Using Well-Understood Single-Objective Functions in Multiobjective Black-Box Optimization Test Suites

Dimo Brockhoff firstname.lastname@inria.fr
Inria, CMAP, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, France

Anne Auger firstname.lastname@inria.fr
Inria, CMAP, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, France

Nikolaus Hansen firstname.lastname@inria.fr
Inria, CMAP, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, France

Tea Tušar firstname.lastname@ijs.si
Jožef Stefan Institute, Ljubljana, Slovenia

Abstract
Several test function suites are being used for numerical benchmarking of multiobjective optimization algorithms. While they have some desirable properties, like well-understood Pareto sets and Pareto fronts of various shapes, most of the currently used functions possess characteristics that are arguably under-represented in real-world problems such as separability, optima located exactly at the boundary constraints, and the existence of variables that solely control the distance between a solution and the Pareto front. Via the alternative construction of combining existing single-objective problems from the literature, we describe the `bbob-biobj` test suite with 55 bi-objective functions in continuous domain, and its extended version with 92 biobjective functions (`bbob-biobj-ext`). Both test suites have been implemented in the COCO platform for black-box optimization benchmarking and various visualizations of the test functions are shown to reveal their properties. Besides providing details on the construction of these problems and presenting their (known) properties, this paper also aims at giving the rationale behind our approach in terms of groups of functions with similar properties, objective space normalization, and problem instances. The latter allows us to easily compare the performance of deterministic and stochastic solvers, which is an often overlooked issue in benchmarking.

Keywords
Black-box optimization benchmarking, multiobjective optimization, algorithm comparison, benchmark suite generator.

1 Introduction

Numerical benchmarking is an important part of (black-box) optimization that helps to understand algorithm behavior and recommend algorithms. In order to obtain meaningful results, a benchmarking experiment should be (i) based on a thorough, well-documented and well-understood methodology and (ii) either be conducted on real-world problems of interest or a collection of artificial test functions that possess comprehensible difficulties observed in practical optimization problems. This holds true

This is the author's final version that has been accepted for publication in Evolutionary Computation.

for both single- and multiobjective problems but for the latter, the methodology is less advanced at the moment.

Many artificial test functions that are frequently used in multiobjective optimization have been derived by setting up the Pareto front shape first without relating it to the intrinsic difficulties of the objective functions. Such an approach has the advantage that the analytical forms of the Pareto front and the Pareto set can be exploited to facilitate the performance assessment. Another aspect of state-of-the-art test suites for multiobjective optimization is the fact that not much progress has been made to avoid the overrepresentation of functions that are too simple or have questionable properties. Several existing (and still frequently used) multiobjective test suites, for example, contain a large share of functions that are separable, have the Pareto set on the domain boundary, or contain distance and position variables[1] —artificial features not reflecting well the difficult black-box problems observed in practice.

In the context of single-objective algorithm benchmarking, a lot of progress has been made in recent years in the design of artificial test functions that represent a wide range of difficulties observed in practice. The black-box optimization benchmarking test suite (`bbob`, Hansen et al. (2009)) in particular has received wide acceptance as its 24 test functions have various advantages over previous test suites. The functions are well understood and expose algorithms to a variety of real-world difficulties such as multimodality, ill-conditioning, non-separability of the variables, and non-linearities. The `bbob` functions are grouped into five function groups with functions within a group sharing similar difficulties (such as multimodality with weak global structure) and with the aim to not overemphasize certain difficulties. Each function has one or several concrete scientific questions associated with it that can be answered by looking at algorithm performance results on that function (or in combination with another function). General statements beyond the tested concrete functions are possible by testing invariance properties of algorithms such as scaling, rotation and affine invariance. In contrast to the previously mentioned approaches to building multiobjective test suites, we suggest to focus on introducing the known difficulties of real-world problems into the test suite. This is analogous to the single-objective case, but has the disadvantage that analytical formulas for the Pareto front and Pareto set might not be available. The motivation behind this approach is that in practice, multiobjective problems are constructed in exactly this way—with each objective corresponding to a separate single-objective function. Concretely, we propose a generic way to combine the well-established and -understood single-objective functions from the `bbob` test suite (Hansen et al., 2009). Using the `bbob` test functions as building blocks allows us to build upon a careful statistical choice of the functions (without overrepresenting a certain type of problem) as well as comprehensive difficulties. In turn, the proposed multiobjective functions are more likely to be practically relevant than previous benchmarks. In particular, our proposal fulfills all five recommendations for benchmark suites mentioned by Huband et al. (2006, page 485), see also Section 3. We showcase our idea by implementing two biobjective test suites within the COCO platform (Hansen et al., 2021) that supports automated benchmarking. The implementation within the COCO platform has two important advantages over many other existing test suites: (1) numerical experiments can be done more easily by automation, and, even more importantly,

[1]A function is said to have a *distance variable* if changing this variable only results in dominating or dominated solutions. In other words, a distance variable determines solely the distance of a solution from the Pareto front. A *position variable*, in turn, only results in incomparable solutions when changed, see Huband et al. (2006) for details.

(2) it is possible to compare new results with a large variety of already benchmarked algorithms and to share the results effortlessly. As of end of 2020, the results of 32 algorithm implementations[2] have been collected and made available online[3].

The lack of analytical expressions for the Pareto sets and fronts in our approach is addressed by collecting and visualizing approximations from numerical experiments with a large variety of algorithms. The collected hypervolume values are available online for performance assessment[4]. To understand the quality of the reference Pareto set approximations, we investigate necessary optimality conditions in Section 6. We also provide dominance- and gradient-based plots to gain additional insight into the problem properties.

The non-existence of analytical forms of Pareto set and Pareto front in our approach can be even seen as an advantage: the combination of existing single-objective test functions allows, in a controlled way, to mimic the typical constructions of real-world problems and to empirically investigate the resulting Pareto set and Pareto front shapes from such constructions. Moreover, the proposed multiobjective benchmark functions come in the form of pseudo-random *instances*, which allows to more easily compare deterministic and stochastic approaches and makes it possible to investigate the variance of properties like a connected or convex Pareto front emanating from the combination of single-objective test functions.

The contributions of this paper are: (i) the purposeful choice of single-objective functions and their combinations into biobjective problems, (ii) cherry-picking instances to avoid pathological combinations, (iii) a classification of the problems, (iv) implementation of two suites for the COCO platform, (v) identifying problem attributes and properties, (vi) provision of Pareto set approximations with an analysis of their quality, (vii) various visualizations of the constructed functions (see also the supplementary material[5]), (viii) empirical identification of target values, and (ix) a guide to performance assessment.

The paper is organized as follows. We start by outlining the fundamental definitions in multiobjective optimization and benchmarking in Section 2 and review multiobjective benchmark suites and their properties in Section 3. Section 4 introduces the main concepts behind the single-objective `bbob` test suite with more details given in the appendix. Section 5 proposes the `bbob-biobj` and `bbob-biobj-ext` test suites which are then analyzed visually in Section 6. The paper concludes with a discussion of how to report algorithm performance data in Section 7 and final remarks in Section 8.

2 Preliminaries

We consider biobjective, unconstrained **minimization** problems of the form

$$\min_{x\in\mathbb{R}^n} F(x) = (f_\alpha(x), f_\beta(x)),$$

where n is the number of variables of the problem (also called the problem dimension), $f_\alpha : \mathbb{R}^n \to \mathbb{R}$ and $f_\beta : \mathbb{R}^n \to \mathbb{R}$ are the two objective functions, and the $\min$ operator is

[2]Updated list of algorithm data sets: `https://numbbo.github.io/data-archive/bbob-biobj/`.

[3]Postprocessed performances of most available algorithms are displayed online at `https://numbbo.github.io/ppdata-archive/bbob-biobj/2016-all/` and `https://numbbo.github.io/ppdata-archive/bbob-biobj/2019-all/`.

[4]The best known hypervolume values for all supported test instances are available via the COCO platform at `https://github.com/numbbo/coco/blob/master/code-experiments/src/suite_biobj_best_values_hyp.c`. To create these hypervolume "reference" values, we relied on data from a large variety of numerical experiments—for details, see Section 6.1.

[5]Webpage with supplementary material: `https://numbbo.github.io/bbob-biobj`.

related to the standard *dominance relation*. A solution $x \in \mathbb{R}^n$ is thereby said to *dominate* another solution $y \in \mathbb{R}^n$ if $f_\alpha(x) \leq f_\alpha(y)$ and $f_\beta(x) \leq f_\beta(y)$ hold and at least one of the inequalities is strict. We adopt the notation f_α for the first objective (resp. f_β for the second objective) instead of f_1 and f_2 to avoid confusion with notations adopted within the single-objective `bbob` test suite.

Solutions which are not dominated by any other solution are called *Pareto-optimal* or *efficient solutions*. All Pareto-optimal solutions constitute the *Pareto set* of which an approximation is sought. The Pareto set's image in the objective space $F(\mathbb{R}^n)$ is called the *Pareto front*. Two specific points in the objective space are important to mention:

- The *ideal point* defined as the vector in objective space that contains the optimal F-value for each objective *independently*. More precisely, if $f_\alpha^{\mathrm{opt}} := \inf_{x\in\mathbb{R}^n} f_\alpha(x)$ and $f_\beta^{\mathrm{opt}} := \inf_{x\in\mathbb{R}^n} f_\beta(x)$, the ideal point is given by $z_{\mathrm{ideal}} = (f_\alpha^{\mathrm{opt}}, f_\beta^{\mathrm{opt}})$.

- The *nadir point* (in objective space) consisting in each objective of the worst value obtained by a Pareto-optimal solution. More precisely, if we denote the Pareto set by $\mathcal{P}$, the nadir point satisfies $z_{\mathrm{nadir}} = (\sup_{x\in\mathcal{P}} f_\alpha(x), \sup_{x\in\mathcal{P}} f_\beta(x))$.

  In the specific case where each of the two objective functions has a unique global minimum (that is, a single point in the search space which maps to the global minimum function value), $z_{\mathrm{nadir}} = (f_\alpha(\arg\min f_\beta(x)), f_\beta(\arg\min f_\alpha(x)))$.

All given definitions generalize trivially to problems with more than two objectives. When solving an unconstrained multiobjective problem as the above, usually the goal is to find, with as few evaluations of F as possible, a set of non-dominated solutions which is (i) as large as possible and (ii) has objective values as close to the Pareto front as possible[6]. Here, we define this optimization goal via the maximization of a quality indicator, namely, the hypervolume (Zitzler et al., 2003) of the set of all non-dominated solutions found so far (Brockhoff et al., 2016).

In this context, an optimization algorithm addressing the above minimization problem is given an *instance* of F: we implement each generic multiobjective function F as a parametrized function $F^\theta : \mathbb{R}^n \to \mathbb{R}^m$ with a problem dimension n and a parameter value $\theta \in \Theta$ that might depend on n and $m = 2$, the number of objectives. The parameter value θ determines a *function instance*. For example, θ can encode the location of the optimum, x_α^{opt}, of the first objective $f_\alpha^\theta : \mathbb{R}^n \to \mathbb{R}$ with $f_\alpha^{x_\alpha^{\mathrm{opt}}}(x) = ||x - x_\alpha^{\mathrm{opt}}||^2$. Search space rotations and shifts in the objective values are other generic transformations that can be encoded in instances (Hansen et al., 2021).

3 Review of Existing Multiobjective Test Suites

Many multiobjective test suites have been proposed throughout the years. Here, we in particular discuss those that are scalable in the problem dimension and that are unconstrained or box-constrained and defined in the continuous domain—the focus of our proposal for a new benchmark suite.

The (evolutionary) multiobjective optimization field first performed numerical comparisons of algorithms on single, independently proposed test problems like those by Kursawe (1990) and Fonseca and Fleming (1995), see for example Tan et al. (2002), or in real-world studies, see Van Veldhuizen and Lamont (1998) for an early overview.

[6]The distance in objective space is defined here in such a way that the nadir and ideal points have in each coordinate the distance of one. Note also that finding a set of non-dominated solutions as large as possible might not always be the ultimate goal, in particular if the number of objective functions is large.

A first attempt to create a consistent multiobjective test *suite* with several problems with desired properties was, to the best of our knowledge, the work of Van Veldhuizen and Lamont (1999b,a). The authors clearly stated the need for scalable test suites and emphasized that problems of a test suite should possess practically relevant features. In the years that followed, several other scalable test suites have been proposed, of which the most established ones are (i) the ZDT suite of Zitzler et al. (2000), scalable in the number of variables but with only two objectives, (ii) its rotated version, the IHR problem suite of Igel et al. (2007), (iii) the DTLZ suite of Deb et al. (2005), with seven problems, all scalable in the number of variables and objectives, (iv) the WFG suite of Huband et al. (2006) with nine scalable problems of various difficulties, (v) the LZ suite of Li and Zhang (2009) containing problems with more complicated Pareto sets, (vi) the CEC2007 suite, combining and extending 13 existing test functions from the literature (Huang et al., 2007), (vii) the CEC2009 suite with 13 problems overall (Zhang et al., 2009), and finally (viii) the CEC2017 suite with 15 collected problems, tailored towards many-objective optimization (Cheng et al., 2017).

Most of these test suites have some desirable properties like well-understood Pareto sets and Pareto fronts with shapes of various kinds (linear, convex, concave, discontinuous). But they also possess artificial characteristics that stem from the easier construction of such problems—overrepresenting properties such as no or only few dependencies among variables, Pareto sets located exactly at the boundary constraints, and the differentiation between position and distance variables. Although, for example, the importance of non-separable test functions in single-objective test suites is unquestioned and even Deb (2001, p.353f) states its significance, most proposed multiobjective test problems are still separable or mostly separable in the sense that a function is separable if it can be optimized variable by variable.

Another way of generating multiobjective test problems is to decide on the single objectives and simply combine them to a multiobjective problem. Examples for this are the double sphere or Schaffer function number 1 (Schaffer, 1985), the combination of convex quadratic functions such as ellipsoids and cigtab functions by Igel et al. (2007) or the multimodal functions based on spherical functions by Emmerich and Deutz (2007), the latter construction of which has been extended by Kerschke et al. (2016).

Even though all test suites in the above list are scalable in the problem dimension, we rarely see performance studies that investigate the scaling of the algorithms with the problem dimension.

The arguably most complete paper on the topic of multiobjective benchmark problems to date is still the work of Huband et al. (2006) where the authors (i) identify important properties test functions should have, (ii) discuss in detail all other available test suites at that time with respect to these properties, and (iii) finally propose a new, well-motivated test suite that avoids many pitfalls of other test suites. In particular, Huband et al. (see Huband et al. (2006), page 485) recommend that multiobjective test suites should, in addition to recommendations for single-objective test suites:

1. contain a few unimodal test problems to test convergence velocity relative to different Pareto optimal geometries and bias conditions,
2. cover the three core types of Pareto optimal geometries: degenerate Pareto optimal fronts, disconnected Pareto optimal fronts, and disconnected Pareto optimal sets,
3. have a majority of its test problems multimodal with a few deceptive problems,
4. have the majority of problems nonseparable, and

5. contain problems that are both nonseparable and multimodal to be representative of real-world problems.

All five recommendations are fulfilled for the test suites proposed in this paper. Similar to the single-objective bbob functions, the WFG suite of Huband et al. (2006) employs problem transformations that change the properties like (non-)separability, bias, and the shape of the Pareto front of underlying *raw* objective functions.

One common property of the above mentioned test suites is that their Pareto sets can be described in analytical form. This certainly has an advantage for performance assessment but it also restricts the types of real-world problem characteristics that can be captured with such functions.

However, in practice, multiobjective optimization problems are typically constructed by combining objective functions that are defined (and understood) independently such as cost and performance of a new product. The objective functions might thereby come from different domains and share or do not share common properties such as uni-/multimodality, (non-)separability, asymmetry, etc.

The idea of defining multiobjective test problems by combining single-objective ones is therefore straightforward and has been proposed before, as mentioned above, for example in (Schaffer, 1985), (Igel et al., 2007), (Emmerich and Deutz, 2007) or (Horn et al., 2015). To create a benchmark suite with challenging properties observed in practice, we follow here the same path and combine some of the existing, well-established, well-understood functions of the bbob test suite to create new multiobjective suites.

4 The Single-Objective bbob Functions

Our multiobjective test suites build on the single-objective bbob function suite (Hansen et al., 2009; Finck et al., 2009). The bbob suite includes unimodal and multimodal test functions, separable and non-separable functions, well-conditioned and ill-conditioned problems, see Appendix A for a short definition of those properties. Each function is parametrized (see Section 2) and is scalable with respect to the dimension. Some function pairs allow to answer specific questions: for instance, the ellipsoid and the Rastrigin functions are present in a separable and a non-separable version such that one can investigate how much an algorithm exploits separability. The functions are unbounded but the search domain of interest is $[-5, 5]^n$. Except for the linear slope function f_5, the optimum of each function lies in the hypercube $[-4, 4]^n$ for all instances and all dimensions. The optimum of the linear function is attained at a (randomly chosen) corner of the hypercube $[-5, 5]^n$ and in the orthant beyond this point.

The bbob test suite has 24 functions, split into five groups. The **Separable** group contains five separable functions, the **Moderate** group consists of four functions with low or moderate conditioning, including multimodal functions, the **Ill-conditioned** functions group contains five unimodal functions with high conditioning, the **Multimodal** functions group comprises five multimodal functions with adequate global structure, and the **Weakly-structured** group consists of five multimodal functions with weak global structure. Problem difficulty is typically increasing from the first to the last group, but there are exceptions, for example, solving the Büche-Rastrigin function from the first group is difficult for most algorithms. The manually assigned function groups are also used to aggregate performance, allowing to make meaningful statements about the performance on functions with specific properties reflected by the groupings.

5 The Proposed Biobjective Test Suites

Given the well-motivated bbob functions from Hansen et al. (2009), it is natural to construct a multiobjective test suite from these single-objective functions. For the biobjective case, a pairwise combination of all 24 functions results in $24^2 = 576$ biobjective functions. We however assume that multiobjective optimization algorithms are not sensitive to permutations of the objectives, so that we do not include $F = (f_\beta, f_\alpha)$ if (f_α, f_β) is already present[7]. This results in the total of $\binom{25}{2} = 300$ function combinations—the number of 2-combinations drawn with replacement.

While a benchmarking suite should contain a large number of different problems to avoid overfitting, first tests in Brockhoff et al. (2015) showed that having 300 functions is impracticable in terms of the overall running time of a benchmarking experiment. Therefore, a subset of these 300 functions was selected.

We present two such selections—the `bbob-biobj` test suite with 55 functions and its extension, the `bbob-biobj-ext` test suite with 92 functions. We also provide visualizations for some of the functions, showing different Pareto set and front shapes. Plots for all 92 functions are provided via the supplementary material webpage[5].

5.1 The `bbob-biobj` Test Suite

The biobjective `bbob-biobj` test suite is created by exploiting the organization of the `bbob` functions into groups. More precisely, only two (representative) functions from each of the `bbob` function groups are chosen. This way, we do not introduce any bias towards a specific group. In addition, within each group, the functions are chosen to be the most representative without repeating similar functions. For example, only one ellipsoid, one Rastrigin, and one Gallagher function are included in the `bbob-biobj` suite although they appear in multiple versions in the `bbob` suite.

These ten `bbob` functions are chosen to create the `bbob-biobj` test suite:

- From the separable functions group: sphere function f_1 and separable ellipsoid function f_2.
- From the functions with low or moderate conditioning: attractive sector function f_6 and original Rosenbrock function f_8.
- From the unimodal functions with high conditioning: sharp ridge function f_{13} and sum of different powers function f_{14}.
- From the multimodal functions with adequate global structure: Rastrigin function f_{15} and Schaffer F7 function with condition number 10 f_{17}.
- From the multimodal functions with weak global structure: Schwefel $x \sin x$ function f_{20} and Gallagher 101 peaks function f_{21}.

Using the previously described pairwise combinations, this results in $\binom{11}{2} = 55$ biobjective functions for the final `bbob-biobj` suite, denoted as F_1 to F_{55} in the rest of the paper. Figure 1 visualizes these combinations of the chosen single-objective `bbob` functions f_1, f_2, f_6, f_8, f_{13}, f_{14}, f_{15}, f_{17}, f_{20}, and f_{21} to the 55 `bbob-biobj` functions and color-codes the resulting 15 function groups, for example the group of separable - ill-conditioned functions (F_5, F_6, F_{14}, F_{15}) where one objective comes from the separable and the other objective comes from the ill-conditioned `bbob` function group.

[7]Note that this assumption is not true for simple classical strategies such as optimizing successively a weighted sum $w \cdot f_1(x) + (1 - w) \cdot f_2(x)$ with increasing weight $0 \leq w \leq 1$.

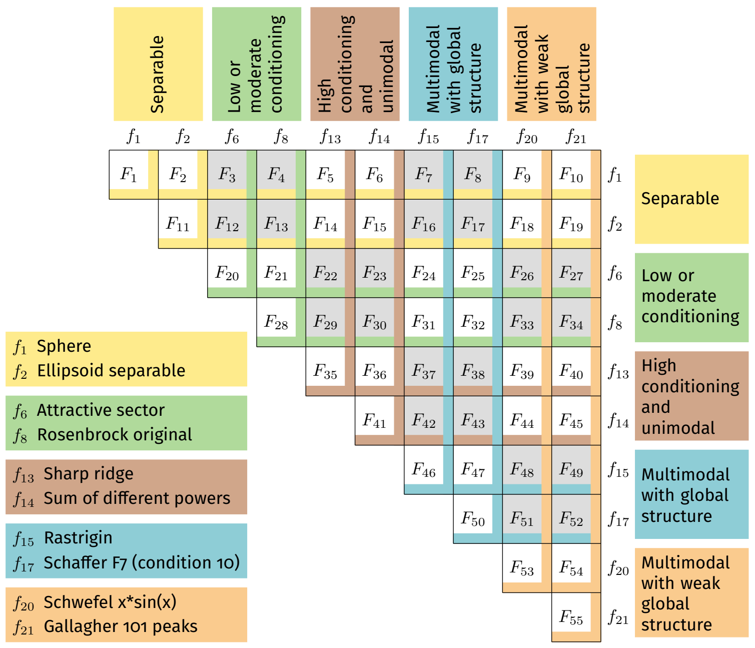
Separable
Low or moderate conditioning
High conditioning and unimodal
Multimodal with global structure
Multimodal with weak global structure
f_1 f_2 f_6 f_8 f_{13} f_{14} f_{15} f_{17} f_{20} f_{21}
F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_8 F_9 F_{10} f_1
F_{11} F_{12} F_{13} F_{14} F_{15} F_{16} F_{17} F_{18} F_{19} f_2
F_{20} F_{21} F_{22} F_{23} F_{24} F_{25} F_{26} F_{27} f_6
F_{28} F_{29} F_{30} F_{31} F_{32} F_{33} F_{34} f_8
F_{35} F_{36} F_{37} F_{38} F_{39} F_{40} f_{13}
F_{41} F_{42} F_{43} F_{44} F_{45} f_{14}
F_{46} F_{47} F_{48} F_{49} f_{15}
F_{50} F_{51} F_{52} f_{17}
F_{53} F_{54} f_{20}
F_{55} f_{21}
f_1 Sphere
f_2 Ellipsoid separable
f_6 Attractive sector
f_8 Rosenbrock original
f_{13} Sharp ridge
f_{14} Sum of different powers
f_{15} Rastrigin
f_{17} Schaffer F7 (condition 10)
f_{20} Schwefel x*sin(x)
f_{21} Gallagher 101 peaks

Figure 1: The functions of the `bbob-biobj` test suite (F_1 to F_{55}) together with the information about which single-objective `bbob` functions are used to define them (top and right annotations). Background color is used to delineate function groups.

All `bbob-biobj` functions are scalable in the search space dimension and come in the form of instances as it is the case with the original `bbob` suite.

In the following, we specify the common properties of the `bbob-biobj` functions and the main rationale behind them while concrete details on each of the 55 functions are provided via the supplementary material webpage[5].

5.1.1 Function Domain

Since we use the single-objective `bbob` functions to construct the `bbob-biobj` suite, all functions are unbounded and the extreme solutions of the Pareto set are guaranteed to lie within $[-5, 5]^n$.

Note that the Pareto set can partially lie outside of this area but that the major part of the Pareto set is expected to lie within it. Initial experiments found Pareto set approximations to be partially outside of the hypercube $[-5, 5]^n$ on a few functions in low dimensions, see also Section 6.5.

5.1.2 Normalization of Objectives

None of the 55 `bbob-biobj` functions is explicitly normalized and the optimization algorithms therefore have to cope with objective values in different ranges. Typically, different orders of magnitude between the objective values can be observed.

However, to assess performance on the test suites, we normalize the objectives based on the ideal and nadir points before calculating the hypervolume indicator (Brockhoff et al., 2016). Both points can be computed, because the unique global optimum is known for the used 10 `bbob` base functions. In the black-box optimization benchmarking setup of the COCO platform, the algorithm is allowed to use the values of the nadir point as an upper bound on a region of interest in objective space but no other normalization is directly available to the algorithm.

5.1.3 Instances

Our proposed test functions are parametrized and their instances are instantiations of the underlying parameters of the `bbob` functions (see Hansen et al., 2021)[8]. In addition, we assert two conditions:

1. The Euclidean distance between the two single-objective optimal solutions (also called the extreme solutions) in the search space is at least 10^{-4}.
2. The Euclidean distance between the ideal and the nadir point in the non-normalized objective space is at least 10^{-1}.

Each function instance has an integer ID. The relation between the instance ID, K_{ID}^{F}, of a biobjective function $F = (f_\alpha, f_\beta)$ and the instance IDs, $K_{\mathrm{ID}}^{f_\alpha}$ and $K_{\mathrm{ID}}^{f_\beta}$, of its underlying single-objective functions is $K_{\mathrm{ID}}^{f_\alpha} = 2K_{\mathrm{ID}}^{F} + 1$ and $K_{\mathrm{ID}}^{f_\beta} = K_{\mathrm{ID}}^{f_\alpha} + 1$. If the two above conditions are not satisfied for all dimensions and functions in the `bbob-biobj` suite, we increase the instance ID of the second objective successively until both properties are fulfilled. Exceptions to the above rule are, for historical reasons, the `bbob-biobj` instance IDs 1 and 2 which contain the single-objective instance IDs 2 and 4 and the IDs 3 and 5, respectively. This results in the same instances 1 to 5 here and in Brockhoff et al. (2015).

An implementation of the above procedure is in effect a test function generator. Yet we emphasize that the generated instances of a parametrized function are meant to represent problems with similar difficulties. In the following, we only use a small number of fixed instances: for each biobjective function and given dimension, the `bbob-biobj` suite contains by default 15 instances[9], which are chosen in advance and will remain fixed for a long period of time for comparability reasons[10].

Problem instances in the multiobjective case tend to differ more wildly than in the single-objective case. Even when for each single objective the different instances are similar, different combinations of them can result in very different shapes of the Pareto front (for example continuous vs. discontinuous) or in different difficulties to solve such problems (the orientation of level sets, for example, might be in accordance between the objectives or perpendicular—resulting in significantly different multiobjective problems when two ill-conditioned functions are combined)[11]. Additionally, for

[8]Performing numerical benchmarking experiments on a set of different instances of a parametrized function instead of experiments on a single fixed function has an immediate advantage: deterministic algorithms and stochastic algorithms can be compared easily in the same way stochastic algorithms are naturally compared. Running a deterministic algorithm on different instances of the same parametrized function introduces stochasticity of the runtime to reach certain target difficulties among runs in the same way as the combined stochasticity from the instance generation and the random events within a stochastic algorithm. Care, however, has to be taken that the variation of problem difficulty among instances is relatively low compared to the variation of difficulty between the actual benchmark functions. This assumption does not always hold for all instances of highly multimodal functions or for instances of most combinatorial optimization problems.

[9]In principle, as for the instance generation for the `bbob` suite, the number of possible instances for the `bbob-biobj` suite is unlimited (Hansen et al., 2021). However, even the tiniest performance difference can be made statistically significant with a high enough number of instances and repetition.

[10]Another reason to fix the instances for a long(er) time period is that no analytical form of the Pareto sets and Pareto fronts is available such that (good enough) Pareto set approximations need to be available for all instances for a proper performance assessment of algorithms.

[11]We have observed performance differences of up to two orders of magnitude (in the number of function evaluations to reach a certain hypervolume indicator precision) between instances in some cases. Differences

very different instances the target levels defined for the performance assessment may become incomparable. Consequently, we estimate the Pareto set's hypervolume per instance and do not adopt the technique from the single-objective case to aggregate results from different instances by simulated restarts (Hansen et al., 2021) to estimate runtimes to target values unattained on a given instance. The second reason to discard simulated restarts in the multiobjective case is the performance evaluation setup in COCO: we measure performance based on the entire archive of all so-far-evaluated solutions. Runs on different instances however produce incompatible archives, hence simulated restarts cannot account for the hypervolume attained by previous runs.

5.2 The `bbob-biobj-ext` Test Suite

Having all combinations of only a subset of the single-objective `bbob` functions in a test suite like the above `bbob-biobj` one has also a few disadvantages. Using only a subset of the 24 `bbob` functions introduces a bias towards the chosen functions and reduces the variety of difficulties a biobjective algorithm is exposed to. Allowing all combinations of (a subset of the) `bbob` functions also increases the percentage of problems for which both objectives are from different `bbob` function groups. In practice, however, both objective functions may often come from a similar "function domain".

The rationale behind the extended test suite, denoted as `bbob-biobj-ext`, is to reduce the mentioned effects. To this end, we add all within-group combinations of `bbob` functions which are not already in the `bbob-biobj` suite and which do not combine a function with itself. For technical reasons, we also remove the Weierstrass function f_{16} because its optimum is not necessarily unique and computing the nadir point is more challenging. This extension adds $3{\cdot}(4{+}3{+}2{+}1{-}1){+}2{\cdot}(3{+}2{+}1{-}1) = 3{\cdot}9{+}2{\cdot}5 = 37$ functions, resulting in 92 functions overall.

5.2.1 The `bbob-biobj-ext` Functions and Function Groups

Figure 2 details the single-objective `bbob` function combinations contained in the 92 `bbob-biobj-ext` functions. The first 55 `bbob-biobj-ext` functions are the same as in the `bbob-biobj` test suite for compatibility reasons. We still obtain 15 function groups to structure the 92 biobjective functions of the `bbob-biobj-ext` test suite. Depending on whether a function group combines functions from the same or from different `bbob` function groups, the new groups contain eight, 12 or just four functions.

The concrete function definitions are given in the supplementary material[5].

5.2.2 Normalization and Instances

Normalization of the objectives and instances for the `bbob-biobj-ext` test suite are handled in the same manner as for the `bbob-biobj` suite, i.e., the objective functions are not normalized and 15 instances are prescribed for a typical experiment.

5.3 Reference Implementation

Both the `bbob-biobj` and the `bbob-biobj-ext` suites have been implemented in the COCO platform which is freely available and supports access in various programming languages (Python, C/C++, Java, Matlab/Octave). After installation of COCO[12], this is as easy as typing (here in an IPython shell):

of more than one order of magnitude happen in maximally 30% of the function/dimension pairs with typical algorithms on the proposed `bbob-biobj` test suite. Due to the higher amount of multimodal functions in the `bbob-biobj-ext` suite, differences among instances are more common.

[12]Besides downloading the code, installation of the experiments part with access to the function suites is a single call `python do.py build-LANGUAGEOFINTEREST`.

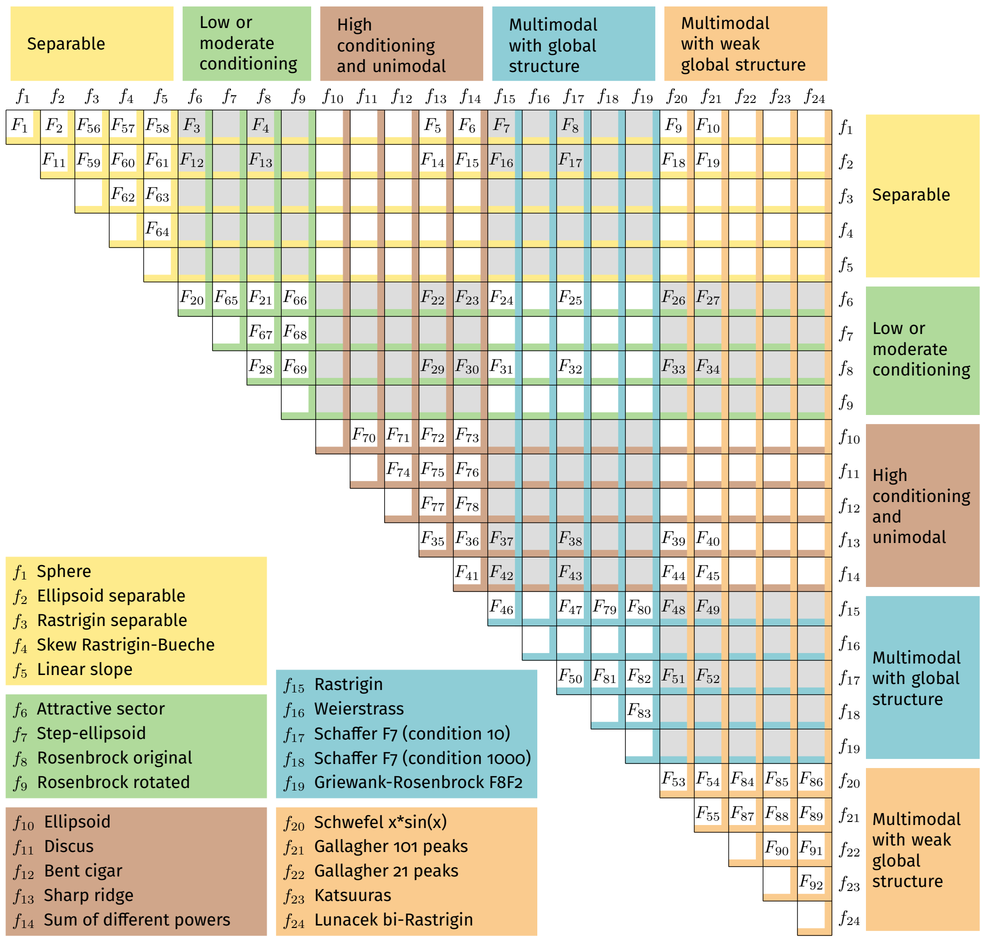
Separable
Low or moderate conditioning
High conditioning and unimodal
Multimodal with global structure
Multimodal with weak global structure
f_1 Sphere
f_2 Ellipsoid separable
f_3 Rastrigin separable
f_4 Skew Rastrigin-Bueche
f_5 Linear slope
f_6 Attractive sector
f_7 Step-ellipsoid
f_8 Rosenbrock original
f_9 Rosenbrock rotated
f_{10} Ellipsoid
f_{11} Discus
f_{12} Bent cigar
f_{13} Sharp ridge
f_{14} Sum of different powers
f_{15} Rastrigin
f_{16} Weierstrass
f_{17} Schaffer F7 (condition 10)
f_{18} Schaffer F7 (condition 1000)
f_{19} Griewank-Rosenbrock F8F2
f_{20} Schwefel x*sin(x)
f_{21} Gallagher 101 peaks
f_{22} Gallagher 21 peaks
f_{23} Katsuuras
f_{24} Lunacek bi-Rastrigin

Figure 2: The functions of the `bbob-biobj-ext` test suite (F_1 to F_{92}) together with the information about which single-objective `bbob` functions are used to define them (top and right annotations). Background color is used to delineate function groups.

```
In [1]: import cocoex
In [2]: biobj_suite = cocoex.Suite('bbob-biobj', '', '')
In [3]: fun = biobj_suite.get_problem_by_function_dimension_instance(17, 10, 1)
In [4]: fun([0]*10)    # evaluate in origin
Out[4]: array([3.33606646e+06, 5.31128506e+01])
```

6 Insights into Problem Properties

In order to better understand the functions in the `bbob-biobj` and `bbob-biobj-ext` suites, we visualize and analyze their properties. We display their best known Pareto set and Pareto front approximations together with 1-dimensional search space cuts in Section 6.1, investigate a necessary optimality condition in Section 6.2, and provide 2-dimensional dominance-based (Section 6.3) and gradient-based plots (Section 6.4) known from the literature for all proposed functions. All discussed plots are available online at the supplementary material webpage[5].

6.1 Plots of Pareto Set and Front Approximations

In this section, we plot the best currently known approximations of the Pareto set and the Pareto front, obtained from running many algorithms and collecting the non-dominated solutions, evaluated during those runs[13].

Figure 3 shows exemplary plots for three functions of the `bbob-biobj` suite in dimension 5: the double sphere function $F_1 = (f_1, f_1)$ with a continuous Pareto front and a straight line as the Pareto set (left column), the sphere/Gallagher 101 peaks function $F_{10} = (f_1, f_{21})$ with a continuous Pareto front but a gap in the Pareto set (middle column), and the double Rastrigin function $F_{46} = (f_{15}, f_{15})$ for which both Pareto set and Pareto front are discontinuous (right column). The best known Pareto set/front approximations are shown in black.

We provide four different plots for each function: the top row of Figure 3 shows the projection onto an axes-parallel cut of the search space defined by two variables (x_1 and x_4 here). The second row shows the projection onto a random plane that contains both single-objective optima[14] and gives contour lines for each objective function. The coordinate system is chosen such that the two optima lie on the first axis and their geometric center is the origin. The third row depicts the same data in the objective space in original scaling (as seen by the algorithm), while the last row shows the same in log-scale, normalized so that the ideal point is at $(0, 0)$ and the nadir point is at $(1, 1)$.

In addition to the best known Pareto set/Pareto front approximations (in black), the plots of Figure 3 show various 1-dimensional cuts ("lines") through the search space: (i) along a random direction through each single-objective optimum (in blue), (ii) along each coordinate axis through each single-objective optimum (blue dotted lines), (iii) along the line connecting both single-objective optima (in red), (iv) two fully random lines[15] (in yellow), and (v) a random line in the random projection plane going through both optima[16] (in green). All those straight lines have the same length of 20 with the support vector in its exact middle. Ticks along the lines indicate line segments of the same length in search space. Thicker points on the lines depict solutions that are non-dominated with respect to all points on the same line.

The shape of these lines in objective space and the distribution of non-dominated solutions along these lines (in objective and search space) indicate the difficulty of the `bbob-biobj-ext` problems for algorithms based on line searches. Sometimes smooth, sometimes rather chaotic looking lines in the objective space indicate that the proposed multiobjective problems show a wide range of different landscapes.

Furthermore, the first two rows of Figure 3 highlight the projected region $[-5, 5]^n$ as a gray-shaded area while the gray-shaded areas in the objective space plots in the last two rows denote the regions of interest between the ideal ($\times$) and nadir points ($+$).

[13]The non-dominated solutions have been obtained from extensive numerical experiments with a wide range of optimization algorithms. In particular, all 32 algorithm data sets[2], submitted to the BBOB workshops in the years 2016–2019, contributed together with data from additional runs of random search, weighted sum and Chebyshev scalarization functions optimized by the quasi-Newton BFGS method and by CMA-ES (Hansen and Ostermeier, 2001). Moreover, other well-known multiobjective algorithms such as NSGA-II (Deb et al., 2002), SMS-EMOA (Beume et al., 2007), HMO-CMA-ES (Loshchilov and Glasmachers, 2016), and COMO-CMA-ES (Touré et al., 2019) have been run as well. Non-dominated solutions evaluated during all these experiments have been collected to form the displayed Pareto set approximations. Those Pareto set approximations have also been used to calculate reference target precision values for the performance assessment—for details, see (Brockhoff et al., 2016).

[14]To obtain the second axis, a standard normally distributed vector is orthogonalized with respect to the first axis (via Gram-Schmidt).

[15]of random direction through a random point drawn uniformly in $[-4, 4]^n$

[16]with a random direction within the plane and through a random point drawn uniformly in $[-4, 4]^2$

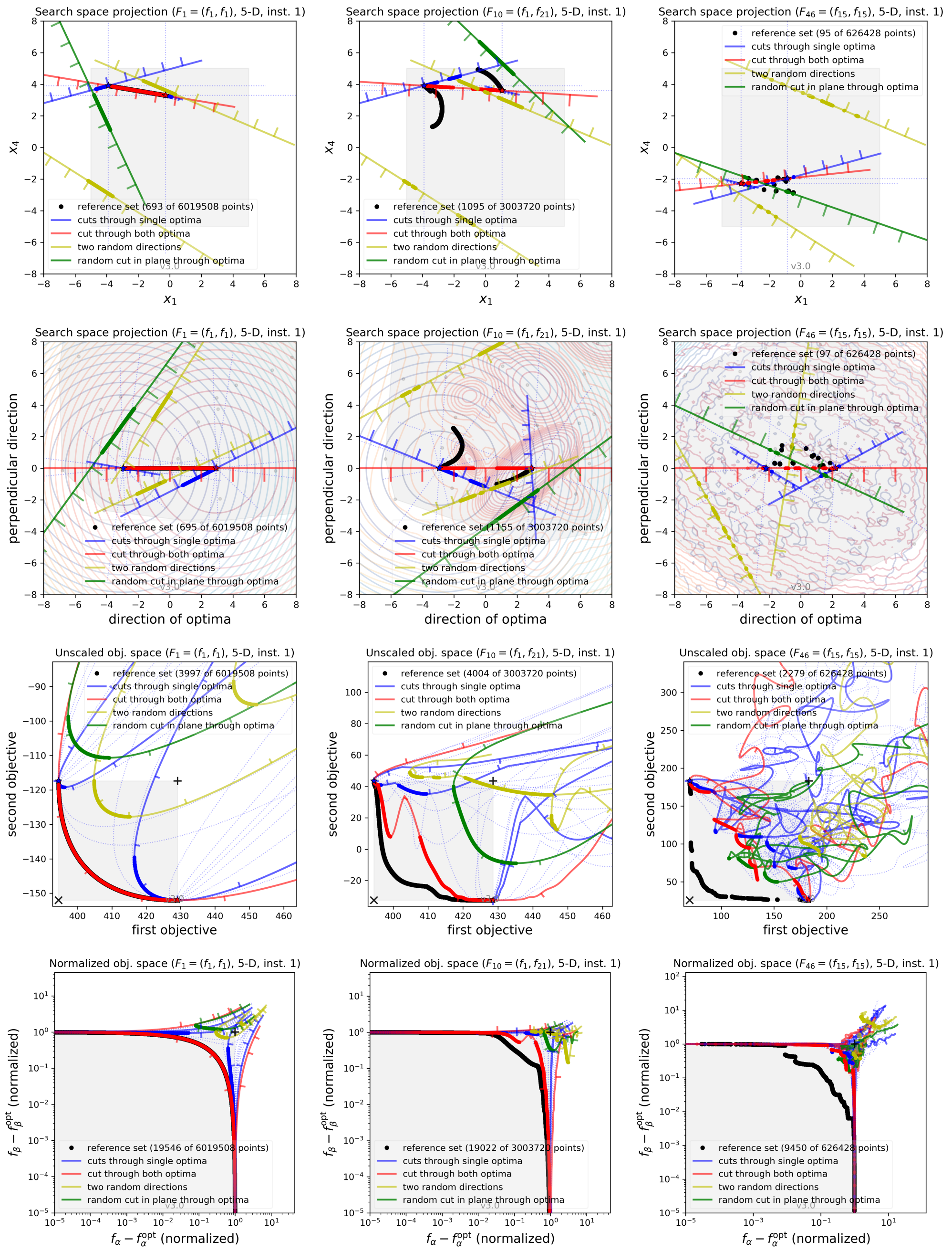

Figure 3: Illustration in the search space (first two rows) and the objective space (third row: original scaling; forth row: normalized in log-scale) of three `bbob-biobj` functions: the double sphere function $F_1 = (f_1, f_1)$ (left column), the sphere/Gallagher 101 peaks function $F_{10} = (f_1, f_{21})$ (middle column), and the double Rastrigin function $F_{46} = (f_{15}, f_{15})$ (right column) in dimension 5 for the first instance.

Pareto set and Pareto front approximations of Figure 3 are downsampled such that only one solution per grid point is shown—with the precision of 2 decimals for the search space plots and 3 decimals for the objective space plots to define the grid. The number of all available and actually displayed solutions is indicated in the legend of each plot. Due to this downsampling, the number of displayed points (~ 20000 or less) in Figure 3 is much smaller than the number of non-dominated solutions contained in the Pareto set approximation (about 6×10^6 for F_1, 3×10^6 for F_{10} and 6×10^5 for F_{46}). For links to the illustrations of *all* functions, we refer to the supplementary material[5].

6.2 Investigations on Necessary Optimality Conditions

An important question to ask is whether our collected Pareto set approximations are qualitatively good, i.e., how close they are to the true Pareto front. Assuming differentiable objective functions, a necessary condition for optimality of a point x is that the two objective gradients are collinear and point in opposite directions (Miettinen, 1999).

For our blackbox functions, we estimate collinearity as follows: for each Pareto set approximation (or function instance), we assess the two objective function gradients of solutions by finite (central) differences[17] and compute the angle between them. The angle between the gradients is closely related to the *normalized multiobjective gradient* by Kerschke and Grimme (2017), in the biobjective case defined as

$$\frac{\nabla f_1(x)}{\|\nabla f_1(x)\|} + \frac{\nabla f_2(x)}{\|\nabla f_2(x)\|} \quad . \tag{1}$$

Its value is zero for locally non-dominated solutions (Kerschke and Grimme, 2017). The cosine of the angle between the two gradients has a strictly increasing bijection to the length of the normalized biobjective gradient (1):

Lemma 1. *The length of the normalized biobjective gradient* (1) *and the cosine of the angle between the two gradients, say v and w, are related like*

$$1 + \cos(v, w) = \frac{1}{2} \left\| \frac{v}{\|v\|} + \frac{w}{\|w\|} \right\|^2 \qquad \forall\, v, w \in \mathbb{R}^n \setminus \{0\} \ . \tag{2}$$

The proof is given in Appendix B. A value of 0, 1, or 2 for $1 + \cos(v, w)$ in (2) means that the gradients point in the exact opposite direction (indicating local non-dominance), are orthogonal, or point in the very same direction, respectively.

Figure 4 shows empirical cumulative distribution functions (ECDFs) of the value $1 + \cos(\alpha)$, where α is the angle between the two gradients, denoted as *gradient angle plots* in the following. Shown are the first five instances (using the same color) of three different `bbob-biobj` functions in dimensions 2, 3, and 5.[18] For the gradient angle plots of *all* functions, we refer to the supplementary material[5].

Depending on the function, two different observations can be made. For simpler, smooth functions such as the double sphere function $F_1 = (f_1, f_1)$, almost all values of $1 + \cos(\alpha)$ are smaller than 10^{-4} for all instances and dimensions—indicating that

[17]Using a step length of $\varepsilon = 10^{-8}$ to approximate the i-th coordinate of the gradient in $x \in \mathbb{R}^n$ by $\frac{f(x+\varepsilon\cdot e_i)-f(x-\varepsilon\cdot e_i)}{2\varepsilon}$ where $e_i \in \mathbb{R}^n$ is the Cartesian unit vector in the ith variable.

[18]Only every hundredth point of the Pareto set approximations is displayed to limit computation time and file sizes. We checked for some cases that the display of all solutions does not change the plots significantly. Reducing the number of displayed points further, for example to every thousandth point, however, starts to visibly roughen the plots. The legends in Figure 4 show the number of displayed points compared to the entire size of the Pareto set approximation for each dimension and for the last instance.

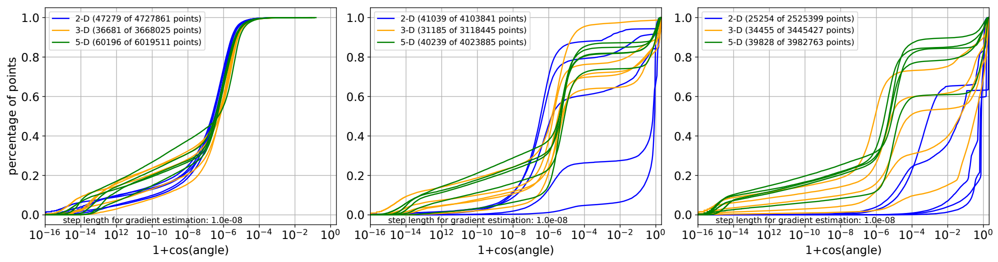

Figure 4: Gradient angle plots of solutions in our Pareto set approximations for three functions. A necessary optimality condition for smooth functions is $1 + \cos(\alpha) = 0$. Left: the double sphere function $F_1 = (f_1, f_1)$, Middle: the sphere/sharp ridge function $F_5 = (f_1, f_{13})$, Right: the original Rosenbrock/sharp ridge function $F_{29} = (f_8, f_{13})$.

the gradients point in nearly opposite directions. For smooth functions, this also suggests that our Pareto set approximations are reasonably close to a true local Pareto set. This also holds for other combinations of objectives such as the sphere/sum of different powers function $F_6 = (f_1, f_{14})$, the sphere/Schwefel function $F_9 = (f_1, f_{20})$, the attractive sector/sum of different powers function $F_{23} = (f_6, f_{14})$, the double sum of different powers function $F_{41} = (f_{14}, f_{14})$, and the double Schwefel $x \sin x$ function $F_{53} = (f_{20}, f_{20})$ and with slightly higher cosine values for several other functions, for example the attractive sector/Gallagher 101 peaks function $F_{27} = (f_6, f_{21})$ and the double original Rosenbrock function $F_{28} = (f_8, f_8)$. Both lists are non-exhaustive[5].

On the other hand, there are functions with higher values of $1 + \cos(\alpha)$, indicating (at first sight) that the Pareto set approximations are less well converged. Two examples, the sphere/sharp ridge function $F_5 = (f_1, f_{13})$ and the original Rosenbrock/sharp ridge function $F_{29} = (f_8, f_{13})$ are shown in Figure 4. A closer look at the functions with the (strongest) right-shift in the ECDF reveals that often one of the objectives is multimodal, flat, or its gradient changes non-smoothly (which is the case for the sharp ridge function of Figure 4). In the latter case in particular, we do not expect that the values of $1 + \cos(\alpha)$ are close to zero. We have verified that for the combination of sphere and sharp ridge, all solutions in our Pareto set approximation lie along a line between the two single-objective optima, but that depending on which side of the sharp ridge a point is located, the two objectives' gradients are pointing in the same or the opposite direction, resulting in small and large values of $1 + \cos(\alpha)$ respectively.

Although we do not know the exact Pareto set for the proposed functions (except for the double sphere function $F_1 = (f_1, f_1)$), the generated non-dominated Pareto set approximations seem to be accurate enough also based on consistency: both, the large amount of non-dominated solutions from several algorithms, implemented by various authors as presented here, and the non-dominated solutions on a 2-dimensional grid, as presented in the next subsection, show qualitatively the same sets[19].

6.3 Dominance-Based Plots

To further investigate and understand the `bbob-biobj-ext` functions and their available Pareto set approximations, we provide three kinds of dominance-based plots for 2-variable problems computed on an axes-parallel grid of 501×501 points. Figure 5 shows examples of these plot for the first instance of three `bbob-biobj` functions

[19]In the cases where our Pareto set approximations go beyond the region $[-5, 5]^n$, the grid-based approach shows additional artifacts for solutions connected to the grid boundary.

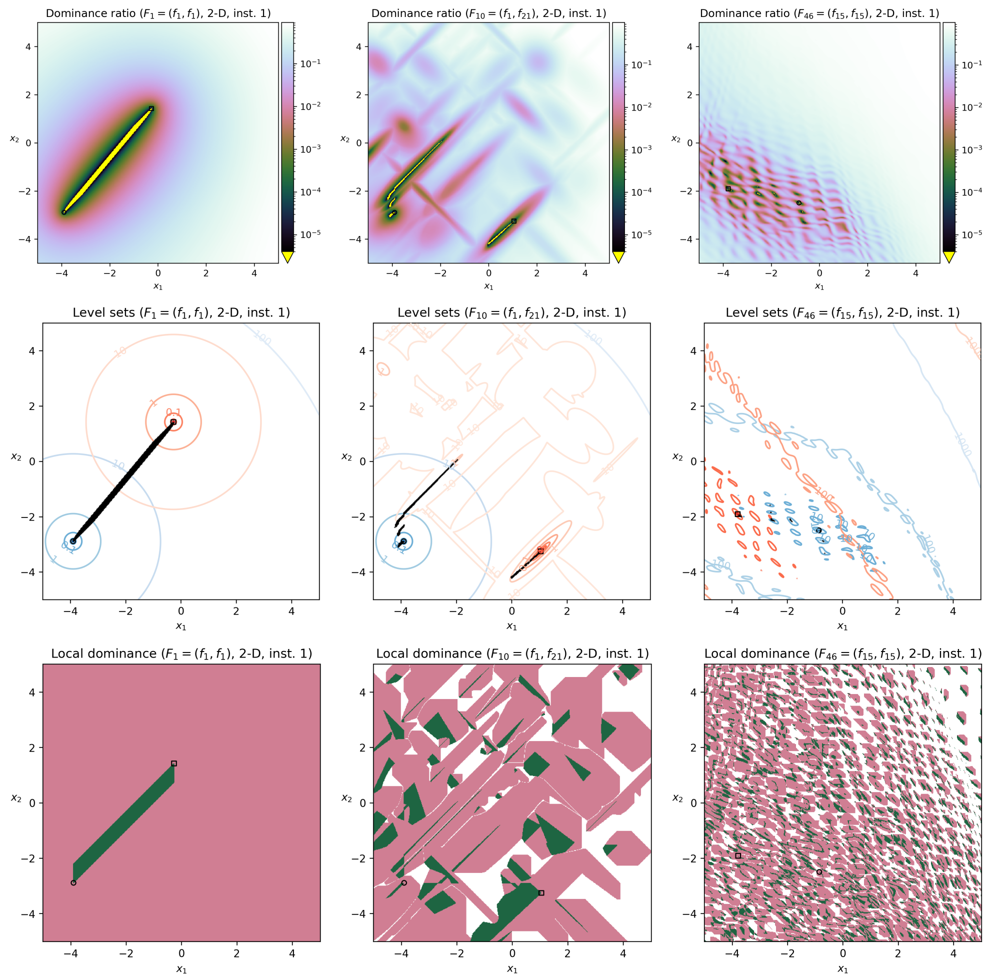

Figure 5: Dominance-based plots for three `bbob-biobj` functions: the double sphere function $F_1 = (f_1, f_1)$ (left column), the sphere/Gallagher 101 peaks function $F_{10} = (f_1, f_{21})$ (middle column), and the double Rastrigin function $F_{46} = (f_{15}, f_{15})$ (right column) in dimension 2 for the first instance. First row: dominance ratio; Second row: non-dominated solutions (in black) and level sets of both objectives; Third row: local dominance landscape plots.

(the double sphere function $F_1 = (f_1, f_1)$, the sphere/Gallagher 101 peaks function $F_{10} = (f_1, f_{21})$ and the double Rastrigin function $F_{46} = (f_{15}, f_{15})$), while the plots for the other functions and instances can be found in the supplementary material[5].

The first row of Figure 5 presents plots of the dominance ratio (Fonseca, 1995, p. 71ff.). Based on solutions on a regular grid in search space, the dominance ratio of each grid point is the ratio of all grid points that dominate it. The plot uses a logarithmic color scale with the overall non-dominated points shown in yellow.

The non-dominated grid points are presented in the second row of Figure 5 in black, together with the level sets of the first objective in blue and the second in red.

The last row of Figure 5 shows the so-called local dominance landscape plots by

Fieldsend et al. (2019). These plots use, for each grid point, one of three different colors. In dark green, we show all grid points P for which *all* 8 neighboring grid points (in the Moore neighborhood) are mutually non-dominated to P. In landscape analysis terms, we can call the green regions in the plots "dominance-neutral local optima regions" (Fieldsend et al., 2019) in the sense that a dominance-based hill-climber will be able to explore a green region by single non-dominated moves. The pink areas are comprised of grid points for which *at least one* neighboring grid point dominates it and all dominating movement paths from neighbors in pink regions lead to the same green region (Fieldsend et al., 2019). Each pink area is considered a "basin of attraction" of a green area in the sense that a local dominance-based hill climber can only move towards the included green area. Last, a grid point is assigned a white color if at least two of its dominating neighbors belong to two different basins of attraction—white areas in the plots therefore show the boundaries between the basins. The existence of multiple, distinct basins of attractions can be interpreted as multimodality of a multiobjective problem: it is denoted unimodal if and only if a single basin of attraction is present (and thus no two distinct local Pareto-optimal sets exist). Otherwise, the problem is called multimodal.

6.4 Gradient-Based Plots

The gradient angle plots of Section 6.2 gave insight into the angle between the gradients of solutions contained in the best known Pareto set approximations. We now present biobjective gradient plots based on 2-dimensional grids.

The first row in Figure 6 visualizes the length of the normalized biobjective gradient (1) at 501×501 grid points for three `bbob-biobj` functions. This length signifies how aligned or opposed the single-objective gradients are. For *all* functions, we again refer to the supplementary material[5].

The second row in Figure 6 shows the length of the path from each grid point towards the next local optimum. Inspired by the cumulated gradient field landscapes (Kerschke and Grimme, 2017), the length is (recursively) defined as Euclidean distance to the Moore neighbor to which the biobjective gradient points[20] plus the path length of this neighbor[21]. Cumulated gradient field landscapes sum gradient lengths instead of Euclidean distances. We use the latter because they better quantify the actual distance to the local optimum, however both approaches lead to qualitatively very similar figures. Plotting the lengths of the normalized grid-based local search paths as in Figure 6 allows to visualize the difficulty of multiobjective landscapes, at least for problems with 2 variables: although not as clearly as in the previous dominance-based plots we can infer the boundaries of the basins of attractions of the local Pareto-optimal sets as the places in the search space where the path lengths for neighbored grid points decrease in two different directions, see the examples in Figure 6. With the normalized biobjective gradient plots, we can therefore see how many distinct local Pareto-optimal sets exist.

6.5 Summary of Observed Problem Properties

From the above visualizations we report basic properties of the `bbob-biobj-ext` test functions and the corresponding Pareto set/front approximations that we provide for performance assessment. We investigate properties like continuous vs. discontinuous Pareto fronts and sets, the existence of non-dominated points outside a certain region,

[20]The 360 degrees of possible directions are assigned evenly to the eight neighbors.

[21]A grid point whose normalized biobjective gradient is below 10^{-6} in length is considered to be at a (local) Pareto-optimum and assigned zero path length.

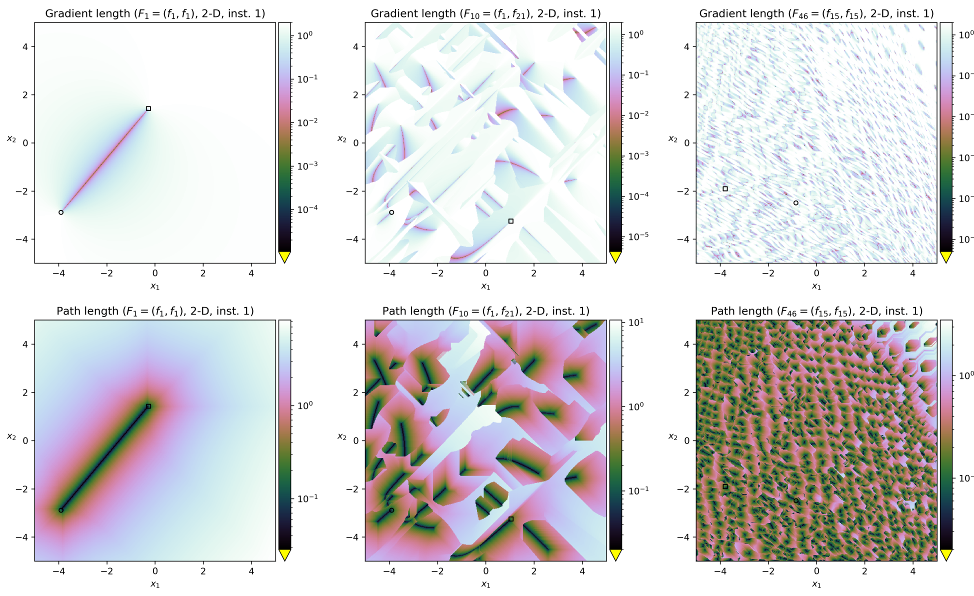

Figure 6: Gradient-based plots for three `bbob-biobj` functions: the double sphere function $F_1 = (f_1, f_1)$ (left column), the sphere/Gallagher 101 peaks function $F_{10} = (f_1, f_{21})$ (middle column), and the double Rastrigin function $F_{46} = (f_{15}, f_{15})$ (right column) in dimension 2 for the first instance. First row: pure gradient length at each grid point, Second row: length of the path from each grid point towards the next local optimum (see text for more information).

and uni- or multimodality. Table 1 details these findings for the 2-variable instances 1–5 where search space related numbers correspond to the region $[-5, 5]^n$.

We briefly summarize main findings from Table 1. First, different instances of the same function share most of the time the same properties—especially for the number of basins of attraction and the Pareto front convexity. Similarly, 78 of the 92 `bbob-biobj-ext` functions do not differ among the instances 1–5 in terms of Pareto set/front (dis-)continuity—only the number of the connected Pareto set/front parts changes in case discontinuities are observed.

Out of the 92 `bbob-biobj-ext` functions, only 27 (of which 18 belong to the 55 `bbob-biobj` suite) have a continuous Pareto front and a continuous Pareto set in all five instances (i.e. 29% resp. 33% of the functions). Only 9 functions show a convex Pareto front for all five instances. Similarly, all five instances of 59 functions are multimodal, as indicated by more than one basin of attraction. Hence we conclude that most of the proposed functions possess challenging Pareto set and Pareto front shapes.

Furthermore, the Pareto set of most problem instances of the proposed test suites lies within the hyperbox $[-5, 5]^n$. However, 51 of the investigated $92 \cdot 5 = 460$ instances (from 28 functions) show non-dominated points outside. For this reason, the COCO implementation defines the region of interest as $[-100, 100]^n$ although the single-objective optima are always in $[-4, 4]^n$ (with the exception of the linear function f_5).

Table 1: Selected properties of the `bbob-biobj-ext` test functions for the first five instances in dimension 2. The columns give the properties, from left to right, the number of distinct Pareto set parts (?=unclear), the number of distinct Pareto front parts, the (visual) convexity of the Pareto front (n=non-convex, y=convex), the existence of solutions within the best known Pareto set approximation that lie outside $[-5, 5]^n$ (n=no such point, y=some points outside of the hyperbox), and finally the number of basins of attraction within $[-5, 5]^n$, induced visually from the plots described in the text.

	# Pareto set subsets					# Pareto front subsets					Convex Pareto front					Pareto set with $\|x_i\| > 5$					# Basins of attraction				
Instance	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
F1=(f1, f1)	1	1	1	1	1	1	1	1	1	1	y	y	y	y	y	n	n	n	n	n	1	1	1	1	1
F2=(f1, f2)	1	1	1	1	1	1	1	1	1	1	y	n	y	n	y	n	n	n	n	n	1	1	1	1	1
F3=(f1, f6)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	n	n	n	n	n	1	1	1	1	1
F4=(f1, f8)	1	1	1	1	1	1	1	1	1	1	y	y	y	y	y	n	n	n	n	n	1	1	1	1	1
F5=(f1, f13)	1	1	1	1	1	1	1	1	1	1	y	y	y	y	y	n	n	n	n	n	1	1	1	1	1
F6=(f1, f14)	1	1	1	1	1	1	1	1	1	1	y	y	y	y	y	n	n	n	n	n	1	1	1	1	1
F7=(f1, f15)	4	10+	5–9	10+	10+	3	10+	5–9	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F8=(f1, f17)	5–9	5–9	5–9	5–9	5–9	5–9	5–9	5–9	5–9	5–9	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F9=(f1, f20)	1	5–9	5–9	1	1	1	5–9	5–9	1	1	y	y	y	y	y	n	n	n	n	n	5–9	5–9	5–9	5–9	5–9
F10=(f1, f21)	3	2	2	5–9	1	2	1	1	3	1	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F11=(f2, f2)	?	?	?	?	10+	1	1	1	1	1	n	n	n	n	n	n	n	n	n	n	1	1	1	1	1
F12=(f2, f6)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	y	n	n	y	y	1	1	1	1	1
F13=(f2, f8)	1	2	1	1	1	1	1	1	1	1	y	y	y	n	n	n	n	n	n	n	3	3	3	1	1
F14=(f2, f13)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	n	n	y	y	n	1	1	1	1	1
F15=(f2, f14)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	y	n	n	y	n	1	1	1	1	1
F16=(f2, f15)	10+	10+	5–9	10+	10+	10+	10+	5–9	5–9	10+	n	n	n	n	n	y	y	n	n	y	10+	10+	10+	10+	10+
F17=(f2, f17)	10+	10+	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F18=(f2, f20)	3	3	3	3	3	3	3	1	3	3	n	n	n	n	n	y	y	n	n	n	10+	10+	10+	10+	10+
F19=(f2, f21)	3	3	2	3	2	2	2	2	1	2	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F20=(f6, f6)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	y	n	n	n	n	1	1	1	1	1
F21=(f6, f8)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	n	n	n	n	n	2	2	1	2	2
F22=(f6, f13)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	n	n	n	n	n	1	1	1	1	1
F23=(f6, f14)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	n	n	y	y	n	1	1	1	1	1
F24=(f6, f15)	10+	10+	5–9	10+	5–9	5–9	3	4	10+	5–9	n	n	n	n	n	n	n	n	y	n	10+	10+	10+	10+	10+
F25=(f6, f17)	10+	10+	10+	10+	10+	10+	10+	5–9	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F26=(f6, f20)	5–9	4	5–9	3	4	5–9	4	2	2	3	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F27=(f6, f21)	2	4	3	3	5–9	2	2	2	2	2	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F28=(f8, f7)	1	1	1	1	1	1	1	1	1	1	y	y	y	y	y	n	n	n	n	n	2	2	2	2	2
F29=(f8, f13)	1	1	2	2	1	1	1	1	1	1	y	y	n	y	y	n	n	y	n	n	2	2	2	2	1
F30=(f8, f14)	1	1	1	1	1	1	1	1	1	1	y	y	y	y	y	n	n	n	n	n	1	2	2	1	1
F31=(f8, f15)	10+	5–9	5–9	10+	10+	5–9	3	4	10+	5–9	n	n	n	n	n	n	n	n	y	n	10+	10+	10+	10+	10+
F32=(f8, f17)	10+	10+	10+	10+	10+	5–9	10+	10+	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F33=(f8, f20)	3	4	5–9	5–9	4	2	3	5–9	3	3	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F34=(f8, f21)	3	3	5–9	5–9	5–9	3	1	1	3	2	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F35=(f13, f13)	1	1	1	1	1	1	1	1	1	1	y	n	y	n	n	n	n	n	n	n	1	1	1	1	1
F36=(f13, f14)	1	1	1	1	1	1	1	1	1	1	y	y	y	y	y	n	n	n	y	n	1	1	1	1	1
F37=(f13, f15)	10+	10+	10+	10+	5–9	10+	10+	10+	10+	4	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F38=(f13, f17)	10+	10+	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F39=(f13, f20)	3	3	3	2	5	3	2	3	2	5	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F40=(f13, f21)	4	2	2	2	4	3	2	2	2	2	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F41=(f14, f14)	1	1	1	1	1	1	1	1	1	1	y	y	y	y	y	n	n	n	n	n	1	1	1	1	1
F42=(f14, f15)	10+	5	10+	10+	10+	10+	5	10+	10+	10+	n	n	n	n	n	y	n	n	n	n	10+	10+	10+	10+	10+
F43=(f14, f17)	10+	10+	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F44=(f14, f20)	5–9	4	4	3	3	5–9	4	4	3	3	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F45=(f14, f21)	5	4	4	5	4	3	3	1	3	2	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F46=(f15, f15)	5–9	5–9	10+	10+	10+	3	5–9	5–9	5–9	5–9	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F47=(f15, f17)	10+	10+	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	y	n	n	n	y	10+	10+	10+	10+	10+
F48=(f15, f20)	2	4	10+	5–9	4	2	3	10+	5–9	3	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F49=(f15, f21)	4	5–9	10+	10+	3	3	3	5–9	10+	3	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F50=(f17, f17)	5–9	10+	5–9	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F51=(f17, f20)	10+	10+	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	n	y	n	n	n	10+	10+	10+	10+	10+
F52=(f17, f21)	10+	10+	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F53=(f20, f20)	5	5–9	5–9	5	5	5	5	5	5	3	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F54=(f20, f21)	5–9	5–9	2	5–9	4	5–9	5–9	2	3	2	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F55=(f21, f21)	3	5–9	5–9	5–9	5–9	1	1	2	1	2	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F56=(f1, f3)	10+	5–9	5–9	10+	10+	5–9	5–9	5–9	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F57=(f1, f4)	10+	10+	5–9	10+	10+	5–9	5–9	5–9	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F58=(f1, f5)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	n	n	n	n	n	1	1	1	1	1
F59=(f2, f3)	10+	10+	5–9	10+	10+	10+	5–9	5–9	5–9	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F60=(f2, f4)	10+	10+	10+	5–9	10+	10+	5–9	5–9	5–9	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F61=(f2, f5)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	y	y	y	y	y	1	1	1	1	1
F62=(f3, f4)	10+	10+	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F63=(f3, f5)	10+	10+	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	n	y	y	n	n	10+	10+	10+	10+	10+
F64=(f4, f5)	10+	10+	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F65=(f6, f7)	10+	10+	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	n	n	y	y	n	10+	10+	10+	10+	10+
F66=(f6, f9)	1	3	1	3	1	1	1	1	1	1	n	n	n	n	n	n	n	n	y	n	1	3	1	3	2
F67=(f7, f8)	10+	10+	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F68=(f7, f9)	10+	10+	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F69=(f8, f9)	1	1	1	2	2	1	1	1	1	1	n	y	y	n	n	n	n	n	n	n	1	1	1	2	2
F70=(f10, f11)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	n	n	n	n	n	1	1	1	1	1
F71=(f10, f12)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	n	n	n	n	n	1	1	1	1	1
F72=(f10, f13)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	n	n	y	n	n	1	1	1	1	1
F73=(f10, f14)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	n	n	n	n	n	1	1	1	1	1
F74=(f11, f12)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	n	n	y	n	y	1	1	1	1	1
F75=(f11, f13)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	n	n	y	n	y	1	1	1	1	1
F76=(f11, f14)	1	1	1	1	1	1	1	1	1	1	n	n	n	n	n	n	n	n	n	n	1	1	1	1	1
F77=(f12, f13)	1	1	2	1	1	1	1	1	1	1	n	n	n	n	n	n	y	y	n	n	1	1	1	1	1
F78=(f12, f14)	2	2	1	2	1	1	1	1	1	1	y	y	y	n	y	y	y	n	y	n	1	1	1	1	1
F79=(f15, f18)	10+	5–9	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	y	n	y	y	n	10+	10+	10+	10+	10+
F80=(f15, f19)	10+	5–9	10+	4	5–9	10+	5–9	10+	4	5–9	n	n	n	n	n	n	n	y	n	n	10+	10+	10+	10+	10+
F81=(f17, f18)	5–9	10+	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	n	n	y	n	y	10+	10+	10+	10+	10+
F82=(f17, f19)	10+	10+	5–9	10+	10+	10+	10+	5–9	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F83=(f18, f19)	10+	10+	5–9	10+	10+	10+	10+	5–9	10+	10+	n	n	n	n	n	n	y	n	n	n	10+	10+	10+	10+	10+
F84=(f20, f22)	3	2	5	5	4	3	2	3	5	2	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F85=(f20, f23)	5–9	5–9	5–9	5–9	5–9	5–9	5–9	5–9	5–9	5–9	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F86=(f20, f24)	4	10+	10+	10+	5–9	5–9	10+	10+	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F87=(f21, f22)	5–9	5–9	5–9	10+	3	3	1	2	2	1	n	n	n	n	n	n	n	y	n	n	10+	10+	10+	10+	10+
F88=(f21, f23)	10+	5–9	5–9	5–9	5–9	5–9	5–9	5–9	5–9	3	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F89=(f21, f24)	10+	10+	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F90=(f22, f23)	3	5–9	2	2	3	5–9	5–9	3	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F91=(f22, f24)	10+	10+	10+	10+	10+	10+	10+	10+	10+	10+	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+
F92=(f23, f24)	5–9	5–9	5–9	3	5–9	5–9	5–9	5–9	3	5–9	n	n	n	n	n	n	n	n	n	n	10+	10+	10+	10+	10+

7 On Reporting Algorithm Performance

The definition of (test) problem instances is only the first step towards benchmarking optimization algorithms. Further steps entail to choose, record and display the actual performance measure(s). As the proposed test problems have been implemented in the COCO software, the postprocessing module of COCO can be directly used for the performance assessment (up to LaTeX templates for publication of the results) and we briefly discuss what this comprises. As best practice, we usually conduct seven steps to analyze the performance of an algorithm, here also illustrated with an example.

Benchmarking Experiment First, we run the algorithm on a chosen benchmark suite. COCO offers example code in all supported languages. A minimal code example in Python is shown in Hansen et al. (2021, Figure 2). Here, we have run the COMO-CMA-ES (Touré et al., 2019) with population size 100 on the `bbob-biobj` suite.

dimo: prepare all algorithm data sets again wrt new hypervolume values do a release with new hv reference values before submission

Choosing Algorithms for Comparison We decide on (baseline) algorithms to compare with. As of May 2021, data for browsing[3] and downloading from 32 algorithms[2] run on the `bbob-biobj` test suite is available online. Here, we choose SMS-EMOA and NSGA-II as baselines.

Postprocessing The postprocessing of COCO, here invoked by the command "`python -m cocopp COMO-100 NSGA-II-Matlab SMS-EMOA-DE`", then downloads the corresponding baseline data and displays the algorithms performance in both html and LaTeX/PDF format.

For multiobjective problems, COCO measures the value of an extended hypervolume indicator of all non-dominated solutions evaluated so far[22]. Within a benchmarking experiment, COCO records the number of function evaluations ("runtime") to reach certain indicator values, given as precisions to a reference value. For each instance, function, and dimension, the runtime to 58 target precisions is displayed (amongst other visualizations) in the form of empirical cumulative distribution functions (ECDF), see Hansen et al. (2016a, 2010).[23]

Test suites that contain many function instances in several dimensions require to aggregate data when analyzing and sharing results. Conveniently, runtimes from different instances or functions can be meaningfully displayed in a single ECDF graph.

Displaying and Discussing Summary Results In publications, we show summary ECDFs over all functions in 5-D and 20-D and the results for each function group in 10-D.

[22]More specifically, the (multiobjective) performance measure used in COCO makes a case distinction. When at least one solution dominates the hypervolume reference point, the hypervolume indicator of all non-dominated solutions is used, normalized so that the ideal point is in the origin and the nadir point is in $[1, 1]$, with this nadir point as the (normalized) reference point. Otherwise, the performance is measured as the negative of the minimal distance of any found solution (in objective space and normalized as above) to the box $[0, 1]^2$, for details see (Brockhoff et al., 2016).

[23]All plots shown in this paper have been prepared with COCO version 2.4 and the corresponding hypervolume reference values. Improved hypervolume reference values as well as the `bbob-biobj-ext` suite are available from version 3.0.

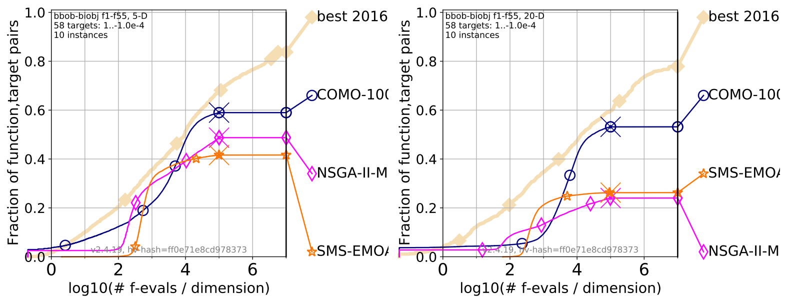

Figure 7: Empirical runtime distributions aggregated over all `bbob-biobj` functions in dimension 5 (left) and 20 (right), comparing the algorithms COMO-100, SMS-EMOA-DE, and NSGA-II-Matlab.

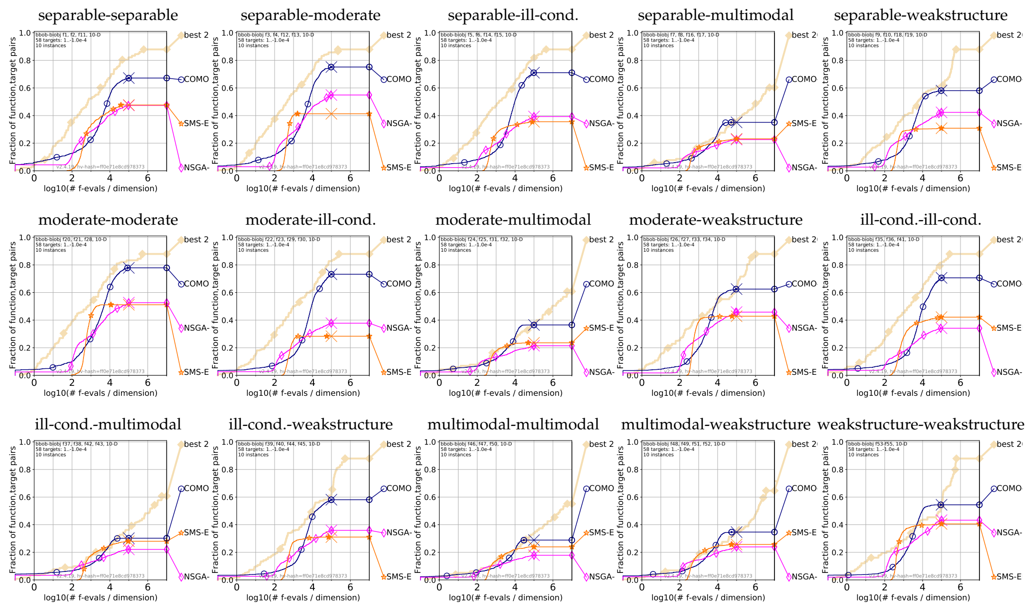

Figure 8: Empirical runtime distributions per function group of the `bbob-biobj` suite for the algorithms COMO-100, SMS-EMOA-DE, and NSGA-II-Matlab in dimension 10.

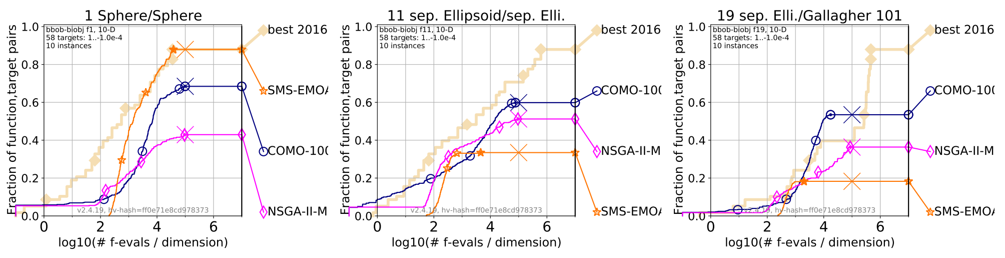

Figure 9: Empirical runtime distributions for the algorithms COMO-100, SMS-EMOA-DE and NSGA-II-Matlab on single functions from the `bbob-biobj` suite in dimension 10. Left: double sphere function $F_1 = (f_1, f_1)$, middle: double ellipsoid function $F_{11} = (f_2, f_2)$, right: ellipsoid/Gallagher 101 peaks function $F_{19} = (f_2, f_{21})$.

Figure 7 shows these runtime distribution graphs aggregated over all functions and Figure 8 shows the aggregations within function groups[24]. For budgets larger than about $5 \cdot 10^3$ times dimension evaluations, COMO-100 overall outperforms SMS-EMOA-DE and NSGA-II-Matlab, where the latter two are also more affected by increasing dimension. The qualitative differences of the runtime behaviors between the algorithms remain similar also within function groups, while differences are smaller in the groups involving multimodal objective functions.

Investigating and Discussing Complementary Results We always inspect all results on all single functions. Depending on the outcome of this inspection, we display further graphs. They may show the dependency on dimension or any other remarkable observation.

Figure 9 shows ECDF graphs for three single functions. We observe exceptionally good behavior of SMS-EMOA-DE on the double sphere function (Figure 9, left). The middle plot of Figure 9 shows an example where NSGA-II is better than SMS-EMOA-DE for all budgets. Finally, on the right plot, we observe that COMO-100 matches and exceeds the performance of the best 2016 reference algorithm on the ellipsoid/Gallagher 101 peaks function $F_{19} = (f_2, f_{21})$ for budgets between $5 \cdot 10^3$ and 10^4 times dimension evaluations but also does not improve anymore shortly after.

Processed Data Sharing We put the html output from the postprocessing online, see *Postprocessed data* in the supplementary material[5]. During scientific collaborations, we often share such data repeatedly with collaborators early in the process.

Raw Data Sharing Finally, we publish the raw data sets by creating a submission issue on GitHub[25]. Raw data can also be "archived" with the `cocopp.archiving` module. Archived data that are then put online can be used in the postprocessing (`cocopp`) by everyone who knows the corresponding URL.

8 Conclusions

Designing test suites is a crucial part of benchmarking optimization algorithms. Arguably, the most problematic aspect of using *artificial* test functions to assess performance is the *representativeness* of these regarding difficulties observed in real-world problems. In this paper, we suggest to address the problem of representativeness in the multiobjective case by combining established single-objective test functions with known difficulties observed in practice. Following the concepts of the single-objective `bbob` test suite, we propose two concrete biobjective test suites based on the idea of combining (subsets of) the existing `bbob` single-objective functions.

Our approach contrasts most of the existing test suites for multiobjective optimization. These are based on the desirable property of having well-understood Pareto sets and Pareto fronts with analytical forms but have, on the other hand, artificial characteristics that are arguably under-represented in real-world problems. Examples of such properties are separability, optima located exactly at the boundary constraints, and the existence of variables that solely control the distance between a solution and the Pareto front.

[24]Displayed is the performance of the three algorithms COMO-CMA-ES with a population size of 100 ("COMO-100", Touré et al. (2019); Dufossé and Touré (2019)), SMS-EMOA with differential evolution as search operator ("SMS-EMOA-DE", Beume et al. (2007); Auger et al. (2016b)) and the Matlab implementation of NSGA-II ("NSGA-II-Matlab", Deb et al. (2002); Auger et al. (2016a)).

[25]See `https://github.com/numbbo/coco/blob/master/howtos/publish-a-dataset-howto.md`

The disadvantage of unknown analytical forms of the Pareto sets and Pareto fronts in our proposal is addressed by collecting the non-dominated solutions from extensive experiments with dozens of different optimization algorithms and providing and visualizing the Pareto set and Pareto front approximations for each problem. These visualizations lead to new insights into how such non-analytical Pareto sets and Pareto fronts may look in practice.

Our proposal is currently restricted to two objectives. With a growing number of objectives, the number of arbitrary combinations of single-objective functions grows quickly. This leads to long running times of the benchmarking experiment and, more importantly, discourages to routinely scrutinize new results on each function individually. Hence, we think that further pruning choices should be made to define test suites with more objectives.

Acknowledgments

This work was supported by the grant ANR-12-MONU-0009 (NumBBO) of the French National Research Agency. We also thank Ilya Loshchilov and Oswin Krause for their initial suggestions on how to extend the `bbob-biobj` test suite. Tea Tušar acknowledges financial support from the Slovenian Research Agency (research project No. Z2-8177 and research program No. P2-0209) and the European Commission's Horizon 2020 research and innovation program (grant agreement No. 692286).

References

Auger, A., Brockhoff, D., Hansen, N., Tušar, D., Tušar, T., and Wagner, T. (2016a). Benchmarking MATLAB's gamultiobj (NSGA-II) on the bi-objective BBOB-2016 test suite. In *Genetic and Evolutionary Computation Conference (Companion)*, GECCO 2016 Companion, pages 1233–1239. ACM.

Auger, A., Brockhoff, D., Hansen, N., Tušar, D., Tušar, T., and Wagner, T. (2016b). The impact of variation operators on the performance of SMS-EMOA on the bi-objective BBOB-2016 test suite. In *Genetic and Evolutionary Computation Conference (Companion)*, GECCO 2016 Companion, pages 1225–1232. ACM.

Beume, N., Naujoks, B., and Emmerich, M. (2007). SMS-EMOA: Multiobjective Selection Based on Dominated Hypervolume. *European Journal of Operational Research*, 181(3):1653–1669.

Brockhoff, D., Tran, T.-D., and Hansen, N. (2015). Benchmarking Numerical Multiobjective Optimizers Revisited. In *Genetic and Evolutionary Computation Conference (GECCO 2015)*, pages 639–646. ACM.

Brockhoff, D., Tušar, T., Tušar, D., Wagner, T., Hansen, N., and Auger, A. (2016). Biobjective Performance Assessment with the COCO Platform. *CoRR*, abs/1605.01746.

Cheng, R., Li, M., Tian, Y., Zhang, X., Yang, S., Jin, Y., and Yao, X. (2017). Benchmark functions for the CEC'2017 competition on many-objective optimization. Technical report, University of Birmingham, UK.

Collange, G., Delattre, N., Hansen, N., Quinquis, I., and Schoenauer, M. (2010). Multidisciplinary Optimisation in the Design of Future Space Launchers. In *Multidisciplinary Design Optimization in Computational Mechanics*, pages 487–496. Wiley.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, UK.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2005). Scalable Test Problems for Evolutionary Multi-Objective Optimization. In Abraham, A., Jain, R., and Goldberg, R., editors, *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, chapter 6, pages 105–145. Springer.

Dufossé, P. and Touré, C. (2019). Benchmarking MO-CMA-ES and COMO-CMA-ES on the bi-objective bbob-biobj testbed. In *Genetic and Evolutionary Computation Conference (Companion)*, GECCO 2019, pages 1920–1927. ACM.

Emmerich, M. T. and Deutz, A. H. (2007). Test problems based on Lamé superspheres. In *Evolutionary Multi-Criterion Optimization (EMO 2007)*, pages 922–936. Springer.

Fieldsend, J. E., Chugh, T., Allmendinger, R., and Miettinen, K. (2019). A feature rich distance-based many-objective visualisable test problem generator. In *Genetic and Evolutionary Computation Conference (GECCO 2019)*, pages 541–549.

Finck, S., Hansen, N., Ros, R., and Auger, A. (2009). Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE. Updated version as of February 2019.

Fonseca, C. M. (1995). *Multiobjective genetic algorithms with application to control engineering problems*. PhD thesis, University of Sheffield.

Fonseca, C. M. and Fleming, P. J. (1995). An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16.

Gould, N. I. M., Orban, D., and Toint, P. L. (2005). CUTEr and SifDec: A Constrained and Unconstrained Testing Environment, revisited. *ACM Transactions on Mathematical Software*, 29(4):373–394.

Hansen, N., Auger, A., Brockhoff, D., Tušar, D., and Tušar, T. (2016a). COCO: Performance assessment. *ArXiv e-prints*, arXiv:1605.03560.

Hansen, N., Auger, A., Finck, S., and Ros, R. (2009). Real-Parameter Black-Box Optimization Benchmarking 2009: Experimental Setup. INRIA Research Report RR-6829, INRIA Saclay—Ile-de-France. updated February 2010.

Hansen, N., Auger, A., Ros, R., Finck, S., and Pošík, P. (2010). Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In *Genetic and evolutionary computation conference companion (GECCO 2010)*, pages 1689–1696, New York, NY, USA. ACM.

Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., and Brockhoff, D. (2021). COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36:114–144.

Hansen, N. and Ostermeier, A. (2001). Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195.

Hansen, N., Tušar, T., Mersmann, O., Auger, A., and Brockhoff, D. (2016b). COCO: The Experimental Procedure. *CoRR*, abs/1603.08776.

Horn, D., Wagner, T., Biermann, D., Weihs, C., and Bischl, B. (2015). Model-based multi-objective optimization: taxonomy, multi-point proposal, toolbox and benchmark. In *Evolutionary Multi-Criterion Optimization (EMO 2015)*, pages 64–78. Springer.

Huang, V. L., Qin, A. K., Deb, K., Zitzler, E., Suganthan, P. N., Liang, J. J., Preuss, M., and Huband, S. (2007). Problem Definitions for Performance Assessment of Multi-objective Optimization Algorithms. Technical report, Nanyang Technological University. Special Session on Constrained Real-Parameter Optimization.

Huband, S., Hingston, P., Barone, L., and While, L. (2006). A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506.

Igel, C., Hansen, N., and Roth, S. (2007). Covariance Matrix Adaptation for Multi-objective Optimization. *Evolutionary Computation*, 15(1):1–28.

Kerschke, P. and Grimme, C. (2017). An expedition to multimodal multi-objective optimization landscapes. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 329–343. Springer.

Kerschke, P., Wang, H., Preuss, M., Grimme, C., Deutz, A., Trautmann, H., and Emmerich, M. (2016). Towards analyzing multimodality of continuous multiobjective landscapes. In *Parallel Problem Solving from Nature (PPSN 2016)*, pages 962–972. Springer.

Kursawe, F. (1990). A Variant of Evolution Strategies for Vector Optimization. In Schwefel, H.-P. and Männer, R., editors, *Parallel Problem Solving from Nature (PPSN 1990)*, pages 193–197. Springer.

Li, H. and Zhang, Q. (2009). Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302.

Loshchilov, I. and Glasmachers, T. (2016). Anytime bi-objective optimization with a hybrid multi-objective CMA-ES (HMO-CMA-ES). In *Genetic and Evolutionary Computation Conference Companion (GECCO 2016 Companion)*, pages 1169–1176. ACM.

Miettinen, K. (1999). *Nonlinear Multiobjective Optimization*. Kluwer, Boston, MA, USA.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *International Conference on Genetic Algorithms and their Applications, 1985*, pages 93–100.

Tan, K. C., Lee, T. H., and Khor, E. F. (2002). Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons. *Artificial intelligence Review*, 17(4):251–290.

Touré, C., Hansen, N., Auger, A., and Brockhoff, D. (2019). Uncrowded hypervolume improvement: COMO-CMA-ES and the Sofomore framework. In *Genetic and Evolutionary Computation Conference (GECCO 2019)*, pages 638–646. ACM.

Van Veldhuizen, D. A. and Lamont, G. B. (1998). Multiobjective evolutionary algorithm research: A history and analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology.

Van Veldhuizen, D. A. and Lamont, G. B. (1999a). MOEA test suite generation, design & use. In *Genetic and Evolutionary Computation Conference (GECCO 1999). Workshop Program*, pages 113–114.

Van Veldhuizen, D. A. and Lamont, G. B. (1999b). Multiobjective evolutionary algorithm test suites. In *Symposium on Applied Computing*, pages 351–357. ACM.

Zhang, Q., Zhou, A., Zhao, S., Suganthan, P. N., Liu, W., and Tiwari, S. (2009). Multiobjective Optimization Test Instances for the CEC 2009 Special Session and Competition. CES 487, The School of Computer Science and Electronic Engieering, University of Essex.

Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Grunert da Fonseca, V. (2003). Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132.

A On the Single-objective bbob Functions

This section discusses properties of real-world problems and how the bbob test suite models different problem difficulties. It then gives more details about the bbob functions, their function groups and instances.

A.1 Real-World Function Properties

We present here in short the general properties of objective functions that are related to difficulties observed in real-world problems. It depends on these properties whether an optimization problem is easy or hard to solve. These properties build the basis of the five function groups described in Section 4.

A *separable* function does not have any dependencies among its variables and can therefore be optimized by applying n independent one-dimensional optimizations along each coordinate axis while keeping the other variables fixed. Difficult optimization problems are typically not separable and thus, *non-separable* optimization problems should be considered. The typical well-established technique to generate non-separable benchmark functions from separable ones is the application of a rotation matrix. That is, if $g(x)$ is a separable function with respect to x and $\mathbf{R} \in \mathbb{R}^{n\times n}$ is a rotation matrix, then $g(\mathbf{R}x)$ will generally be non-separable with respect to x.

A *unimodal* function has only one local minimum which is at the same time also its global one. A *multimodal* function has more local minima which is highly common in practical optimization problems. We consider a multimodal function to have *weak global structure* if the qualities (the f-values) of the local optima are only weakly related with their locations in search space, e.g. when neighboring optima do not generally have similar quality values.

Ill-conditioning is another typical challenge of real-parameter optimization and, besides multimodality, probably the most common one. The condition number measures, loosely speaking, how strongly the steepness of the gradient depends on the position within a level set. A small condition number, close to one, indicates a well-conditioned function with little dependency. A large condition number indicates a more difficult, ill-conditioned function with strong dependency between steepness and position. The condition number of convex quadratic functions is the ratio between largest and smallest eigenvalue of the Hessian matrix. Geometrically, these eigenvalues correspond, respectively, to the shortest and longest principal axis of the contour ellipsoids.

The bbob test suite contains ill-conditioned functions with a typical conditioning of 10^6. We believe this is a realistic requirement, while we have seen practical problems with conditioning as large as 10^{10} (Collange et al., 2010).

A.2 Balancing Problem Difficulties

It is worth noting that in several existing single-objective test suites, some of the easier properties are overrepresented. For example, in the CUTEr/CUTEst test suite (Gould et al., 2005), 202 (54%) out of the 375 functions, that are labeled as unconstrained or bound constrained, are of the "sum of squares" type, a further 58 (15%) are quadratic. Furthermore, out of the 191 problems with a fixed dimension, there are 49 (26%) that have only two variables while only 31 (16%) have a dimension larger than 10.

Such an overrepresentation is not a big problem per se, but when making statements on algorithm performance aggregated over all functions in a suite, one has to keep in mind that the performance of the better algorithms might simply come from the fact that they are tailored towards simpler problems.

With the bbob test suite, all problems are scalable in dimension and belong to a certain problem group, sharing similar difficulties. It is therefore possible to aggregate performance data only over a subset of the functions sharing the same properties. Having all problem groups of similar size also avoids problems of overfitting to certain difficulties if aggregated results are presented.

A.3 Function Instances

All bbob functions come naturally in the form of instances. That is to say, each function optimized by an algorithm takes the form:

$$f(x) = H_1 \circ \ldots \circ H_{k_1}(f_{\text{raw}}(T_1 \circ \ldots \circ T_{k_2}(x)))$$

where f_{raw} is a raw function—usually the simplest representative of the function class (like the sphere function with optimum in zero)—and where $T_i : \mathbb{R}^n \to \mathbb{R}^n$ are search space transformations and $H_i : \mathbb{R} \to \mathbb{R}$ are function value transformations that are applied to the raw function. For example search space transformations can be rotations or translations of the optimum and for example, a function-value transformation can be translating the function by a scalar. Each of those transformations applied to the raw function are actually (pseudo-)random, e.g. when applying a translation in the search space, the vector by which the search point is shifted is randomly sampled. The resulting functions can be seen as instances of a parametrized transformation.

In an abstract manner, the functions optimized are instances of a parametrized function F^θ (as introduced in Section 2); the parameter θ is instantiated (pseudo-)randomly from an integer number, the so-called instance number, as well as potentially from the function number. We refer to a function class as a set of functions $\{F^\theta : \theta \in \Theta\}$ and we often name the function class after its raw function.

Transformations that are shared by all bbob functions are shifts in the optimal function value and a pseudo-random location of the optimum. In addition, several of the non-separable functions are created by pseudo-random rotations of the search space and many of the simpler functions are made less regular by non-linear transformations in both search and objective space. See Hansen et al. (2009) for more details.

Though the potential set of instances for a given bbob function is unbounded (and can be indexed by any positive integer), numerical benchmarking experiments are typically advised on 10–15 of those instances. Default instances in the COCO implementation might change from year to year to avoid overfitting. Note also that in some cases, single instances might be more difficult/easier to solve than others. However, in general, the differences in difficulty among instances of the same bbob function are smaller than among different functions.

A.4 Normalization and Target Difficulties

All bbob functions are normalized in the sense that the given target function values/difficulties around the optimal function value are comparable over functions and instances. Functions are provided with an f-offset such that the optimal function value is, loosely speaking, a realization of a Cauchy distribution with median zero and interquartile range 200. The optimal function value is furthermore rounded to two decimal places and set to ± 1000 if its absolute value exceeds 1000 (Hansen et al., 2009). The target difficulties are computed as a set of differences to the optimal function value. The differences are equally spaced on the log scale and the same for all functions and instances. Algorithms however are not allowed to use or exploit any of this information (Hansen et al., 2016b).

B Proof of Lemma 1 (page 14)

Proof. For $v, w \in \mathbb{R}^n$, we have

$$\begin{aligned}\|v+w\|^2 = \sum_i (v_i + w_i)^2 = \sum_i (v_i^2 + 2v_iw_i + w_i^2) = \|v\|^2 + 2\sum_i v_iw_i + \|w\|^2 \\ = \|v\|^2 + \|w\|^2 + 2\langle v, w\rangle = \|v\|^2 + \|w\|^2 + 2\cos(v,w)\|v\|\|w\| \ ,\end{aligned}$$

where $\|v\|\|w\|\cos(v,w)$ is defined as zero if $\|v\|\|w\| = 0$. If both vectors are non-zero, we get

$$\begin{aligned}\left\|\frac{v}{\|v\|} + \frac{w}{\|w\|}\right\|^2 &= 1 + 1 + 2\cos(v/\|v\|, w/\|w\|) \\ &= 2(1 + \cos(v,w))\end{aligned}$$

□