Informing Multiobjective Optimization Benchmark Construction Through Instance Space Analysis

Estefania Yap, Mario Andrés Muñoz, and Kate Smith-Miles, *Senior Member, IEEE*

***Abstract*—The role of carefully constructed benchmark suites in algorithm design and testing is critical. Within the continuous multiobjective optimization domain, existing suites include the general purpose ZDT, DTLZ, and WFG suites, and more recent ones specifically designed to explore the impacts of a particular problem characteristic. However, the relationship between existing suites is not clear, and the field would benefit from a "stock-take" assessment. This article investigates the coverage of current continuous multiobjective suites using the instance space analysis (ISA) methodology. Exploratory landscape analysis is used to measure critical features of each problem suite. Thereafter, we generate a 2-D visualization of the existing problem instances by locating them in the instance space, assessing their diversity, and identifying whether there are sparse areas of value to fill with new problem instances. Our findings show that the current suites are restricted in diversity when representing the entire problem instance space. We propose and evaluate three problem construction methods: 1) problem tuning; 2) toolkit hybridization; and 3) new function injection. Problem tuning is shown to generate problems surrounding existing instances, while hybridization creates problems falling between existing suites. Furthermore, utilizing the insights afforded by ISA, we show how problem features can be identified to inform the creation of new functions which fill gaps toward the boundaries of the instance space.**

***Index Terms*—Benchmark suites, experimental evaluation, instance space analysis (ISA), multiobjective optimization (MO), problem generation.**

I. Introduction

Multiobjective optimization (MO) involves the simultaneous optimization of two or more objective functions. For a problem within this class, solutions are represented as decision values which map to a vector of corresponding objective values. Optimality is defined as the best tradeoffs between conflicting objectives, demonstrated by a Pareto front (PF). Many real-world problems are multiobjective, often with the added complexity caused by the absence of mathematical models that capture the relationship between the decision variables and objective functions. Instead, such "black-box" problems require resource-intensive simulations or experiments to evaluate the objective functions based on carefully chosen samples in the decision space. Therefore, efficient search methods are required to sample promising areas of the decision space and find an acceptable set of solutions that approximate the PF. Thereafter, a single solution can be identified for implementation in real-world scenarios.

Evolutionary algorithms (EAs) are popular approaches for solving MO problems, since they maintain a set (or population) of solutions corresponding to an estimated PF. A wide variety of EAs have been developed, each following a unique strategy to generate, evaluate, and select new solutions. Such a variety of search strategies naturally leads to a spectrum of performances of algorithms, with no one algorithm dominating all others for all test problems [1]. Indeed, the strengths and weaknesses of each algorithm appear to depend on how well aligned the search strategy is to the mixture of characteristics exhibited by the problem, such as multimodality and variable interdependence [2]. The algorithm selection problem [3] attempts to learn this relationship, in order to identify the optimal algorithm for a given problem instance. However, learning to predict the best algorithm does not result in any transparency into the strengths and weaknesses of algorithms.

Contemporary approaches to the ASP rely on experimental results from an algorithm portfolio across a large set of benchmark problems, which are then presented to a machine learning algorithm as training data to learn to predict the algorithm performance based on the problem characteristics. Since real-world problems are often underrepresented in benchmark sets, synthetic problems have been designed to exhibit specific characteristics known to be challenging for algorithms [2], [4], [5], under the assumption that real-world problems, now or in the future, may also exhibit these characteristics, and we need to understand how they affect algorithm performance. Even from a purely theoretical perspective, without concern for real-world application, there is an argument for constructing synthetic instances with challenging characteristics to expose algorithm strengths and weaknesses in their entirety.

Clearly, well-constructed benchmarking studies are essential if we are to collect reliable data to help us better understand algorithm behavior in the presence of various problem characteristics, and enable robust comparisons [6]. In particular, they are necessary to support three major areas: 1) solving the ASP; 2) exploring the strengths and weaknesses of existing algorithms; and 3) leveraging the insights into algorithm

Manuscript received 8 August 2021; revised 16 December 2021, 11 March 2022, 30 May 2022, and 7 August 2022; accepted 29 August 2022. Date of publication 8 September 2022; date of current version 1 December 2022. This work was supported in part by the Australian Research Council through the Australian Laureate Fellowship under Grant FL140100012, and in part by the ARC Training Centre in Optimisation Technologies, Integrated Methodologies and Applications (OPTIMA) under Grant IC200100009. *(Corresponding author: Estefania Yap.)*

The authors are with the OPTIMA and the School of Mathematics and Statistics, The University of Melbourne, Melbourne, VIC 3010, Australia (e-mail: estefaniay@student.unimelb.edu.au).

This article has supplementary downloadable material available at https://doi.org/10.1109/TEVC.2022.3205165, provided by the authors.

Digital Object Identifier 10.1109/TEVC.2022.3205165

behaviors to support the refinement of existing algorithms and design of new ones.

A key element of such studies (e.g., [5], [6], and [7]) is test suites that are: 1) *comprehensive*, ranging in difficulty to challenge an algorithm in different ways; 2) *representative* of their problem class or characteristics, so that algorithm performance can be generalized and inferred from the class; 3) easily *scalable* and *tunable*, with adjustable properties. In other words, the degree, presence, or location can be tweaked, such as for dimension, variable dependency, PF geometry, and location of optima; and 4) have *known optimal solutions*, such that algorithm solution quality can be assessed and compared accurately. These design principles have led to the ZDT [8], DTLZ [7], and WFG [5] general-purpose suites, aimed to challenge MO algorithms across different characteristics. Complementing them are problem suites focused on specific challenges, such as bias [9] and large-scale problems [10].

While these efforts have taken us a long way toward a comprehensive set of benchmarks, it remains unclear whether the whole *problem space* is being sufficiently represented. For example, Zhou *et al.* [11] asserted that many MO benchmark problems have not been rigorously analyzed, are poorly constructed and, above all, lack diversity. However, the diversity of benchmarks is difficult to quantify when problems are usually only described using broad labels such as "nonseparable" or "multimodal," without further quantification of the degree of such characteristics. Therefore, having a quantifiable measure of the similarity between problems, and evaluating their distribution in the problem space, can better inform benchmark evaluation and construction.

Yamamoto *et al.* [12] conducted a visual inspection of the relationship between the ZDT, DTLZ, and WFG suites. The performance of each algorithm in a portfolio was assessed using their median hypervolume (HV) metric, and for each problem instance, the algorithms were ranked based on this metric. The location of each problem was then projected into a 2-D plane using multidimensional scaling (MDS) based on the ranking of algorithms with Spearman rank correlation as the distance metric. Their analysis showed that problems within each benchmark suite are closely related in terms of algorithm ranking, but the space has many sparse areas. However, due to their choice to define problem instances by algorithm rankings only, their analysis could not explain which types of problems are missing, nor how to construct them.

In our recent work [13], we explored the diversity of the ZDT, DTLZ, WFG, IMMOEA, and RMMEDA suites using measurable properties of the suites themselves. Using a combination of design of experiment (DoE) methods, exploratory landscape analysis (ELA) features, and visualization techniques, we confirmed that using these suites with the commonly used default construction parameters leads to a limited diversity of problems in terms of their landscape characteristics. We then strategically tuned the parameters, transforming the landscapes of several problems, which also resulted in algorithm performance variations. However, there was a limit to the diversity that can be achieved with this parameter tuning strategy, with some problems being quite robust and nontuneable, and it was difficult to control the characteristics of the resulting benchmarks.

In this article, we continue this line of investigation by considering the most comprehensive set of benchmark suites to date, and examining their diversity. For this purpose, we use instance space analysis (ISA) [14], a recent methodological framework that teases out the relationships between the problem characteristics and algorithm performance. A 2-D *instance space* is constructed that shows an empirical boundary within which all MO problems could lie—based on the estimated bounds for each calculated feature—and the location of current benchmarks. The problem instances are represented as feature vectors that capture their relevant characteristics, measured through existing [15], [16] and new ELA metrics. The performance of algorithms across the instance space, measured by a modified version of the inverted generational distance (IGD_M^+) [13], [17], can be inspected to understand how performance depends on problem characteristics. Leveraging the visual insights made available by ISA, we explore several novel strategies to construct new benchmarks that fall within targeted sparse areas of the instance space. Our results demonstrate that these strategies afford far greater flexibility than mere tuning of construction parameters of existing problems suites can achieve. They also provide greater control over the characteristics of the generated benchmarks, enabling harder or more varied problems to be constructed through hybridization choices based on insights. Finally, we propose some diverse new benchmark problems based on the insights gained through the ISA methodology.

The remainder of this article is organized as follows. In Section II, we perform the first ISA of MO problems to evaluate the diversity of existing benchmark suites. This includes devising some new features of MO problems to capture their intrinsic difficulties. Based on this instance space view of existing benchmarks, in Section III, we explore a systematic approach to the generation of new diverse benchmarks. Results are discussed in Section IV. We conclude this article and discuss the future research directions in Section V.

II. Instance Space Analysis of Existing Multiobjective Benchmarks

A. Instance Space Analysis

ISA is a methodology first introduced by Smith-Miles *et al.* [14] that generates a 2-D visualization of a collection of problem instances and analyzes their impact on algorithm performance. The instance space allows for trends in hardness to be observed across the space for different algorithms, providing a method for identifying and objectively measuring the region of the superior performance of an algorithm (known as its footprint). Moreover, the mapping into the instance space facilitates insights into the distribution of existing instances, which allows us to identify sparse, unoccupied regions where new benchmark instances should be generated for a more comprehensive problem set. ISA can be automatically performed through publicly available Web tools [18] or using the MATLAB toolkit [19]. The pseudocode for the algorithms used to develop ISA can also be found in the supplementary materials. In general, ISA involves the following steps.

1) Collecting the metadata regarding test instances: their measured features and corresponding algorithm performance metrics.
2) Selecting a subset of features that best capture similarities and differences in the problem instances, and best discriminate between the performance of algorithms in the portfolio.
3) Projecting a visualization of the problem instances from the high-dimensional feature space into a 2-D instance space.
4) Quantitatively measuring the algorithm footprints, and qualitatively describing them in terms of instance features.
5) Generating new test problem instances to occupy sparse regions.

In the following sections, we will describe the elements of the metadata for our ISA of MO.

B. Benchmark Test Suites

We utilize most of the available benchmark problems in the PlatEMO toolbox [20], discarding those with extremely sparse solutions, as this affects the feature calculation procedure described in Section II-C. Therefore, we utilize the bi- and tri-objective continuous problems in the VNT [21], ZDT [8], DTLZ [7], WFG [5], RMMEDA [22], IMMOEA [23], BT [10], IMOP [24], and SMOP [25] test suites, as well as UF1–10 [26]. The default implementation in the PlatEMO toolbox is used for each suite, ranging in dimensionality across problems (2, 7, 10, 12, 22, 30, and 100). The dimensionality of each problem is available in the supplementary materials. The VNT problems have a discrete set of PFs, while the ZDT, DTLZ, and WFG problems are general purpose and contain a mix of PF geometries and landscape functions. The IMOP suite focuses on complicated PF shapes. The UF problems are constructed to exhibit challenging shapes (such as spirals) of Pareto-optimal solutions in the decision space, rather than challenging PFs themselves. The IMMOEA and RMMEDA suites build upon the ZDT and DTLZ suites, with the addition of variable linkages. The SMOP suite contains problems with adjustable sparseness in the PF, and the BT suite contains problems with bias. We refer to this set of all these suites as *Benchmark Instances*, with the ZDT, DTLZ, and WFG problems separately labeled as *General-purpose Benchmark Instances*.

Furthermore, we include the tuned versions of the ZDT, DTLZ, and WFG problems from our previous work [13]. These suites have *construction parameters*, which control the presence (or absence) of characteristics beyond the dimensionality or number of objectives. For example, in WFG4, multimodality can be tuned for the magnitude of hill sizes, as well as the number of modes. The inclusion of such problems allows us to observe the impact of tuning and the extent of diversity introduced into the instance space for each problem. This is important to understand, since tunability is a key characteristic for benchmark design. The parameter tuning strategy for generating instances has two phases. In the first, we used Latin hypercube sampling (LHS) on the parameter space to generate 30 combinations. This is to ensure diversity in the construction parameters through a space-filling design. Eight state-of-the-art algorithms—including the six described in Section II-D, and two less competitive algorithms (RVEA and NSGA-III) [13]—were evaluated on all problem instance variations, and their average performance on the IGD_M^+ metric, which is described in detail in Section II-D, was estimated. This generated a set of 600 problem instances based on varying construction parameters of the General-purpose Benchmark suites, that we will refer to as *Perturbed Instances*.

Thereafter, we used the design and analysis of experiments (DACEs) [27] surrogate modeling technique to strategically generate harder or more diverse problem instances for each algorithm based on the DTLZ and WFG problems, but excluding the ZDT problems, as most of them had only one construction parameter, which was perturbed for $\{0.5\} \cup \{1, 2, \ldots, 30\}$. The inputs to the DACE model are the construction parameters, while the average algorithm performance is the output. Therefore, the DACE model predicts the IGD_M^+ for any construction parameter combination for a problem instance.

The DACE model then efficiently searches for construction parameter combinations to evaluate for each algorithm by balancing two objectives: 1) exploration, i.e., reducing the model uncertainty by searching for combinations that maximize the expected improvement (EI) in model error and 2) exploitation, i.e., minimizing the performance measure IGD_M^+ to create harder problem instances for each algorithm. The selection of new construction parameter combinations is balanced using a single metric known as EI [28]. Algorithms are then evaluated on these new instances, added to the model, and the process is repeated five times per algorithm. We refer to this set as *Harder* instances.

We use all default parameter settings within PlatEMO to initialize the Benchmark Instances, and tune the construction parameters accordingly for the Perturbed and Harder instance. In total, there are 1314 instances considered across the three instance classes of Benchmark, Perturbed, and Harder Instances.

C. Features

ELA is the term used for extracting from a sample a set of quantitative features that describe the topology of problem instances [29]. While research into single-objective features has grown steadily [2], [30], [31], [32], the MO extensions have not been as prolific. While single-objective features could be used to explore each objective separately and then combine into MO metrics using ratios or linear scalarizations [15], this approach cannot account for the tradeoffs between conflicting objectives. In our preliminary experimental studies supporting this article, we found such ratios to be insufficient. Instead, we believe it is important to adapt single-objective features for the MO case, using procedures such as dominance ranking to truly capture the interplay of objectives.

Previous work on understanding such tradeoffs in MO landscapes is limited to combinatorial problems [34], [37], [38],

TABLE I
LIST OF THE 35 MO FEATURES USED IN THIS ARTICLE AND THE CHARACTERISTICS THEY CAPTURE. THE NUMBER OF FEATURES IN EACH CLASS IS DENOTED BY PARENTHESIS IN THE FIRST COLUMN

Class	Description	Characteristic	Ref.
State-of-the-art features from the literature			
sup_rws (2)	proportion of neighbours dominating sample solution (mean, first autocorrelation)	objective correlation, ruggedness	[16]
inf_rws (2)	proportion of neighbours dominated by sample solution (mean, first autocorrelation)	objective correlation, ruggedness	[16]
lnd_rws (2)	proportion of neighbours non-dominated to sample solution (mean, first autocorrelation)	objective correlation, ruggedness	[16]
inc_rws (2)	proportion of neighbours incomparable to sample solution (mean, first autocorrelation)	objective correlation, ruggedness	[16]
aep (1)	accumulated escape probability	evolvability	[33]
mdl_r2 (1)	R^2 of linear model fitted on the non-dominated front rank versus the decision values	variable scaling	[30]
kurt (4)	kurtosis of the non-dominated front rank (min, mean, max, range)	multi-modality	[30]
rankprop (1)	proportion of best solutions		[34]
obclust_n (1)	optimal number of clusters in the objective space	bias, multi-modality	[30]
obclust (3)	proportion of solutions in clusters (min, max, range)	bias	[30]
Hy (1)	entropy of objective values		[35]
Hxy (1)	joint entropy of decision and objective values	separability	[35]
signif (1)	significance measure	epistasis	[36]
Proposed new features			
dec_range (1)	difference in range of parameter domain (max)	variable scaling	
obdisc (3)	distance between best ranked solutions in objective space (mean, IQR, max)	discontinuous PF	
nbr_dist (3)	total distance walked within each neighbourhood, along the sample random walk (min, mean, first autocorrelation)	neutrality	
nbr_rank_best (3)	number of best ranked solutions (from total sample) in each neighbourhood (mean, min, max)	ruggedness, deceptiveness	
obdist (1)	distance between the first and second best ranked fronts in the objective space (mean)	bias	
decdist (1)	average distance between the first and second best ranked fronts in the decision space	evolvability	
range_coeff (1)	the range of the coefficients in the linear model	variable scaling	

[39], [40], [41], with few exceptions [42], [43]. These studies often propose features for understanding the landscape through global enumeration or a large sample, which can be infeasible in practice, especially for high-dimensional problems and expensive function evaluations. Therefore, MO features which can be obtained through sampling are necessary. In this section, we contribute to this challenge first by adapting both single-objective features and combinatorial MO problem features to the continuous MO domain; and second by proposing a set of new features. Table I summarizes the features employed in this article, with details discussed in the following sections.

1) Adapting Existing Features: Several combinatorial MO features were proposed by Liefooghe *et al.* [16], which were measured by sampling locally during random and adaptive walks. We modify the concept for the continuous MO domain first by generating a sample in the decision space of size 10^3 using LHS, from which a sequence is constructed using nearest neighbor search, with Euclidean distance as metric. This sequence now represents the random walk.

Definition 1: A walk is defined by sequentially connecting generated samples using a nearest neighbor search.

Then, a larger sample of 3×10^5 is taken from the decision space, which is grouped into 10^4 clusters using K-means. These clusters divide the decision space into neighborhoods. Next, for each point in the sequence, we identify its neighborhood and take up to ten neighboring points from the larger sample.

Definition 2: A neighborhood is defined by mapping new points into preexisting clusters. Points in the same cluster are considered neighbors.

For sparse problems which may not have ten neighbors, we obtain the maximum number of neighbors. The upper and lower bounds of the walks and neighborhoods are scaled relative to the domain of each problem's respective x_i in the decision space. Only the sequence and its neighboring points are evaluated on the benchmark function, so as to minimize computational costs. We then calculate the proportion of neighbors dominating, dominated by, nondominated by, and incomparable to each sample of the random walk. The averages and first autocorrelations of these measures are calculated along the walk for each instance. For more robust evaluations for each of these features, repeated sampling can be used. However, for the purposes of this study, a larger sample size has been chosen to ensure consistency of feature measures, following minimum sample recommendations [44]. In our experimental studies, our findings showed that when using a similar budget, a longer walk provided a smaller error rate in comparison to repeated short walks.

In the simplest case, we calculate the kurtosis for each objective individually and then calculate univariate statistics across these kurtosis values. We also adapt features from single-objective optimization, with an approach taken to honor the multiobjective nature of the problem. For features which can only be calculated on a single objective, we use the concept of nondominated ranking [45]. In other words, each solution belongs to a front in the Pareto space, and each front has a ranking. This ranking is used instead of the respective single-objective value.

We measure the accumulated escape probability [33] by substituting the nondominated ranking as fitness. Similarly, we fit a linear model on the ranks versus the decision variables and capture the model R^2 and the range of its coefficients. We also obtain the proportion of nondominated solutions, the optimal number of K-means clusters in the objective space using the Calinski–Harabasz criterion and the minimum, maximum, and range of the proportion of solutions within the clusters. Since nondominated ranking is a measure of relative performance and not an absolute measure in the objective space, it is not applicable for adapting many single-objective techniques. However, ranking is sufficient for the purposes of the features above, since they are able to provide information from a relative hierarchy. Finally, we use Stowell and Plumbley's estimator [35] to calculate the entropy in the objective space, as well as the joint entropy of the decision and objective space, and the entropic significance [36]. The usefulness of these additional features is demonstrated by an experimental study (provided in the supplementary materials) that shows their effectiveness at predicting algorithm performance compared to existing features in the literature.

2) New Set of Continuous MO Landscape Features: While the modification of features by Liefooghe *et al.* [16] is a promising development, they have also been shown to be insufficient for truly discriminating between algorithm performance for continuous MO [46]. As such, in this article, we propose a novel set of features, specifically to address a broader range of relevant characteristics.

Dec_range is a feature to measure dissimilar parameter domains. Since each decision variable x_i is bounded by a lower bound l_i, and upper bound u_i, the dissimilar domains can be calculated as the magnitude of the difference between the maximum upper and minimum lower bounds. This should evaluate the impact of variable scaling on algorithm performance [5]

$$dec\_max\_range = \max_i(u_i) - \min_i(l_i) \tag{1}$$

Obdisc is a set of measures that attempt to identify discontinuous fronts. Large gaps along the best-ranked (first nondominated) front suggest potential discontinuities. This metric utilizes the pairwise distance

$$d_{ij}^2 = \left(\mathbf{f}_i^1 - \mathbf{f}_j^1\right)^\top \left(\mathbf{f}_i^1 - \mathbf{f}_j^1\right) \tag{2}$$

where $\mathbf{f}_i^1$ represents the ith point in the rank 1 (nondominated) front. The maximum, IQR, and mean are then calculated from these pairwise distances for obdisc_max, obdisc_iqr, and obdisc_mean, respectively.

Nbr_dist is a metric which calculates the total distance walked across the objective space by each neighborhood using the summation of the pairwise distance metric as in obdisc. This is then used to measure the differences in the distance in the objective space traversed by each neighborhood. The minimum, average, and first autocorrelation are collected for the walk, as defined in Definition 1. A high autocorrelation demonstrates that the distance across the objective space does not change much between neighborhoods (and therefore is more likely to be neutral), while a low one means there is a large change between neighborhoods. Therefore, when the minimum (min_nbr_dist) and average distance (nbr_dist_avg) are small and the autocorrelation (nbr_dist_r1_rws) is large, we can infer that the landscape is neutral.

Nbr_rank_best measures the average rank for each neighborhood that contains a nondominated (for the entire sample) solution. The nondominated front ranking is used to determine the "goodness" of each solution. Since only neighborhoods with at least one nondominated solution are used, nbr_rank_best measures along the neighborhoods surrounding only the best solutions. The average, minimum, and maximum across all neighborhoods are then recorded. These are used as measures to identify whether there is ruggedness. When the average is high, the nondominated solutions are surrounded by inferior solutions. Simultaneously, it may be a measure of deceptiveness since it is measured using a sample. When problems are highly deceptive and the nondominated solution is away from this solution, a low value of avg_nbr_rank_best will be obtained, since their approximate best neighborhoods will seem similar. Conversely, if the nondominated solution is the global optimal found and the remainder of the neighborhood is filled with poor solutions with high rank, avg_nbr_rank_best will be high.

Obdist measures the average paired distance between the nondominated front and the second-ranked front in the objective space.

$$Obdist = \frac{\sum \left(\mathbf{f}_i^1 - \mathbf{f}_j^2\right)^\top \left(\mathbf{f}_i^1 - \mathbf{f}_j^2\right)}{|\mathbf{f}^1| + |\mathbf{f}^2|}. \tag{3}$$

This captures the improvements between the two best fronts. When this distance is large, there may be bias away from the PF.

Decdist measures the distance in the decision space for solutions across and between the first- and second-best ranked front. This is the equivalent of obdist in the decision space. The purpose of this feature is to capture whether large mutations in algorithm populations are required to identify superior solutions. The average distance is recorded.

D. Algorithms and Performance Metric

The selected benchmark problems were evaluated using a portfolio composed of six state-of-the-art algorithms, across four different classes, ensuring the representation of diverse elitism and niching strategies [47].

1) *Pareto-Dominance:* NSGA-II [48] and SPEA2 [49].
2) *Decomposition:* MOEA/D (Tchebycheff decomposition) [50].
3) *Grid-Based:* GrEA [51].
4) *Indicator-Based:* HypE [52] and IBEA [53].

All algorithms were run 30 times on each benchmark, with each run having a budget of 10^4 evaluations. Default settings of each algorithm were used based on the PLATEMO implementation [20]. To ensure comparability across problems, we evaluate algorithm performance using the average IGD_M^+ (a modified IGD metric) across runs [46]. This metric is used instead of HV, since HV favors knee and boundary points over the distribution of solutions on the PF [10]

$$\text{IGD}(S, P) = \frac{\left(\sum_{i=1}^{|S|} d_i^2\right)^{1/2}}{|S|} \tag{4}$$

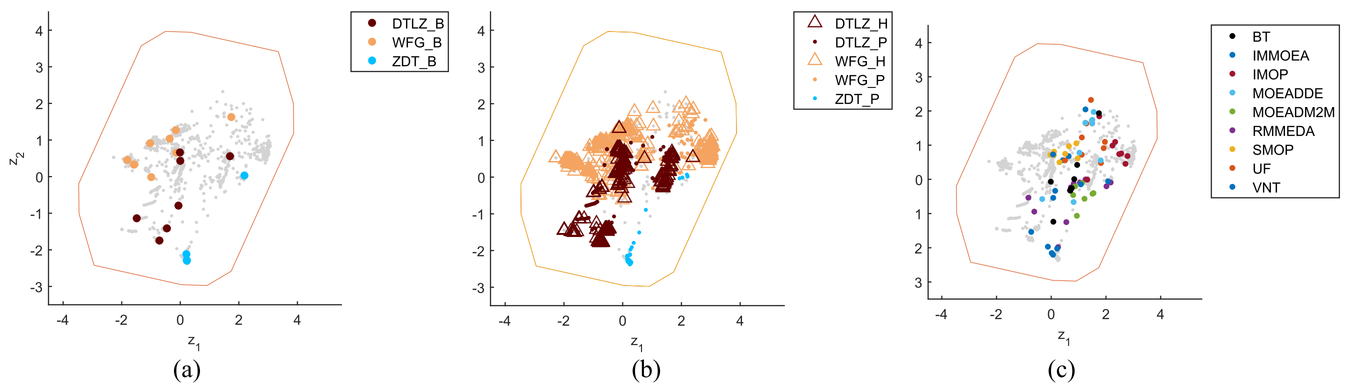

Fig. 1. Location of specific benchmark instance problem classes highlighted against all instances in gray. H and P denote the Harder and Perturbed Instances, respectively. The empirical boundary of the space is shown in orange. (a) General-purpose Benchmark suites. (b) Perturbed and Harder Instances. (c) Other suites.

where P is the true PF reference set, S is the nondominated objective vector obtained, and d_i is the Euclidean distance from the elements of $p \in P$ with its nearest neighbor in set $s \in S$. IGD$^+$ modifies the distance calculation as $\sqrt{\sum_{i=1}^{|Z|} \max(\vec{p}_i - \vec{s}_i, 0)^2}$ to account for dominance relations [54]. However, to allow for comparisons across problems, the metric must be normalized. We, therefore, use IGD^+_M, which is defined as follows:

$$\mathrm{IGD}^+_M = \begin{cases} 1 - \mathrm{IGD}^+_N & 0 \le \mathrm{IGD}^+_N \le 1 \\ \frac{1}{\mathrm{IGD}^+_N} - 1 & 1 < \mathrm{IGD}^+_N \end{cases} \tag{5}$$

where IGD^+_N is IGD$^+$ normalized by dividing by the mean distance between nadir point r, with the PF as the reference set. IGD$^+_M = 1$ is bounded between $[-1, 1]$, where problems become harder as IGD$^+_M \to -1$. For the ISA that follows, we consider an algorithm's performance to be "good" on a problem instance if its IGD^+_M metric is within 1% of the best IGD^+_M obtained by the portfolio.

E. Preliminary Instance Space

We generate an instance space by adopting the methodology [14] for the continuous MO metadata described above. Each instance of the benchmark suites described in Section II-B is initially summarized as a 35-D feature vector. The top 10 most discriminating features for distinguishing between good and bad algorithm performance are identified during a feature selection process within the ISA methodology. In this process, a subset of features which best summarizes the key features of the instances that affect algorithm performance is identified. This is achieved by calculating Pearson correlations between features and algorithm performance. Features which share the highest correlations with algorithm performance are retained, and k-means clustering is used to group similar features together. To identify the best combination of features, one feature from within each cluster is taken, forming a feature subset, and this is repeated for all possible combinations. The optimal feature vector is selected which demonstrates the lowest predictive error when classifying whether an algorithm is good, based on a user-defined metric, upon projection onto 2-D using principal component analysis (PCA). With these features selected, an optimization problem is then solved to determine the best linear transformation for projecting the instances from 10-D to 2-D, such that the feature distributions and algorithm performances are as linear across the instance space as possible to aid the interpretation of trends

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} 0.0644 & -0.4298 \\ -0.4963 & -0.3249 \\ -0.0039 & 0.2565 \\ 0.1997 & -0.1846 \\ -0.0086 & 0.2309 \\ -0.1259 & -0.3304 \\ 0.2513 & 0.1190 \\ 0.1950 & -0.0052 \\ -0.3862 & -0.5401 \\ 0.1953 & -0.0385 \end{bmatrix}^{\top} \begin{bmatrix} \text{inf\_avg\_rws} \\ \text{inc\_avg\_rws} \\ \text{decdist} \\ \text{obclust\_n} \\ \text{obclust\_max} \\ \text{obdisc\_max} \\ \text{kurt\_max} \\ \text{signif} \\ \text{min\_nbrdist} \\ \text{avg\_nbr\_rank\_best} \end{bmatrix}. \tag{6}$$

We can now use the 2-D visualizations to scrutinize the existing test suites and identify gaps in the current benchmarks. However, it is important to note that these are not necessarily robust observations as they are reliant on the metadata and the instance space generated. Therefore, its sensitivity to the metadata would need to be comprehensively explored in order to ensure these observations persist despite small changes in the metadata. Such sensitivity analysis is beyond the scope of this article, which is more focused on how to generate additional instances to a given instance space so that observations can be supported by the most comprehensive set of instances.

Fig. 1 shows the location of the existing benchmarks within the instance space given by (6). The empirical boundary of the instance space is shown by the solid orange lines and informs on the space that could be filled by new instances. The General-purpose Benchmark suites in Fig. 1(a) do not demonstrate much diversity or coverage of the entire space. Problems within suites are also shown to be close to each other, verifying the results of Yamamoto *et al.* [12]. In Fig. 1(b), the instance space reveals that when considering additional instances generated by tuning problem construction parameters, the Perturbed and Harder Instances improve the coverage. As expected,

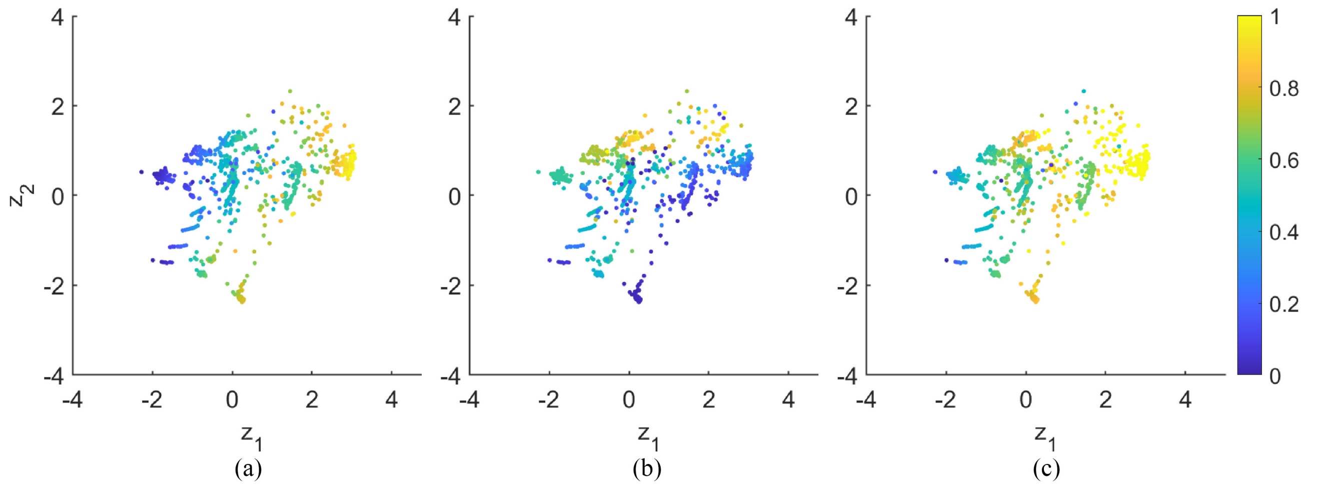

Fig. 2. Feature distribution for three features: (a) inf_avg_rws; (b) obclust_max; and (c) avg_rank_nbr_best.

Fig. 1(b) shows that the ZDT problems do not offer much variation in characteristics of problem instances, while the DTLZ and WFG show greater diversity within the instance space when parameters are varied.

Fig. 1(c) shows that problems in other suites tend to cluster with each other. Intuitively, the RMMEDA and IMMOEA suites, which are built from ZDT and DTLZ problems with the addition of variable linkages, occupy the lower region, much like the ZDT and DTLZ suites. However, they do occupy areas between the suites, suggesting that there are still limitations to filling the instance space only by the addition of variable linkages. Comparing the existing suites to the empirical boundary, there are clear empty regions in the upper, left, and bottom right regions. Therefore, the diversity of existing suites still lacks in a sufficient coverage for a comprehensive problem set.

It is important to note that while the instance space was generated by combining biobjective and triobjective problems, these can be studied separately if the focus is on a more granular separation of characteristics (for each number of objectives). For the purposes of our study, we retain both in the same instance space since there are features which should correlate with the number of objectives, such as objective correlation.

F. Analysis of the Preliminary Instance Space

Inspecting the distribution of three selected features in Fig. 2 provides some insights into the kinds of characteristics present in each suite. The feature inf_avg_rws represents the degree of objective correlation; obclust_max represents the degree of biasedness to one region in the objective space; and avg_nbr_rank_best is used to measure ruggedness and deceptiveness. Fig. 2 colors the instances blue if they have a minimal value of a feature, and yellow for a maximal value. Combining Figs. 1 and 2 allows us to infer some of the characteristics present in the various suites, but also the characteristics of any instance based on its 2-D location.

The majority of the BT suite is located just right of the origin point, with one problem lying in the upper right. This indicates that there is one problem where there is bias toward one region. The IMOP and most of the MOEADDE problems are shown to have low objective correlation and a high degree of ruggedness. The MOEADM2M suite has medium to high levels of ruggedness, a sparse distribution and low objective correlation. SMOP has low to medium sparsity of solutions, as well as medium objective correlation and ruggedness. The UF and VNT suites have a combination of low to medium objective correlation, a range of sparseness, and medium to high ruggedness.

TABLE II
AREA AND DENSITY OF EACH ALGORITHM'S FOOTPRINT DEFINING GOOD PERFORMANCE ON THE EXISTING INSTANCES

Algorithm	Footprint Area	Footprint Density
GrEA	0.549	1.246
HypE	0.343	1.695
IBEA	0.639	1.364
MOEA/D	0.310	1.027
NSGA-II	0.291	0.842
SPEA2	0.375	0.924

The performance of algorithms across the instance space is shown in Table II. *Footprint Area* measures the coverage of good performance of an algorithm across the instance space, while *Footprint Density* measures the number of instances within each footprint of good performance. Since these quantities are normalized by the area and density of the convex hull containing all instances, the density can be greater than one if a footprint is denser than the convex hull. IBEA shows the largest coverage of good performance by far, followed by GrEA and HypE, while NSGA-II has the least coverage. Fig. 3(c) shows that IBEA has the largest footprint for good performance, while Fig. 3(e) NSGA-II has the smallest. Each blue dot represents an instance where good performance is observed, and the areas in light blue are where we can statistically generalize this using the footprint area calculation. Notably, IBEA shows weak performance specifically on the instances toward the lower region, and toward the right.

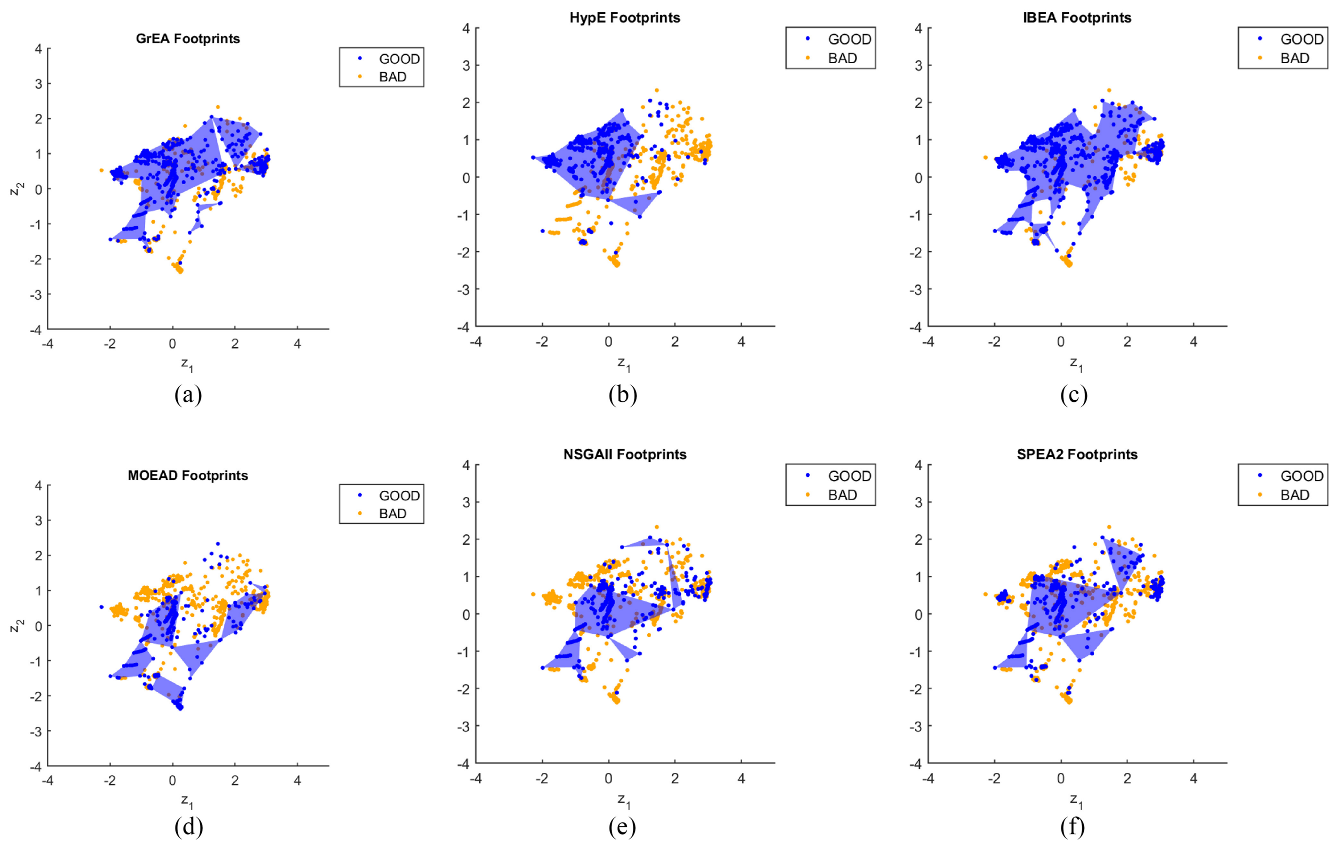

Fig. 3. Algorithm footprints obtained when using the existing benchmark problems and the performance obtained experimentally for: (a) GrEA; (b) HypE; (c) IBEA; (d) MOEA/D; (e) NSGA-II; and (f) SPEA2. An algorithm is good if it is within 1% of the best performing algorithm.

Conversely, MOEA/D shows good performance where IBEA is weak and also performs well toward the lower left regions.

The construction of functions within the Benchmark Instances specifically aims to address specific characteristics, and features provide a quantitative measure of the degree of the characteristic present. For example, SMOP aims to impose challenges for algorithms in obtaining sparse solutions and this is confirmed to be true by low to medium values of obclust_max. Therefore, the instance space confirms that the construction of each successive suite has been successful in creating additional diversity within the interior of the instance space. However, sparseness still remains between suites and particularly in areas surrounding the boundaries.

The results show that when discussing algorithms in a larger context beyond only problems with specific characteristics, the use of a single suite is insufficient for drawing broader conclusions, with each suite demonstrating a limited range across the instance space. In other words, no standalone suite is sufficient for drawing conclusions on an entire problem class or type, or the hierarchy of algorithm performance. As such, generalizations that are made about an algorithm's performance on problem classes may not hold true—it may still have weaknesses within unidentified areas, and further opportunity for developments are possible. By recognizing the diversity present in problem instances and suites, we can generate better informed inferences about where algorithms are likely to perform well, and use these insights to guide development on a broader set of instances. For example, if an algorithm performs well on rugged landscapes which have low to medium bias, but poorly when large bias is present, we can investigate which mechanisms drive behaviors guiding these strengths and weaknesses. Given the included suites in this study, there are unoccupied regions where no suites contribute problem instances in the top left, middle left, and bottom right. As such, we now explore how the instance space can inform the generation of new problems in these regions. Noticeably, the ZDT problems tend to occupy the boundary of the existing suites at the bottom right. We investigate this in Fig. 4, which demonstrates the instance space of existing benchmark instances labeled by the number of objectives. Visually, we observe that the biobjective problems have a tendency to occupy the lower region.

The feature distributions show clear gradients across the instance space and therefore inform us which characteristics a problem must have, such that it belongs to one of these empty regions. For example, inf_avg_rws is negatively correlated with objective correlation, with smaller values suggesting that the problem has a high objective correlation. The gaps in the left demonstrate that there is a lack of high correlation problems, while the right suggests that there is a lack of low objective correlation problems. This is important since practical MO problems typically have conflicting objectives (e.g., [55]). Dominance relations—and therefore elitism strategies—are affected by the degree of objective conflict.

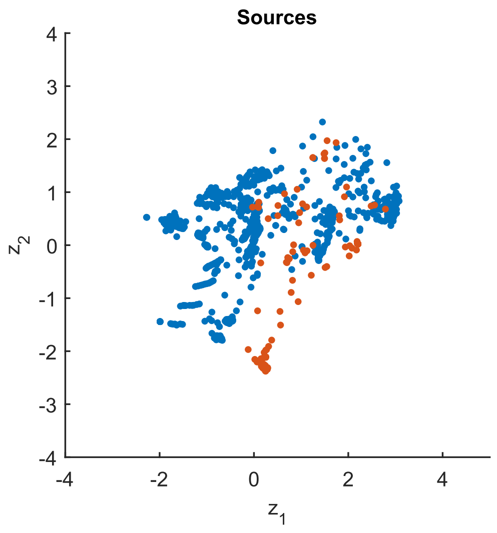

Fig. 4. Existing problems by the number of objectives.

When objective correlation is low, there is a smaller chance of dominated or dominating neighbors [16].

Obclust_max calculates the largest proportion of solutions within a cluster. As such, it measures whether there is bias toward one area within the objective space. This is confirmed by the presence of WFG1 and WFG9 in the upper regions of the instance space. The gaps at the top can be filled with biased problems that map to a minimal number of clusters, while the lower right shows gaps in problems where finding an even distribution is easy. Avg_rank_nbr_best extracts the average rank in the neighborhood of each best solution. A smaller value suggests a consistent neighborhood, resulting in the neighboring solutions being similar in the objective space. This serves as a proxy measure for deceptiveness, with large values of avg_rank_nbr_best showing that neighboring solutions around the best solutions are poor. The gaps on the left can be filled with problems that are less rugged and more deceptive. This is confirmed by the existing problems on the left being DTLZ6 and WFG5, both with a deceptive landscape function/transition. Since the selected benchmark suites lack these types of problems, our knowledge about algorithm performance is deficient when facing problem instances with those specific feature combinations.

In Section III, we utilize the insights provided by the feature distributions to generate new problems with the kind of landscape features that will enable them to occupy sparse and empty regions of the instance space. Generating instances in these regions is important for moving toward a comprehensive benchmark set. It is also critical for understanding the true strengths and weaknesses of algorithms. We already have some studies that have identified challenges for some algorithms. For example, Ishibuchi *et al.* [56] identified that MOEA/D suffers from degrading performance on highly correlated objectives, while NSGA-II and SPEA2 do not. Expanding the instance space to include new instances will enable us to enrich our understanding of algorithm strengths and weaknesses, and make recommendations for appropriate algorithm selection on real-world problems.

III. Constructing New Benchmarks

The existing suites that we have explored show reasonable coverage over a wide range of the instance space. However, they are not yet representative of a comprehensive set. Of course, the generation of a completely comprehensive set may be infeasible, due to the large number of instances which need to be generated to ensure a representative set. Nevertheless, we should explore methods of expanding the existing instances through the inclusion of distinctly different problems.

The results of the ISA in the previous section demonstrate that due to the construction methods of the existing MO suites, their problems tend to cluster, regardless of the problem properties available. This raises a question regarding the limitations imposed by problem generation methods for specific suites. Further research is required into whether an alternative method for generating test suites is possible. Within the instance space of existing suites, we have observed that there are three obvious areas where test problems are lacking: the top left, middle left, and bottom right. In the previous section, we identified the types of problem characteristics represented by features in these empty pockets of the instance space, by analyzing the trends in the feature distributions. We now propose some methods for generating new instances to target such unexplored regions.

A. Methods of Problem Construction

Our motivation for generating new problems is to increase the diversity of the instances and generate a more comprehensive problem set that can expand the instance space. We consider three methods to achieve this.

1) Method 1—Tuning: The first and simplest method is to tune instances, expanding our previous work [13] to additional suites. The tuning of instances has been applied to the landscape and PF shape functions of the IMMOEA and RMMEDA suites. LHS is used to generate combinations of values which are used for tuning. Fig. 1(b) showed that tuning the General-purpose suites results in similar problems being generated, and while we expect this to hold true with the IMMOEA and RMMEDA suites, we test the method for these suites nonetheless.

2) Method 2—Hybridizing: For our next two construction methods, we build onto the LSMOP test toolkit [9] due to its ease of use and tuneability. The toolkit generates objective functions defined as

$$\begin{cases} f_1(x) = h_1\left(\mathbf{x}^f\right)\left(1 + \sum_{j=1}^{M} c_{1,j}\bar{g}_1\left(\mathbf{x}_j^s\right)\right) \\ \dots \\ f_i(x) = h_i\left(\mathbf{x}^f\right)\left(1 + \sum_{j=1}^{M} c_{i,j}\bar{g}_i\left(\mathbf{x}_j^s\right)\right) \\ \dots \\ f_M(x) = h_M\left(\mathbf{x}^f\right)\left(1 + \sum_{j=1}^{M} c_{M,j}\bar{g}_M\left(\mathbf{x}_j^s\right)\right) \end{cases} \tag{7}$$

where $\mathbf{x}^f = (x_1, \dots, x_{m-1})$ is the first part of the decision vector and $\mathbf{x}^s = (x_m, \dots, x_n)$ is the second remaining part. f_i are the objective functions, where M is the number of objectives. g_i are the landscape functions, h_i defines the shape of the PF, and C is the correlation matrix which controls the correlation between the objective $f_i(\mathbf{x})$ and x_j^s. While the original toolkit is proposed for large-scale problems, hence its omission from the Benchmark Instances presented earlier, we now use it to construct general problems by exploring the diversity that can be introduced through its different configurations.

We hybridize suites by combining the landscape functions from DTLZ1–DTLZ7, with the shape function and decision vectors from LSMOP1. Therefore, the LSMOP construction is preserved and the characteristics of each DTLZ problem are already known. In the case of hybridizing, we expect that it is likely to create functions that exist between the initial problems. However, since the instance space does not show many sparse regions in its interior, there may be limitations to the relevance of this technique.

3) Method 3—Injecting New Functions: Alternatively, new functions have the potential to create problems that occupy empty regions with guidance from the feature distributions. The hybridization technique uses existing functions with known properties. We can leverage this information when injecting new functions which have more (or less) extreme values of certain characteristics. For example, the triobjective DTLZ6 landscape function for deceptiveness is defined as

$$g(\mathbf{x}_3) = \sum_{x_i \in \mathbf{x}_3} x_i^{0.1} \tag{8}$$

where x_i are the decision values associated with the third objective. By construction, the Pareto-optimal solutions correspond to $g(\mathbf{x}_3) = 0$. As such, the exponent 0.1 biases the decision variables away from the PF. This introduces deceptiveness, with the majority of the objective space leading toward solutions away from the PF. Simultaneously, as this exponent increases, neutrality is introduced in the search space, since objective values which are not optimal will all approach zero. We can take inspiration from DTLZ6's landscape by selecting functions which share similar traits: a landscape with neutrality and deception present. For example, the following function creates a mostly neutral landscape with a deceptive optimum and multiple local optima:

$$g(x_M) = 10\left|\sum_{x_i \in \mathbf{x}_M}\left(\frac{1}{x_i}\right) - 1\right|^{0.1}. \tag{9}$$

Pareto-optimal solutions are obtained when $x_i = 1\ \forall i$. The coefficient of 10 ensures that algorithms will achieve poor results when away from the PF. The absolute value ensures positive objective values and a ridge, and in combination with the exponent 0.1, defines the extent of deceptiveness and multimodality present. Therefore, if we were exploring the tuning of this landscape function, there are two tuneable construction parameters.

B. Problem Construction Methodology

Based on the ISA, we identify three empty areas that are important to address: 1) the upper region; 2) left; and 3) lower right. Fig. 4 has already illustrated that the bottom right area is populated by biobjective problems. Hence, our first step is to generate such problems to attempt to fill the gap in this region. To address this, we generate bi- and tri-objective versions of the LSMOP problems with a dimension of 30.

We then include tuned versions of the IMMOEA and RMMEDA suites from our previous work [13] as examples of method 1 to compare problem generation methods. Furthermore, we demonstrate hybridization (method 2) by mixing LSMOP1 with each of the DTLZ1–6 landscape functions.

For the injection of new functions based on insights afforded by ISA (method 3) we target the empty upper region and left region. Fig. 2(b) shows that problems located in the upper region have a larger value of obclust_max. Since this feature represents the maximum proportion of solutions within one cluster, we infer that these are unimodal problems with strong bias toward one region in the objective space. On the other hand, Fig. 2(a) and (c) suggests that the left region is likely to be populated by problems with high objective correlation, low ruggedness and are possibly deceptive. Therefore, we aim to inject new functions into the LSMOP suite which exhibit such characteristics to demonstrate the effectiveness of the third method of new problem construction.

Using method 3 to address the upper region, we inject functions defined in Table III. Since they are biased and unimodal, we expect these functions to exist in this upper area. For instance on the left, we must increase the objective correlation by impacting the local Pareto-optimal solutions. As such, we use these same functions with reduced upper bounds $u_i = 1\ \forall i$. This is expected to push instances with these properties toward the left. The full composition of the new test instances is listed in Table IV.

TABLE III
FOUR NEW LANDSCAPE FUNCTIONS INJECTED INTO THE LSMOP SUITE AND THE PROBLEM COMPOSITIONS USED (FROM LSMOP1 AND LSMOP6). p_1 DENOTES THE CONSTRUCTION PARAMETER THAT IS ALTERED FOR NEWLY GENERATED INSTANCES

No.	Function
1	$g_{inj}^1 = p_1(10 - 10exp(-0.2\sqrt{\frac{1}{2}\sum((\mathbf{x}-1)^2)}))^{0.1}$
2	$g_{inj}^2 = p_1(10 - 10exp(-\sqrt{\frac{1}{2}\sum(\mathbf{x}-1)^2}) - \sum(sin(\mathbf{x}-1)^2))^{0.01}$
3	$g_{inj}^3 = 2p_1\sqrt{\lvert\sum(\mathbf{x}-\mathbf{x}^2)\rvert} + 0.01\lvert\sum(\mathbf{x}^2)\rvert^{0.01}$
4	$g_{inj}^4 = p_1\sum cos(\frac{\pi}{2}(\mathbf{x}-1))^{12}(\sum sin(\mathbf{x}-1)^2)^{0.01}$
Problem compositions	
1	$\begin{aligned} f_1 &= (1+g_1)0.5\prod_{i=1}^{M-1} x_1 \\ f_{m=2:M-1} &= (1+g_{m=2:M-1})0.5\left(\prod_{i=1}^{M-m} x_i\right)(1-x_{M-m+1}) \\ f_M &= (1+g_M)0.5(1-x_1) \end{aligned}$
2	$\begin{aligned} f_1 &= (1+g_1)\prod_{i=1}^{M-1} cos(x_i\pi/2) \\ f_{m=2:M-1} &= (1+g_{m=2:M-1})\left(\prod_{i=1}^{M-m} cos(x_i\pi/2)\right)sin(x_{M-m+1}\pi/2) \\ f_M &= (1+g_M)sin(x_1\pi/2) \end{aligned}$

IV. RESULTS

Fig. 5 demonstrates that the region occupied by the LSMOP suite is different from the General-purpose suites. The biobjective versions occupy locations toward the bottom right corner, confirming that the biobjective problems are needed to fill the gap in that region. This result is intuitive given that the literature has progressively moved away from biobjective

TABLE IV
COMBINATION OF PROBLEM COMPOSITION, INJECTED LANDSCAPE FUNCTIONS, BOUNDS, AND CONSTRUCTION PARAMETERS USED TO GENERATE NEW INSTANCES. O REPRESENTS THE ORIGINAL BOUND ON THE DECISION VARIABLES, WHILE U REPRESENTS THE UPDATED UPPER BOUND

Type	Functions	Bound	Parameter
1	$Composition: 1$ $g_1 = g_3 = Schwefel\ 2.2.1$ $g_2 = g^1_{inj}$	O	$p_1 = 2, 5, 10$
2	$Composition: 1$ $g_1 = g_3 = g^2_{inj}$ $g_2 = Schwefel\ 2.2.1$	O	$p_1 = 10$
3	$Composition: 1$ $g_1 = g_3 = g^3_{inj}$ $g_2 = Schwefel\ 2.2.1$	O	$p_1 = 10$
4	$Composition: 1$ $g_1 = g_3 = g^4_{inj}$ $g_2 = Schwefel\ 2.2.1$	O	$p_1 = 10$
5	$Composition: 2$ $g_1 = g_2 = g_3 = g^2_{inj}$	U	$p_1 = 2, 5, 10$
6	$Composition: 2$ $g_1 = g_3 = g^2_{inj}$ $g_2 = Schwefel\ 2.2.1$	O, U	$p_1 = 2, 5, 10$
7	$Composition: 2$ $g_1 = g_3 = g^2_{inj}$ $g_2 = Rastrigin$	U	$p_1 = 10$
8	$Composition: 2$ $g_1 = g_3 = g^3_{inj}$ $g_2 = g^2_{inj}$	O	$p_1 = 2, 5, 10$

problems following the work of Deb [4]. Since biobjective problems are simpler to construct, we focus our efforts on more challenging parts of the instance space. Conversely, the triobjective LSMOP1–9 are located toward the upper region of the instance space. We observe that the LSMOP suite originally demonstrates the clustering limitation for problems within a suite. However, once we reduce the upper bound for the decision variables and therefore reduce the sparsity in the decision space, we are able to generate problems in the previously unoccupied bottom left of the instance space.

A. New Problems

We now examine the effectiveness of the three different methods we have proposed for new problem generation.

1) Tuning: Problems from the IMMOEA and RMMEDA suites are tuned to demonstrate the diversity achievable by tuning methods. In Fig. 5, the tuned functions demonstrate similar behavior to that of the General-purpose Benchmark Instances, whereby instances tend to cluster together. However, since both IMMOEA and RMMEDA are located in between the General-purpose suites, more diversity is introduced to cover the interior of the instance space. As such, this method is useful for filling empty areas surrounding existing instances.

2) Hybridizing: The hybridized problems consist of DTLZ1–6 functions combined with the sphere function within

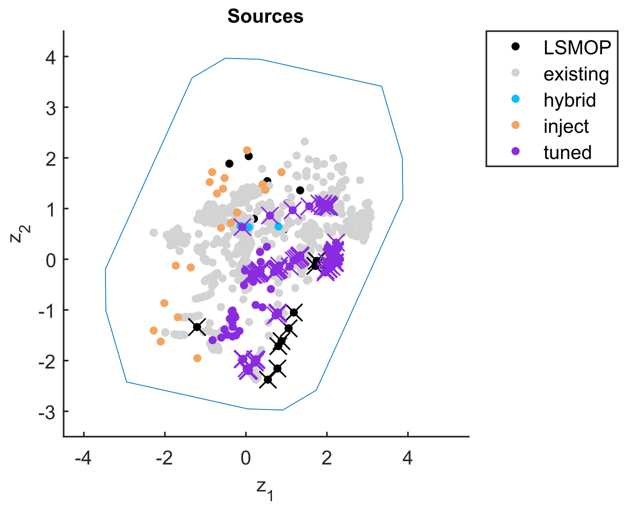

Fig. 5. Newly generated problems shown by different methods. New problems include the LSMOP suite, hybrid LSMOP with DTLZ functions, tuned IMMOEA and RMMEDA, and newly injected functions for bias and deceptive problems. Biobjective problems are marked by ×.

the construction of LSMOP1. Fig. 5 shows that the hybrid problems are located between the LSMOP and DTLZ problems from which they are built—within the region of the WFG suite. Therefore, hybridizing these two suites may not prove to be useful for generating diverse new benchmarks. However, it does provide us with intuition toward which suites can be hybridized for diversity—but only between suites. These additional instances are useful for clarifying contradicting areas and boundaries of performance for algorithms though.

3) Injecting New Functions: The feature distributions have been used to inform where new functions with certain characteristics should be injected. We note that obclust_max captures the bias toward regions and unimodality, avg_rank_nbr_best measures the deceptiveness and ruggedness, while inf_avg_rws captures objective correlation. The numbers denoted in Fig. 6 relate to the instance numbers in Table IV. Instance types 1–4 and 8 generate in the upper region—the area which it is expected to since the triobjective LSMOP problems are located in this area. However, we note that these functions do generate problems that push further toward the top left than the LSMOP problems.

Attempting to create problems in the left region initially proved more complex, with a limited reach for deceptive problems using the LSMOP suite construction. In particular, the lack of problems with objective correlation leaves this region unoccupied. However, since objective correlation shares a relationship with the landscape of the local Pareto-optimal solutions, we targeted this area by forcing a stronger dominance relation.

Upon reducing the upper bounds to force stronger objective correlation, the instances move toward the left. Notably, this comparison is visible by the movement of instance type 6. Similarly, instance type 8 remains in the upper region since they have not utilized the reduced upper bounds, while types 5 and 7 do, and are located in the lower left region. We also note that by instantiating these problems with different parameters, we are able to reach a larger area than with a single instance. Therefore, utilizing the insights from the

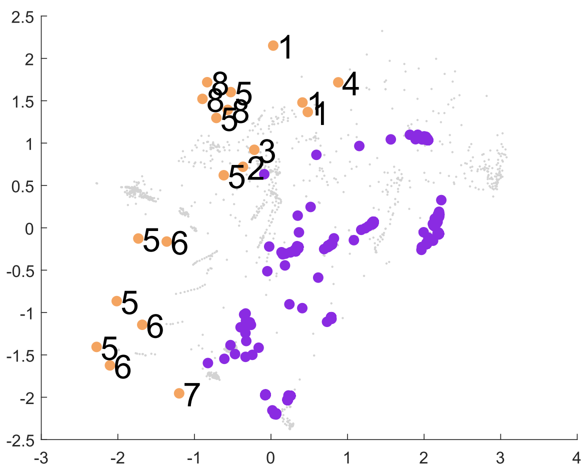
Fig. 6. Focused graph of the tuned and injected functions. Numbers on the plot for the injected functions detail the instance type generated from Table IV.

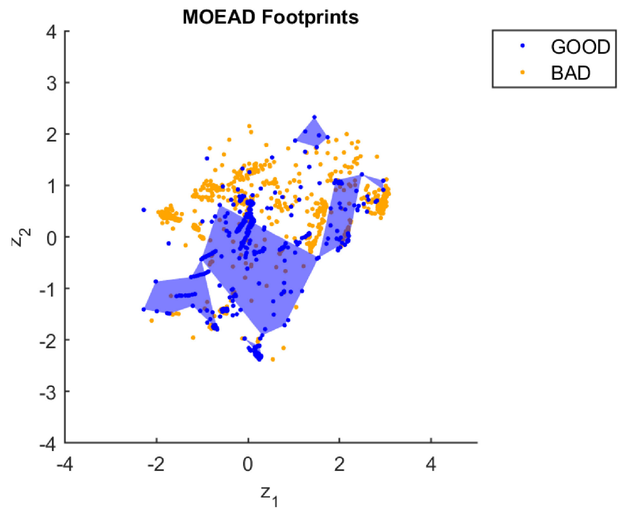

Fig. 7. Footprint for MOEA/D obtained after using the complete set of benchmark problems and the performance obtained experimentally.

instance space has allowed us to increase problem diversity by generating instances in regions of interest.

While we have demonstrated the addition of new problems onto the existing instance space, it is important to note that bias may be introduced when an excessive number of new instances are generated toward one area of the instance space. It is also necessary to recognize that upon the inclusion of a large number of new instances, the resultant instance space will likely change shape. Therefore, such bias can be avoided by conservatively filling new areas, and by iteratively repeating the ISA methodology to ensure that we are continuing to generate instances in sparse regions. Furthermore, the removal of similar instances can be investigated to reduce the number of redundant instances that may cause oversampling bias. Therefore, each progressive iteration of the instance space provides us with the most up-to-date information from which we can draw conclusions applicable only to the studied instances. Notably, the purpose of the ISA methodology is to enable the assessment of the strengths and weaknesses of algorithms once the metadata (including the instance set) reaches convergence. Therefore, a strong dependency exists between the instance space and the metadata. However, in this article, we do not comprehensively explore the robustness of the instance space construction to the metadata. Instead, we explore methods of instance generation to fill gaps en route to creating a final instance space.

B. Effect on Algorithm Performance

While the diversity of problems is important, it is also necessary to consider whether the performance of algorithms changes on these new problems. This allows us to validate whether the new information can advance our understanding of the boundaries of performance of an algorithm. Previously, we noted that IBEA had the largest coverage of good performance, and this persists with the inclusion of new problems. The footprint MOEA/D, which previously had an area of 0.310, increased by over 45% to 0.451, being the algorithm that had the second largest change in performance. Furthermore, the footprint of MOEA/D is the most disjoint, with no consistent footprint section that is larger than the rest. Therefore, will focus our analysis on MOEA/D. The footprints for all algorithms can be found in the supplementary materials.

Fig. 3(d) shows the footprint of MOEA/D on the original instances, whereas Fig. 7 shows the updated footprint after including the new instances, showing an expansion of the area of good performance with the additional information provided by the new functions. In particular, MOEA/D's footprint in the regions below the origin is now much larger. Notably, the area to the upper right previously did not form part of its footprint, but now there are demonstrated strengths. Furthermore, newly generated instances through function injection on the left and upper right also confirm good performance, resulting in a change in the shape of the footprint. Previously, there was a lack of instances in these regions, and therefore uncertainty on whether MOEA/D would perform well on such instances. However, with the inclusion of additional instances, new regions of good performance have emerged.

The change in the footprint provides additional confidence in MOEA/D across the instance space by the inclusion of new instances. MOEA/D mostly suffers poorer performance in the right and upper left regions of the instance space. Notably, when considering only the Benchmark Instances, MOEA/D's performance is understated in several regions. This validates the ongoing requirement for the diversity of problems to investigate and correctly identify algorithm footprints. The updated footprint generated is still not definitive, as there are still gaps in the instance space. Therefore, a comprehensive and representative instance set is important for correctly inferring the performance of algorithms, and is an ongoing research challenge. Since this instance space is still incomplete, there will continue to be regions of unreliable performance generalization for some algorithms—due to sparse instances in these areas. As more instances are added, the ISA methodology should be repeated to allow for robust generalizations about areas of strong algorithm performance.

V. Conclusion

Despite benchmark problem suites playing a large role in algorithm testing and development, they are typically considered in isolation or independently. This limits the strength of the conclusions drawn about algorithms, since superior performance should be demonstrated using a diverse set of test problems. The ZDT, DTLZ, and WFG suites were designed to be general-purpose suites and toolkits. As such, they are expected to contain a comprehensive and representative set of problems. However, Yamamoto *et al.* [12] showed that problems within each suite tended to cluster together. This is consistent with what is known in the literature—given the numerous test suites that have since been developed.

Since the ZDT, DTLZ, and WFG suites, newer suites have been proposed with specific problems, for the purpose of testing an algorithm on its capacity to account for a combination of characteristics not typically found in the general-purpose suites. However, due to the differences between each suite and their considerations, their contributions toward a more comprehensive problem set is not transparent.

In this article, we explored the diversity of nine of the current continuous MO benchmark suites using ISA. Our research confirmed the results of Yamamoto *et al.* for the ZDT, DTLZ, and WFG suites. Furthermore, we observed that within six other suites, problems still had a tendency to cluster together. While diversity across the problem instances is present, we also identified opportunities for the development of new benchmarks, including problems with low correlation, high correlation, bias, and deceptiveness. The contributions of this article by creating and analyzing the first instance space for MO problems are twofold: 1) the identification of which problems are missing and the features we should target to create them and 2) the generation of problems which are tuneable, providing potential for filling a larger surrounding area of the instance space to create diverse and comprehensive benchmarks.

We explored three methods for generating new problems within the instance space perspective: 1) problem tuning; 2) hybridizing; and 3) injecting new functions through analyzing the feature distributions. The three methods were shown to address different limitations. Method 1 using problem tuning showed the potential to fill the area around the existing instances. This is particularly useful for generating a larger density of instances with similar characteristics, for example, to identify the partition in feature characteristics between an algorithm performing strongly or weakly. Method 2 of hybridizing was able to fill in gaps between the existing suites. This is useful for generating instances in gaps in the instance space's interior and is simple to define since we utilize existing functions. However, the two methods are limited in their flexibility when attempting to define problems which exist toward the boundaries of the instance space. Method 3, which injects new functions into the LSMOP toolkit using insights from the feature distributions, shows more potential for generating diverse instances which are important for a comprehensive problem set, as well as defining the true boundaries of an algorithm's footprint. In the future, the identification of further features which scale with the number of objectives will be necessary to generate instance spaces beyond triobjective.

By analyzing feature distributions and generating functions with known characteristics, we have been able to address gaps in the instance space by creating biased and deceptive instances, as well as those with higher objective correlation. However, due to the interplay of conflicting objectives, the injection of functions for method 3 may not result in transparent movement across the instance space. Furthermore, our results show that problem suites tend to be restricted in their movement across the instance space due to the composition of their problem generation. However, variable linkages show potential in moving problems within the instance space, as demonstrated by the RMMEDA and IMMOEA linkages on ZDT and DTLZ problems. Similarly, reducing the upper bounds for decision variables (and therefore sparseness) resulted in a shift away from the initial LSMOP suite. This is an intuitive result since LSMOP is a large-scale problem toolkit, and by reducing the large-scale nature of the problems, LSMOP demonstrates a different range of problems. Therefore, the clustering limitation observed may be overcome by strategically changing suite components beyond objective functions.

The increased diversity in problems also demonstrated an impact on algorithm performance. Upon the inclusion of instances in the exterior of the instance space, we were able to both clarify contradictions in algorithm footprints, as well as infer larger areas of good performance for each algorithm. In particular, the footprint of MOEA/D not only grew but also changed shape. This demonstrates the importance of a comprehensive and representative test set for understanding algorithms, since our knowledge about strong algorithm performance is only verifiable where test instances are available.

Overall, we find that these three methods are a useful intermediate step for generating a comprehensive continuous MO benchmark set. However, limitations imposed by the composition of test suites demonstrate the need to find a more sustainable method of generating MO benchmarks, as there is still more diversity to be achieved, especially toward the boundary points of the instance space. While exploring feature distributions can help inform what type of problem characteristics will occupy target regions, the function combination required to reach an exact target point is difficult to establish and requires creativity and intuition. We, therefore, consider the identification of new functions within this framework to be an ongoing challenge. One alternative strategy worth exploring in future research is to evolve new problems that lie at target locations in the instance space, with a specific feature combination, through the use of genetic programming, expanding our earlier work on generating new single-objective BBO functions [57], [58]. This may provide an automated approach for filling the instance space, made feasible by considering feature information available from ISA, and using target feature combinations to inform the combination of functions which fill gaps. It is also necessary to recognize that these artificially generated problems will not necessarily be representative of real-world problems. An interesting future direction would be the identification of regions where various real-world problems lie, and therefore algorithms performance comparisons can be inferred within real-world contexts. However, there may be limitations since many real-world problems are expensive and obtaining a large sample and their corresponding feature vectors may not be possible.

While our study has focused on analysis across problems and suites, further opportunities exist for the investigation of within-instance structures of problems. In particular, visualization of instance landscapes [59] is complementary to ISA. Such studies allow for generation of greater insights through evaluation of instances through a more granular lens. Furthermore, an analysis of the sensitivity of features to the sample size is also necessary. In particular, there is an opportunity to identify the best sampling methods and sizes for such problems, as we do not explore this in this article.

Ultimately, this study demonstrates that the instance space is not yet complete and more data is required. The development of new suites, features, and different algorithm portfolios can be added iteratively to improve upon the current instance space. The superset of test suites can be considered comprehensive when there are no gaps left to fill in the instance space. For computational efficiency, it is a natural next challenge to determine the most representative benchmark set possible to allow for unbiased comparison of algorithms [60], [61].

Acknowledgment

The authors gratefully acknowledge the support of Dr. Neelofar in developing the online ISA tool MATILDA. The materials and contents necessary for reproducing this study can be found at MATILDA and on GitHub [62].

References

[1] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann, "Automated algorithm selection: Survey and perspectives," *Evol. Comput.*, vol. 27, no. 1, pp. 3–45, Mar. 2019.

[2] K. M. Malan and A. P. Engelbrecht, "A survey of techniques for characterising fitness landscapes and some possible ways forward," *Inf. Sci.*, vol. 241, pp. 148–163, Aug. 2013.

[3] J. Rice, "The algorithm selection problem," *Adv. Comput.*, vol. 15, pp. 65–118, Jan. 1976.

[4] K. Deb, "Multi-objective genetic algorithms: Problem difficulties and construction of test problems," *Evol. Comput.*, vol. 7, no. 3, pp. 205–230, 1999.

[5] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 477–506, Oct. 2006.

[6] T. Bartz-Beielstein *et al.*, "Benchmarking in optimization: Best practice and open issues," 2020, *arXiv:2007.03488*.

[7] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, *Scalable Test Problems for Evolutionary Multiobjective Optimization*. London, U.K.: Springer, 2005, pp. 105–145.

[8] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, Jun. 2000.

[9] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "Test problems for large-scale multiobjective and many-objective optimization," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4108–4121, Dec. 2017.

[10] H. Li, Q. Zhang, and J. Deng, "Biased multiobjective optimization and decomposition algorithm," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 52–66, Jan. 2017.

[11] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32–49, 2011.

[12] K. Yamamoto, T. Takagi, K. Takadama, and H. Sato, "Visual mapping of multi-objective optimization problems and evolutionary algorithms," in *Proc. Genet. Evol. Comput. Conf. Companion*, New York, NY, USA, 2020, pp. 1872–1879.

[13] E. Yap, M. A. Muñoz, and K. Smith-Miles, "On the diversity and robustness of parameterised multi-objective test suites," *App. Soft Comput.*, vol. 110, Oct. 2021, Art. no. 107613.

[14] K. Smith-Miles, D. Baatar, B. Wreford, and R. Lewis, "Towards objective measures of algorithm performance across instance space," *Comput. Oper. Res.*, vol. 45, pp. 12–24, May 2014.

[15] P. Kerschke and H. Trautmann, "The R-package FLACCO for exploratory landscape analysis with applications to multi-objective optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2016, pp. 5262–5269.

[16] A. Liefooghe, F. Daolio, S. Verel, B. Derbel, H. Aguirre, and K. Tanaka, "Landscape-aware performance prediction for evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 24, no. 6, pp. 1063–1077, Dec. 2020.

[17] H. Ishibuchi, Y. Setoguchi, H. Masuda, and Y. Nojima, "Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 169–190, Apr. 2017.

[18] K. Smith-Miles. "Melbourne algorithm test instance library with data analytics (MATILDA)." 2020. [Online]. Available: https://matilda.unimelb.edu.au

[19] M. Muñoz. "Instance space analysis: A toolkit for the assessment of algorithmic power." 2020. [Online]. Available: https://github.com/andremun/InstanceSpace

[20] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: A MATLAB platform for evolutionary multi-objective optimization," *IEEE Comput. Intell. Mag.*, vol. 12, no. 4, pp. 73–87, Nov. 2017.

[21] R. Viennet, C. Fonteix, and I. Marc, "Multicriteria optimization using a genetic algorithm for determining a Pareto set," *Int. J. Syst. Sci*, vol. 27, no. 2, pp. 255–260, 1996.

[22] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 41–63, Feb. 2008.

[23] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, "A multiobjective evolutionary algorithm using Gaussian process-based inverse modeling," *IEEE Trans. Evol. Comput.*, vol. 19, no. 6, pp. 838–856, Dec. 2015.

[24] Y. Tian, R. Cheng, X. Zhang, M. Li, and Y. Jin, "Diversity assessment of multi-objective evolutionary algorithms: Performance metric and benchmark problems [research frontier]," *IEEE Comput. Intell. Mag.*, vol. 14, no. 3, pp. 61–74, Aug. 2019.

[25] Y. Tian, X. Zhang, C. Wang, and Y. Jin, "An evolutionary algorithm for large-scale sparse multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 380–393, Apr. 2020.

[26] A. Zhang, Q. Zhou, S. Zhao, P. Suganthan, W. Liu, and S. Tiwari, "Multiobjective optimization test instances for the CEC 2009 special session and competition," Dept. Comput. Sci., Univ. Essex Colchester, U.K., Nanyang Technol. Univ., Singapore, Rep. CES-887, Jan. 2008.

[27] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Stat. Sci.*, vol. 4, no. 4, pp. 409–423, 1989.

[28] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *J. Global Optim.*, vol. 13, no. 4, pp. 455–492, Dec. 1998.

[29] O. Mersmann, M. Preuss, and H. Trautmann, "Benchmarking evolutionary algorithms: Towards exploratory landscape analysis," in *Parallel Problem Solving From Nature (PPSN XI)*. Berlin, Germany: Springer, 2010, pp. 73–82.

[30] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph, "Exploratory landscape analysis," in *Proc. 13th Annu. Conf. Genet. Evol. Comput.*, 2011, pp. 829–836.

[31] M. A. Muñoz, M. Kirley, and S. K. Halgamuge, "Exploratory landscape analysis of continuous space optimization problems using information content," *IEEE Trans. Evol. Comput.*, vol. 19, no. 1, pp. 74–87, Feb. 2015.

[32] K. M. Malan, "A survey of advances in landscape analysis for optimisation," *Algorithms*, vol. 14, no. 2, p. 40, 2021.

[33] G. Lu, J. Li, and X. Yao, "Fitness-probability cloud and a measure of problem hardness for evolutionary algorithms," in *Evolutionary Computation in Combinatorial Optimization*. Berlin, Germany: Springer, 2011, pp. 108–117.

[34] J. Knowles and D. Corne, "Instance generators and test suites for the multiobjective quadratic assignment problem," in *Evolutionary Multi-Criterion Optimization*. Berlin, Germany: Springer, 2003, pp. 295–310.

[35] D. Stowell and M. D. Plumbley, “Fast multidimensional entropy estimation by *k*-d partitioning,” *IEEE Signal Process. Lett.*, vol. 16, no. 6, pp. 537–540, Jun. 2009.
[36] D.-I. Seo and B.-R. Moon, “An information-theoretic analysis on the interactions of variables in combinatorial optimization problems,” *Evol. Comput.*, vol. 15, no. 2, pp. 169–198, Jun. 2007.
[37] H. E. Aguirre and K. Tanaka, “Working principles, behavior, and performance of MOEAs on MNK-landscapes,” *Eur. J. Oper. Res.*, vol. 181, no. 3, pp. 1670–1690, 2007.
[38] L. Paquete, T. Schiavinotto, and T. Stützle, “On local optima in multiobjective combinatorial optimization problems,” *Ann. Oper. Res.*, vol. 156, no. 1, pp. 83–97, 2007.
[39] D. Garrett and D. Dasgupta, “Multiobjective landscape analysis and the generalized assignment problem,” in *Learning and Intelligent Optimization*. Berlin, Germany: Springer, 2008, pp. 110–124.
[40] L. Paquete and T. Stützle, “Clusters of non-dominated solutions in multiobjective combinatorial optimization: An experimental analysis,” in *Multiobjective Programming and Goal Programming*. Berlin, Germany: Springer, 2009, pp. 69–77.
[41] S. Verel, A. Liefooghe, L. Jourdan, and C. Dhaenens, “On the structure of multiobjective combinatorial search space: MNK-landscapes with correlated objectives,” *Eur. J. Oper. Res.*, vol. 227, no. 2, pp. 331–342, 2013.
[42] P. Kerschke *et al.*, “Towards analyzing multimodality of continuous multiobjective landscapes,” in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2016, pp. 962–972.
[43] P. Kerschke and C. Grimme, “An expedition to multimodal multi-objective optimization landscapes,” in *Proc. Int. Conf. Evol. Multi-Criterion Optim.*, 2017, pp. 329–343.
[44] M. A. Muñoz, M. Kirley, and K. Smith-Miles, “Analyzing randomness effects on the reliability of exploratory landscape analysis,” *Nat. Comput.*, vol. 21, pp. 131–154, Feb. 2021.
[45] N. Srinivas and K. Deb, “Muiltiobjective optimization using nondominated sorting in genetic algorithms,” *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1994.
[46] E. Yap, M. A. Muñoz, K. Smith-Miles, and A. Liefooghe, “Instance space analysis of combinatorial multi-objective optimization problems,” in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2020, pp. 1–8.
[47] Z. He and G. G. Yen, “Comparison of many-objective evolutionary algorithms using performance metrics ensemble,” *Adv. Eng. Softw.*, vol. 76, pp. 1–8, Oct. 2014.
[48] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[49] E. Zitzler, M. Laumanns, and 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*. Athens, Greece: Int. Center Numer. Methods Eng., 2001, pp. 95–100.
[50] Q. Zhang and H. Li, “MOEA/D: A multiobjective evolutionary algorithm based on decomposition,” *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.
[51] S. Yang, M. Li, X. Liu, and J. Zheng, “A grid-based evolutionary algorithm for many-objective optimization,” *IEEE Trans. Evol. Comput.*, vol. 17, no. 5, pp. 721–736, Oct. 2013.
[52] J. Bader and E. Zitzler, “HypE: An algorithm for fast hypervolume-based many-objective optimization,” *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, Mar. 2011.
[53] E. Zitzler and S. Künzli, “Indicator-based selection in multiobjective search,” in *Parallel Problem Solving From Nature (PPSN VIII)*. Berlin, Germany: Springer, 2004, pp. 832–842.
[54] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, “Modified distance calculation in generational distance and inverted generational distance,” in *Evolutionary Multi-Criterion Optimization*. Cham, Switzerland: Springer Int., 2015, pp. 110–125.
[55] Y. Xu, R. Qu, and R. Li, “A simulated annealing based genetic local search algorithm for multi-objective multicast routing problems,” *Ann. Oper. Res*, vol. 206, pp. 527–555, Feb. 2013.
[56] H. Ishibuchi, Y. Hitotsuyanagi, H. Ohyanagi, and Y. Nojima, “Effects of the existence of highly correlated objectives on the behavior of MOEA/D,” in *Evolutionary Multi-Criterion Optimization*. Berlin, Germany: Springer, 2011, pp. 166–181.
[57] K. Smith-Miles and S. Bowly, “Generating new test instances by evolving in instance space,” *Comput. Oper. Res.*, vol. 63, pp. 102–113, Nov. 2015.
[58] M. A. Muñoz and K. Smith-Miles, “Generating new space-filling test instances for continuous black-box optimization,” *Evol. Comput.*, vol. 28, no. 3, pp. 379–404, Sep. 2020.
[59] L. Schäpermeier, C. Grimme, and P. Kerschke, “To boldly show what no one has seen before: A dashboard for visualizing multi-objective landscapes,” in *Evolutionary Multi-Criterion Optimization*. Cham, Switzerland: Springer Int., 2021, pp. 632–644.
[60] T. Weise, X. Wang, Q. Qi, B. Li, and K. Tang, “Automatically discovering clusters of algorithm and problem instance behaviors as well as their causes from experimental data, algorithm setups, and instance features,” *App. Soft Comput.*, vol. 73, pp. 366–382, Dec. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494618304903
[61] T. Weise, Y. Chen, X. Li, and Z. Wu, “Selecting a diverse set of benchmark instances from a tunable model problem for black-box discrete optimization algorithms,” *App. Soft Comput.*, vol. 92, Jul. 2020, Art. no. 106269. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S156849462030209X
[62] E. Yap. “Continuous multi-objective optimisation benchmark analysis.” 2021. [Online]. Available: https://github.com/estefaniay9/continuousMOP_ISA/

Estefania Yap received the B.Sc. degree (Hons.) in statistics from RMIT University, Melbourne, VIC, Australia, in 2014, and the Ph.D. degree in operations research from The University of Melbourne, Melbourne, in 2022.

She is currently a Researcher with the Melbourne Defence Enterprise, The University of Melbourne. Her research interests lie in optimization, evolutionary computation, and machine learning.

Mario Andrés Muñoz received the B.Eng. and M.Eng. degrees in electronics engineering from the Universidad del Valle, Cali, Colombia, in 2005 and 2008, respectively, and the Ph.D. degree in engineering from The University of Melbourne, Melbourne, VIC, Australia, in 2014.

He is currently a Researcher with the School of Computer and Information Systems, The University of Melbourne; and the ARC Training Centre in Optimisation Technologies, Integrated Methodologies and Applications (OPTIMA). He has published over 50 papers. His research interests focus on the application of optimization, computational intelligence, signal processing, data analysis, and machine learning methods to ill-defined science, engineering, and medicine problems.

Kate Smith-Miles (Senior Member, IEEE) received the B.Sc. degree (Hons.) in mathematics and the Ph.D. degree in electrical engineering from The University of Melbourne, Melbourne, VIC, Australia, in 1993 and 1996, respectively.

She is a Professor of Applied Mathematics with The University of Melbourne, where she holds the title of Melbourne Laureate Professor. She is the Founder and the Director of a doctoral training center with industrial partners, known as the ARC Training Centre in Optimisation Technologies, Integrated Methodologies and Applications (OPTIMA). She is also the Associate Dean (Enterprise and Innovation) of the Faculty of Science, The University of Melbourne. She has published two books on neural networks and data mining, and over 280 refereed journal and international conference papers in the areas of neural networks, optimization, machine learning, instance space analysis, and various applied mathematics topics.

Prof. Smith-Miles is a Fellow of the Australian Academy of Science, the Institute of Engineers Australia, and the Australian Mathematical Society.