COCO: The Large Scale Black-Box Optimization Benchmarking (bbob-largescale) Test Suite

Ouassim Elhara, Konstantinos Varelas, Duc Nguyen, Tea Tušar, Dimo Brockhoff, Nikolaus Hansen, Anne Auger

COCO: The Large Scale Black-Box Optimization Benchmarking (bbob-largescale) Test Suite

Ouassim Ait Elhara[1], Konstantinos Varelas[1], Duc Manh Nguyen[2], Tea Tušar[3], Dimo Brockhoff[1], Nikolaus Hansen[1], Anne Auger[1]
[1]RandOpt team, Inria research centre Saclay and CMAP, Ecole Polytechnique, France
[2]Hanoi National University of Education, Vietnam
[3]Jožef Stefan Institute, Ljubljana, Slovenia

Abstract

The `bbob-largescale` test suite, containing 24 single-objective functions in continuous domain, extends the well-known single-objective noiseless `bbob` test suite *[HAN2009]*, which has been used since 2009 in the BBOB workshop series, to large dimension. The core idea is to make the rotational transformations $\mathbf{R}, \mathbf{Q}$ in search space that appear in the `bbob` test suite computationally cheaper while retaining some desired properties. This documentation presents an approach that replaces a full rotational transformation with a combination of a block-diagonal matrix and two permutation matrices in order to construct test functions whose computational and memory costs scale linearly in the dimension of the problem.

Contents

1 Introduction

In the `bbob-largescale` test suite, we consider single-objective, unconstrained minimization problems of the form

$$\min_{x \in \mathbb{R}^n} \ f(x),$$

with problem dimensions $n \in \{20, 40, 80, 160, 320, 640\}$.

The objective is to find, as quickly as possible, one or several solutions x in the search space $\mathbb{R}^n$ with *small* value(s) of $f(x) \in \mathbb{R}$. We generally measure the *time* of an optimization run as the number of calls to (queries of) the objective function f.

We remind in the next sections some notations and definitions.

1.1 Terminology

function We talk about an objective *function* f as a parametrized mapping $\mathbb{R}^n \to \mathbb{R}$ with scalable input space, that is, n is not (yet) determined. Functions are parametrized such that different *instances* of the "same" function are available, e.g. translated or rotated versions.

problem We talk about a *problem*, `coco_problem_t`, as a specific *function instance* on which an optimization algorithm is run. Specifically, a problem can be described as the triple `(dimension, function, instance)`. A problem can be evaluated and returns an f-value. In the context of performance assessment, a target f- or indicator-value is attached to each problem. That is, a target value is added to the above triple to define a single problem in this case.

runtime We define *runtime*, or *run-length* as the *number of evaluations* conducted on a given problem, also referred to as number of *function* evaluations. Our central performance measure is the runtime until a given target value is hit.

suite A test- or benchmark-suite is a collection of problems, typically between twenty and a hundred.

1.2 Functions, Instances and Problems

Each function is *parametrized* by the (input) dimension, n, its identifier i, and the instance number, j, that is:

$$f_i^j \equiv f(n, i, j) : \mathbb{R}^n \to \mathbb{R} \quad x \mapsto f_i^j(x) = f(n, i, j)(x).$$

Varying n or j leads to a variation of the same function i of a given suite. By fixing n and j for function f_i, we define an optimization **problem** $(n, i, j) \equiv (f_i, n, j)$ that can be presented to the optimization algorithm. Each problem receives again an index in the suite, mapping the triple (n, i, j) to a single number.

We can think of j as an index to a continuous parameter vector setting, as it parametrizes, among others things, translations and rotations. In practice, j is the discrete identifier for single instantiations of these parameters.

1.3 Runtime and Target Values

In order to measure the runtime of an algorithm on a problem, we establish a hitting time condition. For a given problem (f_i, n, j), we prescribe a **target value** t as a specific f-value of interest *[HAN2016perf]*. For a single run, when an algorithm reaches or surpasses the target value t on problem (f_i, n, j), we say that it has *solved the problem* (f_i, n, j, t) — it was successful.[1]

The **runtime** is, then, the evaluation count when the target value t was reached or surpassed for the first time. That is, the runtime is the number of f-evaluations needed to solve the problem (f_i, n, j, t).[2] *Measured runtimes are the only way how we assess the performance of an algorithm.* Observed success rates are generally translated into runtimes on a subset of problems.

If an algorithm does not hit the target in a single run, its runtime remains undefined — while, then, this runtime is bounded from below by the number of evaluations in this unsuccessful run. The number of available runtime values depends on the budget the algorithm has explored (the larger the budget, the more likely the target-values are reached). Therefore, larger budgets are preferable — however they should not come at the expense of abandoning reasonable termination conditions. Instead, restarts should be done *[HAN2016ex]*.

2 Overview of the Proposed `bbob-largescale` Test Suite

The `bbob-largescale` test suite provides 24 functions in six dimensions (20, 40, 80, 160, 320 and 640) within the COCO framework *[HAN2016co]*. It is derived from the existing single-objective, unconstrained `bbob` test suite with modifications that allow the user to benchmark algorithms on high dimensional problems efficiently. We will explain in this section how the `bbob-largescale` test suite is built.

[1] Note the use of the term *problem* in two meanings: as the problem the algorithm is benchmarked on, (f_i, n, j), and as the problem, (f_i, n, j, t), an algorithm can solve by hitting the target t with the runtime, $\mathrm{RT}(f_i, n, j, t)$, or may fail to solve. Each problem (f_i, n, j) gives raise to a collection of dependent problems (f_i, n, j, t). Viewed as random variables, the events $\mathrm{RT}(f_i, n, j, t)$ given (f_i, n, j) are not independent events for different values of t.

[2] Target values are directly linked to a problem, leaving the burden to properly define the targets with the designer of the benchmark suite. The alternative is to present final f-values as results, leaving the (rather unsurmountable) burden to interpret these values to the reader. Fortunately, there is an automatized generic way to generate target values from observed runtimes, the so-called run-length based target values *[HAN2016perf]*.

2.1 The single-objective bbob functions

The bbob test suite relies on the use of a number of raw functions from which 24 bbob functions are generated. Initially, so-called *raw* functions are designed. Then, a series of transformations on these raw functions, such as linear transformations (e.g., translation, rotation, scaling) and/or non-linear transformations (e.g., $T_{\text{osz}}, T_{\text{asy}}$) will be applied to obtain the actual bbob test functions. For example, the test function $f_{13}(\mathbf{x})$ (Sharp Ridge function) with (vector) variable $\mathbf{x}$ is derived from a raw function defined as follows:

$$f_{\text{raw}}^{\text{Sharp Ridge}}(\mathbf{z}) = z_1^2 + 100\sqrt{\sum_{i=2}^{n} z_i^2}.$$

Then one applies a sequence of transformations: a translation by using the vector $\mathbf{x}^{\text{opt}}$; then a rotational transformation $\mathbf{R}$; then a scaling transformation $\mathbf{\Lambda}^{10}$; then another rotational transformation $\mathbf{Q}$ to get the relationship $\mathbf{z} = \mathbf{Q}\mathbf{\Lambda}^{10}\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})$; and finally a translation in objective space by using $\mathbf{f}_{\text{opt}}$ to obtain the final function in the testbed:

$$f_{13}(\mathbf{x}) = f_{\text{raw}}^{\text{Sharp Ridge}}(\mathbf{z}) + \mathbf{f}_{\text{opt}}.$$

There are two main reasons behind the use of transformations here:

1. provide non-trivial problems that cannot be solved by simply exploiting some of their properties (separability, optimum at fixed position, ...) and
2. allow to generate different instances, ideally of similar difficulty, of the same problem by using different (pseudo-)random transformations.

Rotational transformations are used to avoid separability and thus coordinate system dependence in the test functions. The rotational transformations consist in applying an orthogonal matrix to the search space: $x \to z = \mathbf{R}x$, where $\mathbf{R}$ is the orthogonal matrix. While the other transformations used in the bbob test suite could be naturally extended to the large scale setting due to their linear complexity, rotational transformations have quadratic time and space complexities. Thus, we need to reduce the complexity of these transformations in order for them to be usable, in practice, in the large scale setting.

2.2 Extension to large scale setting

Our objective is to construct a large scale test suite where the cost of a function call is acceptable in higher dimensions while preserving the main characteristics of the original functions in the bbob test suite. To this end, we will replace the full orthogonal matrices of the rotational transformations, which would be too expensive in our large scale setting, with orthogonal transformations that have linear complexity in the problem dimension: *permuted orthogonal block-diagonal matrices* (*[AIT2016]*).

Specifically, the matrix of a rotational transformation $\mathbf{R}$ will be represented as:

$$\mathbf{R} = P_{\text{left}} B P_{\text{right}}.$$

Here, P_{left} and P_{right} are two permutation matrices[3] and B is a block-diagonal matrix of the form:

$$B = \begin{pmatrix} B_1 & 0 & \dots & 0 \\ 0 & B_2 & \dots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \dots & B_{n_b} \end{pmatrix},$$

where n_b is the number of blocks and $B_i, 1 \leq i \leq n_b$ are square matrices of sizes $s_i \times s_i$ satisfying $s_i \geq 1$ and $\sum_{i=1}^{n_b} s_i = n$. In this case, the matrices $B_i, 1 \leq i \leq n_b$ are all orthogonal. Thus, the matrix B is also an orthogonal matrix.

This representation allows the rotational transformation **R** to satisfy three desired properties:

1. Have (almost) linear cost (due to the block structure of B).
2. Introduce non-separability.
3. Preserve the eigenvalues and therefore the condition number of the original function when it is convex quadratic (since **R** is orthogonal).

2.3 Generating the orthogonal block matrix B

The block-matrices $B_i, i = 1, 2, ..., n_b$ will be uniformly distributed in the set of orthogonal matrices of the same size. To this end, we first generate square matrices with sizes s_i (*i=1,2,...,n_b*) whose entries are i.i.d. standard normally distributed. Then we apply the Gram-Schmidt process to orthogonalize these matrices.

The parameters of this procedure include:

- the dimension of the problem n,
- the block sizes $s_1, \dots, s_{n_b}$, where n_b is the number of blocks. In this test suite, we set $s_i = s := \min\{n, 40\} \forall i = 1, 2, ..., n_b$ (except, maybe, for the last block which can be smaller)[4] and thus $n_b = \lceil n/s \rceil$.

2.4 Generating the permutation matrices P

In order to generate the permutation matrix P, we start from the identity matrix and apply, successively, a set of so-called *truncated uniform swaps*. Each row/column (up to a maximum number of swaps) is swapped with a row/column chosen uniformly from the set of rows/columns within a fixed range r_s. A random order of the rows/columns is generated to avoid biases towards the first rows/columns.

[3] A *permutation matrix* is a square binary matrix that has exactly one entry of 1 in each row and each column and 0s elsewhere.

[4] This setting allows to have the problems in dimensions 20 and 40 overlap between the `bbob` test suite and its large-scale extension since in these dimensions, the block sizes coincide with the problem dimensions.

Let i be the index of the first variable/row/column to be swapped and j be the index of the second swap variable. Then

$$j \sim U(\{l_b(i), l_b(i)+1, \ldots, u_b(i)\}\backslash\{i\}),$$

where $U(S)$ is the uniform distribution over the set S and $l_b(i) = \max(1, i - r_s)$ and $l_b(i) = \min(n, i + r_s)$ with r_s a parameter of the approach. If $r_s \leq (n-1)/2$, the average distance between the first and the second swap variable ranges from $(\sqrt{2}-1)r_s + 1/2$ (in the case of an asymmetric choice for j, i.e. when i is chosen closer to 1 or n than r_s) to $r_s/2 + 1/2$ (in the case of a symmetric choice for j). It is maximal when the first swap variable is at least r_s away from both extremes or is one of them.

Algorithm 1 below describes the process of generating a permutation using a series of truncated uniform swaps with the following parameters:

- n, the number of variables,
- n_s, the number of swaps.
- r_s, the swap range.

Starting with the identity permutation p and another permuation π, drawn uniform at random, we apply the swaps defined above by taking $p_\pi(1), p_\pi(2), \ldots, p_\pi(n_s)$, successively, as first swap variable. The resulting vector p will be the desired permutation.

Algorithm 1: Truncated Uniform Permutations

- Inputs: problem dimension n, number of swaps n_s, swap range r_s.
- Output: a vector $\mathbf{p} \in \mathbb{N}^n$, defining a permutation.

1. $\mathbf{p} \leftarrow (1, \ldots, n)$
2. Generate a permutation π uniformly at random
3. **for** $1 \leq k \leq n_s$ **do**
4. • $i \leftarrow \pi(k)$, i.e., $\mathbf{p}_{\pi(k)}$ is the first swap variable
5. • $l_b \leftarrow \max(1, i - r_s)$
6. • $u_b \leftarrow \min(n, i + r_s)$
7. • $S \leftarrow \{l_b, l_b + 1, \ldots, u_b\}\backslash\{i\}$
8. • Sample j uniformly at random in S
9. • Swap $\mathbf{p}_i$ and $\mathbf{p}_j$
10. **end for**
11. **return p**

In this test suite, we set $n_s = n$ and $r_s = \lfloor n/3 \rfloor$. Some numerical results in *[AIT2016]* show that with such parameters, the proportion of variables that are moved from their original position when

applying Algorithm 1 is approximately 100% for all dimensions 20, 40, 80, 160, 320, and 640 of the bbob-largescale test suite.

2.5 Implementation

Now, we describe how these changes to the rotational transformations are implemented with the realizations of $P_{\text{left}} B P_{\text{right}}$. This will be illustrated through an example on the Ellipsoidal function (rotated) $f_{10}(\mathbf{x})$ (see the table in the next section), which is defined by

$$f_{10}(\mathbf{x}) = \gamma(n) \times \sum_{i=1}^{n} 10^{6\frac{i-1}{n-1}} z_i^2 + \mathbf{f}_{\text{opt}}, \text{with } \mathbf{z} = T_{\text{osz}}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})), \mathbf{R} = P_1 B P_2,$$

as follows:

(i) First, we obtain the three matrices needed for the transformation, B, P_1, P_2, as follows:

```
coco_compute_blockrotation(B, seed1, n, s, n_b);
coco_compute_truncated_uniform_swap_permutation(P1, seed2, n,␣
↪n_s, r_s);
coco_compute_truncated_uniform_swap_permutation(P2, seed3, n,␣
↪n_s, r_s);
```

2. Then, whereever in the bbob test suite, we use the following

```
problem = transform_vars_affine(problem, R, b, n);
```

to make a rotational transformation, then in the bbob-largescale test suite, we replace it with the three transformations

```
problem = transform_vars_permutation(problem, P2, n);
problem = transform_vars_blockrotation(problem, B, n, s, n_b);
problem = transform_vars_permutation(problem, P1, n);
```

Here, n is again the problem dimension, s the size of the blocks in B, n_b : the number of blocks, n_s : the number of swaps, and r_s : the swap range as presented previously.

Important remark: Although the complexity of bbob test suite is reduced considerably by the above replacement of rotational transformations, we recommend running the experiment on the bbob-largescale test suite in parallel.

3 Functions in bbob-largescale test suite

The table below presents the definition of all 24 functions of the bbob-largescale test suite in detail. Beside the important modification on rotational transformations, we also make two changes to the raw functions in the bbob test suite.

- All functions, except for the Schwefel, Schaffer, Weierstrass, Gallagher, and Katsuura functions, are normalized by the parameter $\gamma(n) = \min(1, 40/n)$ to have uniform target values that are comparable, in difficulty, over a wide range of dimensions.
- The Discus, Bent Cigar and Sharp Ridge functions are generalized such that they have a constant proportion of distinct axes that remain consistent with the `bbob` test suite.

For a better understanding of the properties of these functions and for the definitions of the used transformations and abbreviations, we refer the reader to the original `bbob` function documention for details.

Acknowledgments

This work was supported by the grant ANR-12-MONU-0009 (NumBBO) of the French National Research Agency. This work was further supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, in a joint call with Gaspard Monge Program for optimization, operations research and their interactions with data sciences.

References

[AIT2016] O. Ait Elhara, A. Auger, N. Hansen (2016). Permuted Orthogonal Block-Diagonal Transformation Matrices for Large Scale Optimization Benchmarking. GECCO 2016, Jul 2016, Denver, United States.

[HAN2009] N. Hansen, S. Finck, R. Ros, and A. Auger (2009). Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Research Report RR-6829, Inria, updated February 2010.

[HAN2016ex] N. Hansen, T. Tusar, A. Auger, D. Brockhoff, O. Mersmann (2016). COCO: The Experimental Procedure, *ArXiv e-prints*, arXiv:1603.08776.

[HAN2016perf] N. Hansen, A. Auger, D. Brockhoff, D. Tusar, T. Tusar (2016). COCO: Performance Assessment. *ArXiv e-prints*, arXiv:1605.03560.

[AN2016co] Nikolaus Hansen, Anne Auger, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff (2016). COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting, *ArXiv e-prints*, arXiv:1603.08785.

Table 1: Function descriptions of the separable, moderate, and ill-conditioned function groups of the `bbob-largescale` test suite.

	Formulation	Transformations
Group 1: Separable functions		
Sphere Function	$f_1(\mathbf{x}) = \gamma(n) \times \sum_{i=1}^{n} z_i^2 + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = \mathbf{x} - \mathbf{x}^{\text{opt}}$
Ellipsoidal Function	$f_2(\mathbf{x}) = \gamma(n) \times \sum_{i=1}^{n} 10^{6\frac{i-1}{n-1}} z_i^2 + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = T_{\text{osz}}\left(\mathbf{x} - \mathbf{x}^{\text{opt}}\right)$
Rastrigin Function	$f_3(\mathbf{x}) = \gamma(n) \times \left(10n - 10\sum_{i=1}^{n} \cos\left(2\pi z_i\right) + \|\|z\|\|^2\right) + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = \boldsymbol{\Lambda}^{10} T_{\text{asy}}^{0.2}\left(T_{\text{osz}}\left(\mathbf{x} - \mathbf{x}^{\text{opt}}\right)\right)$
Bueche-Rastrigin Function	$f_4(\mathbf{x}) = \gamma(n) \times \left(10n - 10\sum_{i=1}^{n} \cos\left(2\pi z_i\right) + \|\|z\|\|^2\right) + 100 f_{pen}(\mathbf{x}) + \mathbf{f}_{\text{opt}}$	$z_i = s_i T_{\text{osz}}\left(x_i - x_i^{\text{opt}}\right)$ for $i = 1, \ldots, n$ $s_i = \begin{cases} 10 \times 10^{\frac{1}{2}\frac{i-1}{n-1}} & \text{if } z_i > 0 \text{ and } i \text{ odd} \\ 10^{\frac{1}{2}\frac{i-1}{n-1}} & \text{otherwise} \end{cases}$ for $i = 1, \ldots, n$
Linear Slope	$f_5(\mathbf{x}) = \gamma(n) \times \sum_{i=1}^{n} \left(5\|s_i\| - s_i z_i\right) + \mathbf{f}_{\text{opt}}$	$z_i = \begin{cases} x_i & \text{if } x_i^{\text{opt}} x_i < 5^2 \\ x_i^{\text{opt}} & \text{otherwise} \end{cases}$ for $i = 1, \ldots, n$, $s_i = \text{sign}\left(x_i^{\text{opt}}\right) 10^{\frac{i-1}{n-1}}$ for $i = 1, \ldots, n$, $\mathbf{x}^{\text{opt}} = \mathbf{z}^{\text{opt}} = 5 \times \mathbf{1}_-^+$
Group 2: Functions with low or moderate conditioning		
Attractive Sector Function	$f_6(\mathbf{x}) = T_{\text{osz}}\left(\gamma(n) \times \sum_{i=1}^{n} \left(s_i z_i\right)^2\right)^{0.9} + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = \mathbf{Q}\boldsymbol{\Lambda}^{10}\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})$ with $\mathbf{R} = P_{11}B_1P_{12}, \mathbf{Q} = P_{21}B_2P_{22}$, $s_i = \begin{cases} 10^2 & \text{if } z_i \times x_i^{\text{opt}} > 0 \\ 1 & \text{otherwise} \end{cases}$ for $i = 1, \ldots, n$
Step Ellipsoidal Function	$f_7(\mathbf{x}) = \gamma(n) \times 0.1 \max\left(\|\hat{z}_1\|/10^4, \sum_{i=1}^{n} 10^{2\frac{i-1}{n-1}} z_i^2\right) + f_{pen}(\mathbf{x}) + \mathbf{f}_{\text{opt}}$	$\hat{\mathbf{z}} = \boldsymbol{\Lambda}^{10}\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})$ with $\mathbf{R} = P_{11}B_1P_{12}$, $\tilde{z}_i = \begin{cases} \lfloor 0.5 + \hat{z}_i \rfloor & \text{if } \|\hat{z}_i\| > 0.5 \\ \lfloor 0.5 + 10\hat{z}_i \rfloor / 10 & \text{otherwise} \end{cases}$ for $i = 1, \ldots, n$, $\mathbf{z} = \mathbf{Q}\tilde{\mathbf{z}}$ with $\mathbf{Q} = P_{21}B_2P_{22}$
Rosenbrock Function, original	$f_8(\mathbf{x}) = \gamma(n) \times \sum_{i=1}^{n} \left(100\left(z_i^2 - z_{i+1}\right)^2 + \left(z_i - 1\right)^2\right) + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = \max\left(1, \frac{\sqrt{s}}{8}\right)\left(\mathbf{x} - \mathbf{x}^{\text{opt}}\right) + \mathbf{1}$, $\mathbf{x}^{\text{opt}} \in [-3, 3]^n$
Rosenbrock Function, rotated	$f_9(\mathbf{x}) = \gamma(n) \times \sum_{i=1}^{n} \left(100\left(z_i^2 - z_{i+1}\right)^2 + \left(z_i - 1\right)^2\right) + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = \max\left(1, \frac{\sqrt{s}}{8}\right)\mathbf{R}\left(\mathbf{x} - \mathbf{x}^{\text{opt}}\right) + \mathbf{1}$ with $\mathbf{R} = P_1BP_2$, $\mathbf{x}^{\text{opt}} \in [-3, 3]^n$
Group 3: Functions with high conditioning and unimodal		
Ellipsoidal Function	$f_{10}(\mathbf{x}) = \gamma(n) \times \sum_{i=1}^{n} 10^{6\frac{i-1}{n-1}} z_i^2 + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = T_{\text{osz}}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$ with $\mathbf{R} = P_1BP_2$
Discus Function	$f_{11}(\mathbf{x}) = \gamma(n) \times \left(10^6 \sum_{i=1}^{\lceil n/40 \rceil} z_i^2 + \sum_{i=\lceil n/40 \rceil + 1}^{n} z_i^2\right) + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = T_{\text{osz}}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$ with $\mathbf{R} = P_1BP_2$
Bent Cigar Function	$f_{12}(\mathbf{x}) = \gamma(n) \times \left(\sum_{i=1}^{\lceil n/40 \rceil} z_i^2 + 10^6 \sum_{i=\lceil n/40 \rceil + 1}^{n} z_i^2\right) + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = \mathbf{R}T_{\text{asy}}^{0.5}(\mathbf{R}((\mathbf{x} - \mathbf{x}^{\text{opt}}))$ with $\mathbf{R} = P_1BP_2$
Sharp Ridge Function	$f_{13}(\mathbf{x}) = \gamma(n) \times \left(\sum_{i=1}^{\lceil n/40 \rceil} z_i^2 + 100\sqrt{\sum_{i=\lceil n/40 \rceil + 1}^{n} z_i^2}\right) + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = \mathbf{Q}\boldsymbol{\Lambda}^{10}\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})$ with $\mathbf{R} = P_{11}B_1P_{12}, \mathbf{Q} = P_{21}B_2P_{22}$
Different Powers Function	$f_{14}(\mathbf{x}) = \gamma(n) \times \sum_{i=1}^{n} \|z_i\|^{\left(2+4\times\frac{i-1}{n-1}\right)} + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = \mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})$ with $\mathbf{R} = P_1BP_2$

Table 2: Function descriptions of the multi-modal function group with adequate global structure of the `bbob-largescale` test suite.

	Formulation	Transformations
Group 4: Multi-modal functions with adequate global structure		
Rastrigin Function	$f_{15}(\mathbf{x}) = \gamma(n) \times \left(10n - 10\sum_{i=1}^{n} \cos(2\pi z_i) + \|\mathbf{z}\|^2\right) + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = \mathbf{R}\boldsymbol{\Lambda}^{10}\mathbf{Q}T_{\text{asy}}^{0.2}\left(T_{\text{osz}}\left(\mathbf{R}\left(\mathbf{x} - \mathbf{x}^{\text{opt}}\right)\right)\right)$ with $\mathbf{R} = P_{11}B_1P_{12}, \mathbf{Q} = P_{21}B_2P_{22}$
Weierstrass Function	$f_{16}(\mathbf{x}) = 10\left(\frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{11}\frac{1}{2^k}\cos\left(2\pi 3^k\left(z_i + 1/2\right)\right) - f_0\right)^3 + \frac{10}{n}f_{pen}(\mathbf{x}) + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = \mathbf{R}\boldsymbol{\Lambda}^{1/100}\mathbf{Q}T_{\text{osz}}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$ with $\mathbf{R} = P_{11}B_1P_{12}, \mathbf{Q} = P_{21}B_2P_{22}$, $f_0 = \sum_{k=0}^{11}\frac{1}{2^k}\cos(\pi 3^k)$
Schaffers F7 Function	$f_{17}(\mathbf{x}) = \left(\frac{1}{n-1}\sum_{i=1}^{n-1}\left(\sqrt{s_i} + \sqrt{s_i}\sin^2\left(50(s_i)^{1/5}\right)\right)\right)^2 + 10f_{pen}(\mathbf{x}) + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = \boldsymbol{\Lambda}^{10}\mathbf{Q}T_{\text{asy}}^{0.5}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$ with $\mathbf{R} = P_{11}B_1P_{12}, \mathbf{Q} = P_{21}B_2P_{22}$, $s_i = \sqrt{z_i^2 + z_{i+1}^2}, i = 1, \ldots, n-1$
Schaffers F7 Function, moderately ill-conditioned	$f_{18}(\mathbf{x}) = \left(\frac{1}{n-1}\sum_{i=1}^{n-1}\left(\sqrt{s_i} + \sqrt{s_i}\sin^2\left(50(s_i)^{1/5}\right)\right)\right)^2 + 10f_{pen}(\mathbf{x}) + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = \boldsymbol{\Lambda}^{1000}\mathbf{Q}T_{\text{asy}}^{0.5}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$ with $\mathbf{R} = P_{11}B_1P_{12}, \mathbf{Q} = P_{21}B_2P_{22}$, $s_i = \sqrt{z_i^2 + z_{i+1}^2}, i = 1, \ldots, n-1$
Composite Griewank-Rosenbrock Function F8F2	$f_{19}(\mathbf{x}) = \frac{10}{n-1}\sum_{i=1}^{n-1}\left(\frac{s_i}{4000} - \cos(s_i)\right) + 10 + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = \max\left(1, \frac{\sqrt{s}}{8}\right)\mathbf{R}\mathbf{x} + \frac{\mathbf{1}}{2}$ with $\mathbf{R} = P_1BP_2$, $s_i = 100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2$, for $i = 1, \ldots, n-1, \mathbf{z}^{\text{opt}} = \mathbf{1}$

Table 3: Function descriptions of the multi-modal function group with weak global structure of the `bbob-largescale` test suite.

	Formulation	Transformations
Group 5: Multi-modal functions with weak global structure		
Schwefel Function	$f_{20}(\mathbf{x}) = -\frac{1}{100n}\sum_{i=1}^{n} z_i \sin\left(\sqrt{\lvert z_i\rvert}\right) + 4.189828872724339 + 100 f_{pen}(\mathbf{z}/100) + \mathbf{f}_{\text{opt}}$	$\hat{\mathbf{x}} = 2 \times \mathbf{1}^+_- \otimes \mathbf{x}$, $\hat{z}_1 = \hat{x}_1, \hat{z}_{i+1} = \hat{x}_{i+1} + 0.25\left(\hat{x}_i - 2\left\lvert x_i^{\text{opt}}\right\rvert\right)$, for $i = 1, \ldots, n-1$, $\mathbf{z} = 100\left(\boldsymbol{\Lambda}^{10}\left(\hat{\mathbf{z}} - 2\lvert\mathbf{x}^{\text{opt}}\rvert\right) + 2\lvert\mathbf{x}^{\text{opt}}\rvert\right)$, $\mathbf{x}^{\text{opt}} = 4.2096874633/2\mathbf{1}^+_-$
Gallagher's Gaussian 101-me Peaks Function	$f_{21}(\mathbf{x}) = T_{\text{osz}}\left(10 - \max_{i=1}^{101} w_i \exp\left(-\frac{1}{2n}(\mathbf{z}-\mathbf{y}_i)^T\mathbf{B}^T\mathbf{C_i}\mathbf{B}(\mathbf{z}-\mathbf{y}_i)\right)\right)^2 + f_{pen}(\mathbf{x}) + \mathbf{f}_{\text{opt}}$	$w_i = \begin{cases} 1.1 + 8 \times \frac{i-2}{99} & \text{for } 2 \le i \le 101 \\ 10 & \text{for } i = 1 \end{cases}$ $\mathbf{B}$ is a block-diagonal matrix without permuations of the variables. $\mathbf{C_i} = \Lambda^{\alpha_i}/\alpha_i^{1/4}$ where Λ^{α_i} is defined as usual, but with randomly permuted diagonal elements. For $i = 2, \ldots, 101$, α_i is drawn uniformly from the set $\left\{1000^{2\frac{j}{99}}, j = 0, \ldots, 99\right\}$ without replacement, and $\alpha_i = 1000$ for $i = 1$. The local optima $\mathbf{y}_i$ are uniformly drawn from the domain $[-5, 5]^n$ for $i = 2, ..., 101$ and $\mathbf{y}_1 \in [-4, 4]^n$. The global optimum is at $\mathbf{x}^{\text{opt}} = \mathbf{y}_1$.
Gallagher's Gaussian 21-hi Peaks Function	$f_{22}(\mathbf{x}) = T_{\text{osz}}\left(10 - \max_{i=1}^{21} w_i \exp\left(-\frac{1}{2n}(\mathbf{z}-\mathbf{y}_i)^T\mathbf{B}^T\mathbf{C_i}\mathbf{B}(\mathbf{z}-\mathbf{y}_i)\right)\right)^2 + f_{pen}(\mathbf{x}) + \mathbf{f}_{\text{opt}}$	$w_i = \begin{cases} 1.1 + 8 \times \frac{i-2}{19} & \text{for } 2 \le i \le 21 \\ 10 & \text{for } i = 1 \end{cases}$ $\mathbf{B}$ is a block-diagonal matrix without permuations of the variables. $\mathbf{C_i} = \Lambda^{\alpha_i}/\alpha_i^{1/4}$ where Λ^{α_i} is defined as usual, but with randomly permuted diagonal elements. For $i = 2, \ldots, 21$, α_i is drawn uniformly from the set $\left\{1000^{2\frac{j}{19}}, j = 0, \ldots, 19\right\}$ without replacement, and $\alpha_i = 1000^2$ for $i = 1$. The local optima $\mathbf{y}_i$ are uniformly drawn from the domain $[-4.9, 4.9]^n$ for $i = 2, ..., 21$ and $\mathbf{y}_1 \in [-3.92, 3.92]^n$. The global optimum is at $\mathbf{x}^{\text{opt}} = \mathbf{y}_1$.
Katsuura Function	$f_{23}(\mathbf{x}) = \left(\frac{10}{n^2}\prod_{i=1}^{n}\left(1 + i\sum_{j=1}^{32}\frac{\lvert 2^j z_i - [2^j z_i]\rvert}{2^j}\right)^{10/n^{1.2}} - \frac{10}{n^2}\right) + f_{pen}(\mathbf{x}) + \mathbf{f}_{\text{opt}}$	$\mathbf{z} = \mathbf{Q}\boldsymbol{\Lambda}^{100}\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})$ with $\mathbf{R} = P_{11}B_1P_{12}, \mathbf{Q} = P_{21}B_2P_{22}$
Lunacek bi-Rastrigin Function	$f_{24}(\mathbf{x}) = \gamma(n) \times \left(\min\left(\sum_{i=1}^{n}(\hat{x}_i - \mu_0)^2, n + s\sum_{i=1}^{n}(\hat{x}_i - \mu_1)^2\right) + 10\left(n - \sum_{i=1}^{n}\cos(2\pi z_i)\right)\right) + 10^4 f_{pen}(\mathbf{x}) + \mathbf{f}_{\text{opt}}$	$\hat{\mathbf{x}} = 2\text{sign}(\mathbf{x}^{\text{opt}}) \otimes \mathbf{x}, \mathbf{x}^{\text{opt}} = 0.5\mu_0\mathbf{1}^+_-, \mathbf{z} = \mathbf{Q}\boldsymbol{\Lambda}^{100}\mathbf{R}(\hat{\mathbf{x}} - \mu_0\mathbf{1})$ with $\mathbf{R} = P_{11}B_1P_{12}, \mathbf{Q} = P_{21}B_2P_{22}$, $\mu_0 = 2.5, \mu_1 = -\sqrt{\frac{\mu_0^2 - 1}{s}}$, $s = 1 - \frac{1}{2\sqrt{n+20} - 8.2}$