority of problems in the WFG suite are unstable. The best algorithm on WFG3, WFG4 and WFG6 almost never changes, but for WFG1, WFG2, WFG5, WFG7 and WFG9 the rankings change in at least 25% of instances. WFG1 shows the most extreme change, with the best algorithm on each instance differing from the original best in most instances from LHS and the DACE model generation. WFG5 and WFG8 show minor changes in ranking.



(a) All suites

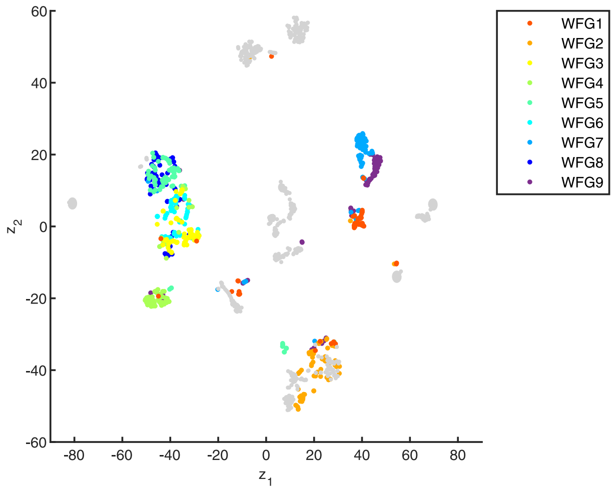


(b) ZDT



(c) DTLZ

Fig. 7. The distribution of landscape features for different instances of benchmark problems, visualised using t-SNE. ZDT1 and ZDT2 share the same points, as they are identical variations to ZDT1 (since they are the same problem with a different α parameter).



(d) WFG

Fig. 7. (*continued*).

Table 6
Summary of types of robustness satisfied by each problem.

Problem	Robustness		
	Absolute	Relative	Landscape
ZDT1		✓	✓
ZDT2		✓	✓
ZDT3	✓	✓	
ZDT4	✓	✓	✓
ZDT6	✓	✓	
DTLZ1			
DTLZ2	✓	✓	✓
DTLZ3			
DTLZ4	✓		
DTLZ5	✓		✓
DTLZ6			
DTLZ7			✓
WFG1			
WFG2	✓		✓
WFG3	✓	✓	✓
WFG4	✓	✓	
WFG5	✓		
WFG6	✓	✓	✓
WFG7			
WFG8			
WFG9			

4.3. Landscape robustness

Table 5 provides the rescaled average distance and standard deviation of the feature vectors of the benchmark problems and their new instances, using the eight features described in Table 3. Small mean distances of ZDT1, ZDT2 and ZDT4 demonstrate that these problems are not particularly tuneable. While ZDT6 shows a small mean distance, the standard deviation suggests that it is possible to generate instances with a different feature landscape. ZDT3 is both large in average distance and standard deviation, and thus is tuneable. DTLZ1, DTLZ3, DTLZ4 and DTLZ6 all show larger deviance than that of their average distance, much like ZDT6. DTLZ2, DTLZ5 and DTLZ7 seem to be non-tuneable. For the WFG suite, only WFG3, WFG4 and WFG6 are shown to have robust landscapes, making them less tuneable.

We further investigate the landscape robustness by visualising the feature vectors of each problem variant in a dimensional reduced 2*d* space, and observing whether any clusters have formed. Fig. 7 shows the projection of all problem variants using t-SNE, where we see that most of the problem variants from ZDT1, ZDT2

Table 7
Suggested construction parameter alternatives for problems which are non-robust for at least two of absolute, relative and landscape measures, based on the setup which maximises the Euclidean distance between the variant and the original benchmark on the t-SNE plot.

Problem	Suggested	Original
DTLZ1	$p_1 = 10$	$p_1 = 10$
	$p_2 = 0.9451$	$p_2 = 0.5$
	$p_3 = 81.5176$	$p_3 = 100$
DTLZ3	$p_1 = 1$	$p_1 = 10$
	$p_2 = 0.3623$	$p_2 = 0.5$
	$p_3 = 64.9709$	$p_3 = 100$
DTLZ4	$p_1 = 8.3135$	$p_1 = 10$
	$p_2 = 0.3156$	$p_2 = 0.5$
DTLZ6	$p_1 = 0.0438$	$p_1 = 0.1$
DTLZ7	$p_1 = 50$	$p_1 = 3$
	$p_2 = 1$	$p_2 = 1$
WFG1	$p_1 = 0.9656$	$p_1 = 0.35$
	$p_2 = 0.9984$	$p_2 = 0.8$
	$p_3 = 0.9922$	$p_3 = 0.75$
	$p_4 = 0.9335$	$p_4 = 0.85$
	$p_5 = 0.0109$	$p_5 = 0.02$
	$p_6 = 9.6587$	$p_6 = 1$
	$p_7 = 10$	$p_7 = 5$
WFG5	$p_1 = 0.5292$	$p_1 = 0.35$
	$p_2 = 0.1$	$p_2 = 0.001$
	$p_3 = 0.01$	$p_3 = 0.05$
WFG7	$p_1 = 0.9900$	$p_1 = 0.35$
	$p_2 = 0.0100$	$p_2 = 0.98/49.98$
	$p_3 = 0.6543$	$p_3 = 0.02$
	$p_4 = 0.0100$	$p_4 = 50$
WFG8	$p_1 = 0.3376$	$p_1 = 0.35$
	$p_2 = 0.4167$	$p_2 = 0.98/49.98$
	$p_3 = 0.4952$	$p_3 = 0.02$
	$p_4 = 86.8786$	$p_4 = 50$
WFG9	$p_1 = 0.1512$	$p_1 = 0.35$
	$p_2 = 0.8386$	$p_2 = 0.98/49.98$
	$p_3 = 2.4272$	$p_3 = 0.02$
	$p_4 = 56.4503$	$p_4 = 50$
	$p_5 = 0.0953$	$p_5 = 0.001$
	$p_6 = 0.0704$	$p_6 = 0.05$
	$p_7 = 94$	$p_7 = 30$
	$p_8 = 8.6592$	$p_8 = 95$

and ZDT4 are closely clustered, while ZDT3 and ZDT6 show each have two separate groupings of problems. DTLZ1, DTLZ3, DTLZ4 mostly exist within one cluster, but have a few instances which are further away. DTLZ2 has one cluster but shows a lambda shape, while DTLZ5 shows a hook shape, suggesting that both these problems, given their single construction parameter, are confined within their respective regions. DTLZ6 and DTLZ7 are clustered closely together. As expected, compared to the ZDT problems which mostly have only one α parameter (except for ZDT3), the DTLZ suite clusters show larger variation in landscapes. This is also visible in the WFG suites. WFG2, WFG3, WFG6 and WFG8 largely exist within clear clusters. While Table 5 did not demonstrate much diversity for WFG4, upon investigating the visualisation we see that WFG4 actually occupies three separate clusters. There is also a great deal of diversity and overlap in the space across the WFG suite, which is promising for developing more diverse problems and instances in future. Overall, WFG1, WFG4, WFG5, WFG7 and WFG9 are capable of generating a wider range of problems beyond the default parameter setting, and therefore demonstrate tuneability and an absence of landscape robustness.

5. Extending onto recent suites

The ZDT, DTLZ and WFG suites are older suites, and therefore we further extend our analyses to two suites that are more recent, with parameterisable construction. Both the problems generated in the RMMEDA [28] and IMMOEA [25] papers are chosen for our analyses and will henceforth be referred to as the RMMEDA and IMMOEA suites. Both suites are constructed using the ZDT1, ZDT2 and ZDT6 problems, with different variable linkages and g functions included. As such, we only perturb the α construction parameter to be consistent with our treatment of the ZDT problems. We consider these suites an interesting extension since they are ZDT problems with newly introduced characteristics (linkages, landscape search functions) intended to inject additional problem difficulty. Problems 4 and 8 for each suite are DTLZ2 variants, but are excluded from our analysis since the

Table 8
The obtained p-values when comparing the difference in distribution between the benchmark problem against the generated instances by perturbation. Significant differences at an α level of 0.05 are bold, denoting wide applicability.

Problem	GrEA	HypE	IBEA	MOEA/D	NSGA-II	NSGA-III	RVEA	SPEA2
IMMOEA_F1	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
IMMOEA_F2	**0.00**	**0.00**	**0.00**	**0.03**	**0.00**	**0.00**	**0.00**	**0.00**
IMMOEA_F3	**0.00**	0.45	**0.00**	0.17	0.07	0.15	0.81	**0.00**
IMMOEA_F5	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
IMMOEA_F6	**0.00**	**0.00**	**0.00**	0.15	**0.00**	**0.00**	**0.00**	**0.00**
IMMOEA_F7	**0.00**	0.02	**0.00**	**0.01**	**0.00**	**0.00**	**0.00**	**0.00**
IMMOEA_F9	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
IMMOEA_F10	0.54	0.12	1	0.15	0.14	0.94	0.21	0.35
RMMEDA_F1	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
RMMEDA_F2	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
RMMEDA_F3	**0.00**	**0.00**	**0.00**	0.19	**0.00**	**0.01**	0.02	**0.01**
RMMEDA_F5	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
RMMEDA_F6	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
RMMEDA_F7	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
RMMEDA_F9	**0.00**	**0.00**	**0.01**	0.11	**0.03**	0.07	**0.00**	**0.04**
RMMEDA_F10	0.5	0.64	0.85	**0.01**	0.15	0.08	0.62	0.18

Table 9
Average feature vector distance and standard deviation after rescaling, between instances for each problem in 2d t-SNE projection.

Problem	Mean	Std Dev.
IMMOEA_F1	0.060	0.011
IMMOEA_F2	0.059	0.009
IMMOEA_F3	0.010	0.000
IMMOEA_F5	0.077	0.026
IMMOEA_F6	0.069	0.024
IMMOEA_F7	0.017	0.003
IMMOEA_F9	0.647	0.131
IMMOEA_F10	0.018	0.001
RMMEDA_F1	0.065	0.015
RMMEDA_F2	0.056	0.013
RMMEDA_F3	0.017	0.019
RMMEDA_F5	0.073	0.015
RMMEDA_F6	0.053	0.013
RMMEDA_F7	0.013	0.000
RMMEDA_F9	0.175	0.393
RMMEDA_F10	0.030	0.001

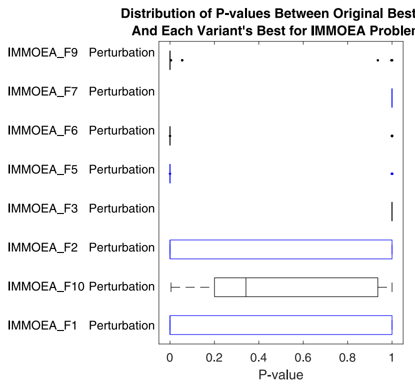
Table 10
Summary of types of robustness satisfied by each of the IMMOEA and RMMEDA problems.

	Robustness		
Problem	Absolute	Relative	Landscape
IMMOEA_F1			✓
IMMOEA_F2			✓
IMMOEA_F3	✓	✓	✓
IMMOEA_F5			✓
IMMOEA_F6			✓
IMMOEA_F7		✓	✓
IMMOEA_F9			
IMMOEA_F10	✓		✓
RMMEDA_F1		✓	✓
RMMEDA_F2		✓	✓
RMMEDA_F3			✓
RMMEDA_F5		✓	✓
RMMEDA_F6		✓	✓
RMMEDA_F7		✓	✓
RMMEDA_F9			
RMMEDA_F10	✓		✓

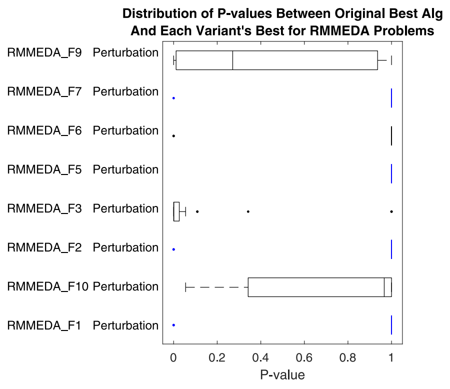
variable linkage replaces the DTLZ2's construction parameter. Table 8 confirms that IMMOEA_F3, IMMOEA_F10 and RMMEDA_F10 are widely applicable and/or non-fallible in this suite, with p-values above 0.05. In terms of relative performance, Fig. 8 shows that only IMMOEA_F3, IMMOEA_F7, RMMEDA_F1, RMMEDA_F2 and RMMEDA_F5, RMMEDA_F6, RMMEDA_F7 are stable. In the updated t-SNE plot in Fig. 9, we observe similar behaviours for all the IMMOEA, RMMEDA and ZDT suites. While there is some dispersion introduced in the visual plot, the landscape projections tend to collect together in a concentrated area. This is expected, as we have previously observed the extent of landscape change by the α parameter. Table 9 shows similar mean distances and standard deviations for all but IMMOEA_F9 and RMMEDA_F9. As such, all but these two problems are considered robust in landscape (not tuneable). Interestingly, while most of these problems are not tuneable by varying the construction parameter α, the change in g functions and linkages shift the location of these problems, compared to the ZDT problems from which they are built upon. We therefore conclude based on this evidence that the α construction parameter in the IMMOEA and RMMEDA suites tends to generate similar behaviours of robustness as with the ZDT suite, despite the additional linkages and changes in landscape functions. A summary of the robustness types is found in Table 10. It is important to consider that the landscape features discussed here may not be sufficiently capturing information regarding the change in linkages, and further investigation into appropriate features will be an important research area in the future.

6. Conclusions

Researchers use benchmark test instances to compare algorithms and draw conclusions about their relative merits. In this paper we have investigated the robustness of test suites and whether such conclusions when based on the default parameter settings for the popularly used ZDT, DTLZ and WFG benchmarks. Problems are required to be robust if broader conclusions regarding algorithm performance and suitability for problems with certain characteristics are to be accurate and reliable. A modification of the original IGD metric, IGD^+_M, was introduced as a performance metric to allow for comparison across different PFs. A methodology for evaluating problem robustness – through two indirect measures of algorithm robustness and one direct measure of landscape robustness – was presented, extending the work of Eiben and Smit [18] from Algorithm Tuning to Problem Tuning. We found evidence of algorithm weaknesses that would otherwise not be identified when using only the default benchmark instances. In most cases, variation in construction parameters led to more difficult instances within problem classes. Notably, the ranking order for the best algorithm also changed for many of these problems. This suggests that the default parameter settings have created popular benchmark test problems that may not support conclusions as robustly as required to understand algorithm strengths and weaknesses. As such, these problems could serve the research community better if additional parameter settings were explored to create more diverse instances with



(a) IMMOEA



(b) RMMEDA

Fig. 8. The *p*-value distribution of the Mann–Whitney U test when comparing between the performance of 30 runs of the best algorithm on the original problem, and the best algorithm for each problem variant on: (a) IMMOEA problems. (b) RMMEDA problems.

the same general characteristics. We have provided a table with some suggestions for alternative instances to consider for a more comprehensive evaluation of MOOP algorithms.

For the ZDT test suite, the α parameter did not greatly impact difficulty for most algorithms across ZDT3, ZDT4 and ZDT6. Within the other suites, only instances of DTLZ2, DTLZ4, DTLZ5, WFG2, WFG3, WFG4 and WFG6 remained at a similar level of difficulty. The benchmark DTLZ suites showed a larger range of difficulty than the original WFG, but the WFG problems were more consistent pertaining to algorithm performance. However, despite the goal of the WFG toolkit to generate more difficult problems, all algorithms in our portfolio found the WFG benchmark problems, except WFG1, to be easy. By exploring construction parameter variation, problem instances of increased difficulty for different algorithms were generated. As a result, we were able to observe more diversity in algorithm performance across an expanded test suite.

Hypothesis testing was performed to check if the conclusions drawn based on the ranking performance of the best algorithm on a benchmark problem would change if problem parameters were varied. Results showed that a total of nine benchmark problems were stable for algorithm ranking. DTLZ2, WFG3, WFG4, WFG6, and all of the ZDT suite, almost always had the same rank for best algorithm across their respective instances.

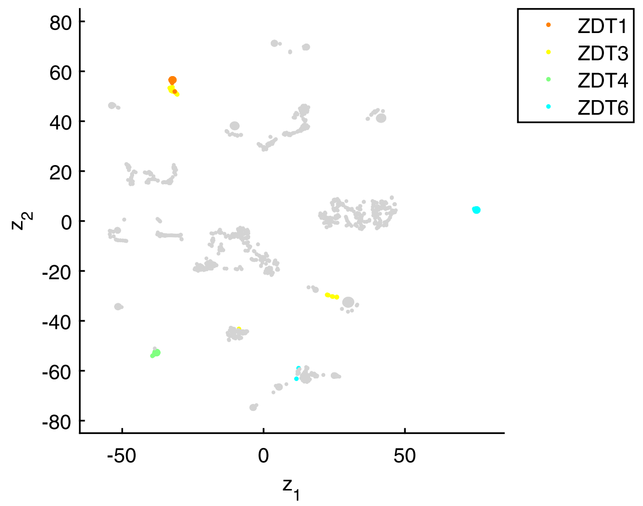
The original best algorithm on WFG1 is not shared for the majority of other instances. This highlights the dangers of drawing misleading conclusions using only the original problem with default construction parameters, especially if the conclusion attempts to discuss the suitability of an algorithm in the presence of the problem characteristic in general.

The introduction of DACE problems, while successfully introducing greater diversity of instance difficulty did not necessarily introduce more algorithm instability. This is a reasonable outcome, as the objective of each DACE model was to generate problems that were more difficult for a given algorithm, which may not affect rankings if all algorithms also struggled with these instances. It is an interesting future direction to extend the DACE approach to generate new problem instances that are simultaneously more difficulty and more discriminating of algorithm performance.

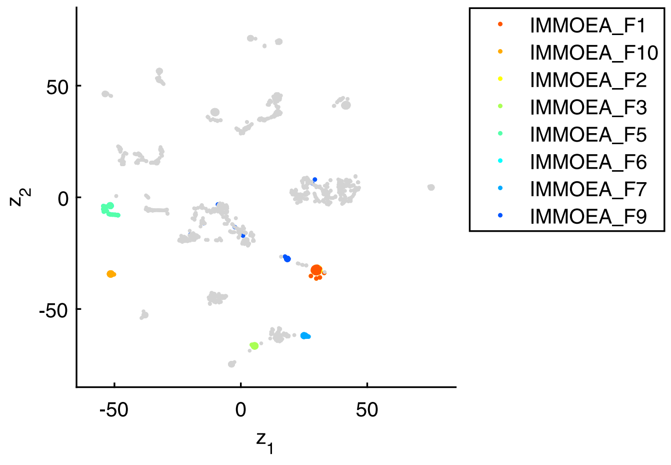
We also investigated the landscape changes using t-SNE to visualise and measure the diversity of the benchmark problems, and to evaluate problem robustness (tuneability). We observed that all problems which were non-tuneable were either widely applicable or non-fallible (or both) for algorithms. This is expected, as algorithms which exploit information on landscapes should be expected to perform consistently if it remains largely unchanged. Conversely, problems which are tuneable provide us with an opportunity to further explore algorithm performance in the presence of changes in existing problem characteristics after changing construction parameters. Interestingly, DTLZ7 was the only problem for which algorithm performance was not robust, despite evidence of landscape robustness (non-tuneability).

The robustness of test suites plays an important role in how their respective problems should be used. Table 6 summarises our findings which support that both robust and non-robust problems exist in the ZDT, DTLZ and WFG suites, and is likely the case for other suites, as demonstrated by our extension into the RMMEDA and IMMOEA suites. Ultimately, of the main suites we studied, only ZDT4, DTLZ2, WFG3 and WF6 satisfy all problem robustness requirements. Problems which are robust are advantageous because they ensure that drawing broader conclusions about algorithm performance holds, without requiring additional evaluations on alternate instances. This provides an opportunity for exploring the relationship between the properties and parameters within a problem and algorithm performance, e.g., how does an algorithm's performance change when we vary the degree of convexity in a problem, or move the global optima? Alternatively, when robustness does not hold, we are afforded the opportunity to tease out more diverse characteristics of problems and understand their influence on algorithm behaviour. As such, both types of problems – robust and non-robust – are important, but we must understand this robustness property for all test problems in order to ensure they are fit for purpose. In this study, we focused on bi- and tri-objective problems. When dealing with problems beyond three objectives, Pareto geometry may sometimes become unclear [5]. Investigation of whether the robustness types remain the same when scaling problems to higher dimensions is an interesting future direction.

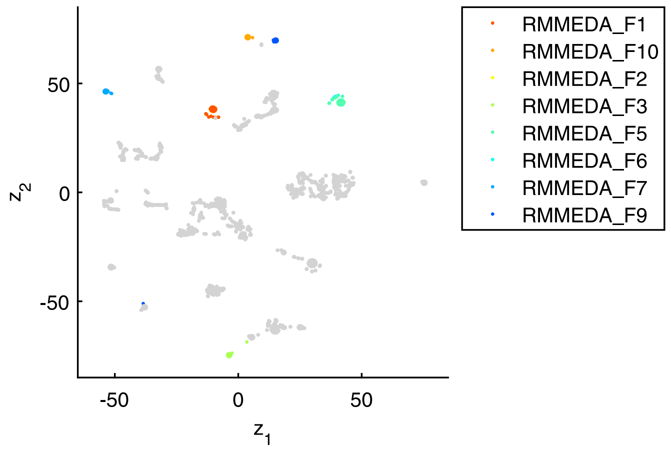
While the development of new test suites is important for providing more diversity and broadening our understanding of algorithm applicability on different properties, our study provides the basis for a more granular exploration of specific characteristics. By generating new instances, we can measure algorithm response to changes in each additional global minima or discontinuous front, instead of entirely new problems. The 2d t-SNE plots also showed that we can generate more diversity in the already existing suites. Thus combining this exploration of instances with additional suites will provide us with a comprehensive space for studying continuous multi-objective problems. We can then build instance spaces [56] which allow us to understand which features of problems affect algorithm performance, and how to best select a well-performing algorithm on a newly encountered black-box problem. By extracting features of a black-box problem



(a) ZDT



(b) IMMOEA



(c) RMMEDA

Fig. 9. The updated distribution of landscape features for different instances of benchmark problems with IMMOEA and RMMEDA problems included, visualised using t-SNE. ZDT1 and ZDT2 share the same points, as they are identical variations to ZDT1 (since they are the same problem with a different α parameter).

and mapping it to the instance space, an algorithm can be allocated based on whether it exhibits dominant performance over that region. Furthermore, we provide suggestions in Table 7 for alternative parameter choices for problems which were not found to be robust for at least two of absolute, relative and landscape measures. These are made by identifying the parameter choice which is located furthest from the original problem in the t-SNE plots in Fig. 7. While these are not the only choices which will lead to further understanding of performance, they provide a basis upon which algorithms can be studied on a different version (instance) of the same problem class, to support the need for generalisable conclusions [17].

This paper demonstrates the importance of using a diverse set of problem instances when evaluating algorithms to maximise the information gained from existing problem classes. When more challenging problem instances were generated, we have observed evidence of variation in landscape, algorithm performance, as well as changing ranks of the best algorithm. Many of the problems in the test suites, when considered using only their default parameter configuration, do not support algorithm robustness on given characteristics and consequently, the conclusions they draw. Diverse problem instances would serve the research community well in studying the influence of characteristics on algorithm performance. While we have focused on generating new instances within the ZDT, DTLZ and WFG test suites here, new diverse problems should continue to be generated to provide different challenges for algorithms. Furthermore, our study focused on bi- and tri-objective problems. However, we consider the assessment of robustness and diversity in higher dimensions to be an important future research direction.

CRediT authorship contribution statement

Estefania Yap: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Mario Andrés Muñoz:** Conceptualization, Methodology, Software, Writing - original draft, Writing - review & editing, Supervision. **Kate Smith-Miles:** Conceptualization, Methodology, Writing - original draft, Writing - review & editing, Supervision, Funding acquisition.

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.

Acknowledgements

Funding for this research was provided by the Australian Research Council through the Australian Laureate Fellowship FL140100012.

References

[1] J.C. Culberson, On the futility of blind search: An algorithmic view of “no free lunch”, Evol. Comput. 6 (2) (1998) 109–127, http://dx.doi.org/10.1162/evco.1998.6.2.109.

[2] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82, http://dx.doi.org/10.1109/4235.585893.

[3] K. Deb, Multi-objective genetic algorithms: Problem difficulties and construction of test problems, Evol. Comput. 7 (3) (1999) 205–230.

[4] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, Evol. Comput. 8 (2) (2000) 173–195, http://dx.doi.org/10.1162/106365600568202.

[5] 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, http://dx.doi.org/10.1109/TEVC.2005.861417.

[6] K.E. Parsopoulos, D.K. Tasoulis, M.N. Vrahatis, et al., Multiobjective optimization using parallel vector evaluated particle swarm optimization, in: Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, Vol. 2, Acta Press, 2004, pp. 823–828.

[7] G. Fang, Y. Guo, X. Wen, X. Fu, X. Lei, Y. Tian, T. Wang, Multi-objective differential evolution-chaos shuffled frog leaping algorithm for water resources system optimization, Water Resour. Manag. 32 (12) (2018) 3835–3852, http://dx.doi.org/10.1007/s11269-018-2021-6.

[8] P. Kaufmann, M. Platzner, Combining local and global search: A multi-objective evolutionary algorithm for cartesian genetic programming, in: S. Stepney, A. Adamatzky (Eds.), Inspired By Nature: Essays Presented To Julian F. Miller on the Occasion of His 60th Birthday, Springer International Publishing, Cham, 2018, pp. 175–194, http://dx.doi.org/10.1007/978-3-319-67997-6_8.

[9] R. Landa, G. Lárraga, G. Toscano, Use of a goal-constraint-based approach for finding the region of interest in multi-objective problems, J. Heuristics 25 (1) (2019) 107–139, http://dx.doi.org/10.1007/s10732-018-9387-8.

[10] M. Tabatabaei, M. Hartikainen, K. Sindhya, J. Hakanen, K. Miettinen, An interactive surrogate-based method for computationally expensive multi-objective optimisation, J. Oper. Res. Soc. (2018) 1–17, http://dx.doi.org/10.1080/01605682.2018.1468860.

[11] L. Tang, X. Wang, Z. Dong, Adaptive multiobjective differential evolution with reference axis vicinity mechanism, IEEE Trans. Cybern. (2018) 1–15, http://dx.doi.org/10.1109/TCYB.2018.2849343.

[12] F. Wang, H. Zhang, Y. Li, Y. Zhao, Q. Rao, External archive matching strategy for MOEA/D, Soft Comput. 22 (23) (2018) 7833–7846, http://dx.doi.org/10.1007/s00500-018-3499-9.

[13] J.E. Fieldsend, T. Chugh, R. Allmendinger, K. Miettinen, A feature rich distance-based many-objective visualisable test problem generator, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2019, pp. 541–549.

[14] H. Ishibuchi, Y. Setoguchi, H. Masuda, Y. Nojima, Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes, IEEE Trans. Evol. Comput. 21 (2) (2017) 169–190.

[15] 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) (2014) 577–601, http://dx.doi.org/10.1109/TEVC.2013.2281535.

[16] J. Maltese, B.M. Ombuki-Berman, A.P. Engelbrecht, A scalability study of many-objective optimization algorithms, IEEE Trans. Evol. Comput. 22 (1) (2018) 79–96, http://dx.doi.org/10.1109/TEVC.2016.2639360.

[17] T. Liao, D. Molina, T. Stützle, Performance evaluation of automatically tuned continuous optimizers on different benchmark sets, Appl. Soft Comput. 27 (2015) 490–503, http://dx.doi.org/10.1016/j.asoc.2014.11.006, http://www.sciencedirect.com/science/article/pii/S1568494614005584.

[18] A.E. Eiben, S.K. Smit, Parameter tuning for configuring and analyzing evolutionary algorithms, Swarm Evol. Comput. 1 (1) (2011) 19–31.

[19] H. Ishibuchi, H. Masuda, Y. Tanigaki, Y. Nojima, Modified distance calculation in generational distance and inverted generational distance, in: A. Gaspar-Cunha, C. Henggeler Antunes, C.C. Coello (Eds.), Evolutionary Multi-Criterion Optimization, Springer International Publishing, Cham, 2015, pp. 110–125.

[20] R. Tanabe, H. Ishibuchi, An easy-to-use real-world multi-objective optimization problem suite, Appl. Soft Comput. 89 (2020) 106078, http://dx.doi.org/10.1016/j.asoc.2020.106078, http://www.sciencedirect.com/science/article/pii/S1568494620300181.

[21] 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.

[22] Q. Zhang, W. Liu, H. Li, The performance of a new version of MOEA/D on CEC09 unconstrained mop test instances, in: 2009 IEEE Congress on Evolutionary Computation, 2009, pp. 203–208.

[23] H. Li, Q. Zhang, J. Deng, Biased multiobjective optimization and decomposition algorithm, IEEE Trans. Cybern. 47 (1) (2017) 52–66.

[24] L. Ke, Q. Zhang, R. Battiti, MOEA/D-ACO: a multiobjective evolutionary algorithm using decomposition and antcolony, IEEE Trans. Cybern. 43 (6) (2013) 1845–1859.

[25] R. Cheng, Y. Jin, K. Narukawa, B. Sendhoff, A multiobjective evolutionary algorithm using Gaussian process-based inverse modeling, IEEE Trans. Evol. Comput. 19 (6) (2015) 838–856.

[26] Y. Tian, X. Zhang, C. Wang, Y. Jin, An evolutionary algorithm for large-scale sparse multiobjective optimization problems, IEEE Trans. Evol. Comput. 24 (2) (2020) 380–393.

[27] H. Li, Q. Zhang, Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II, IEEE Trans. Evol. Comput. 13 (2) (2009) 284–302.

[28] Q. Zhang, A. Zhou, Y. Jin, RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm, IEEE Trans. Evol. Comput. 12 (1) (2008) 41–63, http://dx.doi.org/10.1109/TEVC.2007.894202.

[29] K. Deb, Multi-objective optimization, in: E.K. Burke, G. Kendall (Eds.), Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, Springer US, Boston, MA, 2014, pp. 403–449, http://dx.doi.org/10.1007/978-1-4614-6940-7_15.

[30] M. Reyes-Sierra, C.A. Coello Coello, et al., Multi-objective particle swarm optimizers: A survey of the state-of-the-art, Int. J. Comput. Intell. Res. 2 (3) (2006) 287–308.

[31] A. Trivedi, D. Srinivasan, K. Sanyal, A. Ghosh, A survey of multiobjective evolutionary algorithms based on decomposition, IEEE Trans. Evol. Comput. 21 (3) (2017) 440–462, http://dx.doi.org/10.1109/TEVC.2016.2608507.

[32] A. Fischbach, T. Bartz-Beielstein, Improving the reliability of test functions generators, Appl. Soft Comput. 92 (2020) 106315, http://dx.doi.org/10.1016/j.asoc.2020.106315, http://www.sciencedirect.com/science/article/pii/S1568494620302556.

[33] W.B. Langdon, R. Poli, Evolving problems to learn about particle swarm optimizers and other search algorithms, IEEE Trans. Evol. Comput. 11 (5) (2007) 561–578, http://dx.doi.org/10.1109/TEVC.2006.886448.

[34] Y. Tanigaki, Y. Nojima, H. Ishibuchi, Meta-optimization based multi-objective test problem generation using WFG toolkit, in: 2016 IEEE Congress on Evolutionary Computation (CEC), 2016, pp. 2768–2775, http://dx.doi.org/10.1109/CEC.2016.7744138.

[35] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable multi-objective optimization test problems, in: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), Vol. 1, 2002, pp. 825–830, http://dx.doi.org/10.1109/CEC.2002.1007032.

[36] R. Cheng, Y. Jin, M. Olhofer, B. Sendhoff, A reference vector guided evolutionary algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 20 (5) (2016) 773–791, http://dx.doi.org/10.1109/TEVC.2016.2519378.

[37] S. Huband, L. Barone, L. While, P. Hingston, A scalable multi-objective test problem toolkit, in: C.A. Coello Coello, A. Hernández Aguirre, E. Zitzler (Eds.), Evolutionary Multi-Criterion Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 280–295.

[38] R. Cheng, Y. Jin, M. Olhofer, B. Sendhoff, Test problems for large-scale multiobjective and many-objective optimization, IEEE Trans. Cybern. 47 (12) (2017) 4108–4121, http://dx.doi.org/10.1109/TCYB.2016.2600577.

[39] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P.N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: A survey of the state of the art, Swarm Evol. Comput. 1 (1) (2011) 32–49, http://dx.doi.org/10.1016/j.swevo.2011.03.001, https://www.sciencedirect.com/science/article/pii/S2210650211000058.

[40] J. Zhang, L. Xing, A survey of multiobjective evolutionary algorithms, in: 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Vol. 1, 2017, pp. 93–100, http://dx.doi.org/10.1109/CSE-EUC.2017.27.

[41] Z. He, G. Yen, Comparison of many-objective evolutionary algorithms using performance metrics ensemble, Adv. Eng. Softw. 76 (Supplement C) (2014) 1–8, http://dx.doi.org/10.1016/j.advengsoft.2014.05.006, http://www.sciencedirect.com/science/article/pii/S0965997814000908.

[42] 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, http://dx.doi.org/10.1109/4235.996017.

[43] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization, in: Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, International Center for Numerical Methods in Engineering, Athens, Greece, 2001, pp. 95–100.

[44] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731, http://dx.doi.org/10.1109/TEVC.2007.892759.

[45] S. Yang, M. Li, X. Liu, J. Zheng, A grid-based evolutionary algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 17 (5) (2013) 721–736, http://dx.doi.org/10.1109/TEVC.2012.2227145.

[46] J. Bader, E. Zitzler, Hype: An algorithm for fast hypervolume-based many-objective optimization, Evol. Comput. 19 (1) (2011) 45–76, http://dx.doi.org/10.1162/EVCO_a_00009.

[47] E. Zitzler, S. Künzli, Indicator-based selection in multiobjective search, in: Parallel Problem Solving from Nature - PPSN VIII, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 832–842.

[48] N. Riquelme, C.V. Lücken, B. Baran, Performance metrics in multi-objective optimization, in: 2015 Latin American Computing Conference (CLEI), 2015, pp. 1–11, http://dx.doi.org/10.1109/CLEI.2015.7360024.

[49] S. Jiang, Y.S. Ong, J. Zhang, L. Feng, Consistencies and contradictions of performance metrics in multiobjective optimization, IEEE Trans. Cybern. 44 (12) (2014) 2391–2404, http://dx.doi.org/10.1109/TCYB.2014.2307319.

[50] G.G. Yen, Z. He, Performance metric ensemble for multiobjective evolutionary algorithms, IEEE Trans. Evol. Comput. 18 (1) (2014) 131–144, http://dx.doi.org/10.1109/TEVC.2013.2240687.

[51] H. Li, Q. Zhang, J. Deng, Biased multiobjective optimization and decomposition algorithm, IEEE Trans. Cybern. 47 (1) (2017) 52–66, http://dx.doi.org/10.1109/TCYB.2015.2507366.

[52] J. Mueller, Matsumoto: The MATLAB surrogate model toolbox for computationally expensive black-box global optimization problems, 2014, ArXiv Preprint arXiv:1404.4261.

[53] D.R. Jones, M. Schonlau, W.J. Welch, Efficient global optimization of expensive black-box functions, J. Global Optim. 13 (4) (1998) 455–492, http://dx.doi.org/10.1023/A:1008306431147.

[54] J. Sacks, W.J. Welch, T.J. Mitchell, H.P. Wynn, Design and analysis of computer experiments, Statist. Sci. 4 (4) (1989) 409–423, http://www.jstor.org/stable/2245858.

[55] A. Liefooghe, F. Daolio, S. Verel, B. Derbel, H. Aguirre, K. Tanaka, Landscape-aware performance prediction for evolutionary multi-objective optimization, IEEE Trans. Evol. Comput. (2019) http://dx.doi.org/10.1109/TEVC.2019.2940828, https://hal.archives-ouvertes.fr/hal-02294201.

[56] K. Smith-Miles, D. Baatar, B. Wreford, R. Lewis, Towards objective measures of algorithm performance across instance space, Comput. Oper. Res. 45 (2014) 12–24, http://dx.doi.org/10.1016/j.cor.2013.11.015, http://www.sciencedirect.com/science/article/pii/S0305054813003389.