Using structural bias to analyse the behaviour of modular CMA-ES

Vermetten, D.L.; Caraffini, F.; Stein, B. van; Kononova, A.V.; Fieldsend, J.E.

Using Structural Bias to Analyse the Behaviour of Modular CMA-ES

Diederick Vermetten
LIACS, Leiden University
The Netherlands
d.l.vermetten@liacs.leidenuniv.nl

Fabio Caraffini
Institute of Artificial Intelligence,
De Montfort University
Leicester, UK
fabio.caraffini@dmu.ac.uk

Bas van Stein*
LIACS, Leiden University
The Netherlands
b.van.stein@liacs.leidenuniv.nl

Anna V. Kononova
LIACS, Leiden University
The Netherlands
a.kononova@liacs.leidenuniv.nl

ABSTRACT

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is a commonly used iterative optimisation heuristic for optimising black-box functions. CMA-ES comes in many flavours with different configuration settings. In this work, we investigate whether CMA-ES suffers from structural bias and which modules and parameters affect the strength and type of structural bias. Structural bias occurs when an algorithm or a component of the algorithm biases the search towards a specific direction in the search space irrespective of the objective function. In addition to this investigation, we propose a method to assess the relationship between structural bias and the performance of configurations with different types of bias on the BBOB suite of benchmark functions. Surprisingly for such a popular algorithm, 90.3% of the 1 620 CMA-ES configurations were found to have Structural Bias. Some interesting patterns between module settings and bias types are presented and further insights are discussed.

CCS CONCEPTS

• **Theory of computation** → *Theory of randomized search heuristics*; **Random search heuristics**; **Bio-inspired optimization**.

KEYWORDS

structural bias, algorithmic behaviour, evolutionairy strategies, benchmarking

ACM Reference Format:
Diederick Vermetten, Fabio Caraffini, Bas van Stein, and Anna V. Kononova. 2022. Using Structural Bias to Analyse the Behaviour of Modular CMA-ES. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3520304.3534035

*Corresponding author

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07.
https://doi.org/10.1145/3520304.3534035

1 INTRODUCTION

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is a very popular heuristic optimisation algorithm for continuous optimisation problems. CMA-ES is considered state-of-the-art in evolutionary computation and has been adopted as one of the standard tools for continuous optimisation in many research labs. There are many flavours and variants of CMA-ES developed through the years and different implementations of sub-components such as the sampling strategy and the boundary correction method. In recent works, the different modules and configurations of CMA-ES are explored and analysed based on their performance [4]. In that research a modular CMA-ES framework is presented, representing a plethora of different CMA-ES configurations. In this paper, the modular CMA-ES framework is used to analyse the many different aspects of CMA-ES with respect to Structural Bias (SB). Structural Bias is a known deficiency found in many iterative optimisation heuristics (IOHs), as previously shown in [3, 15, 16, 22] for differential evolution, single solution optimisation methods and compact optimisation methods. Does CMA-ES also suffer from Structural Bias? And if so, which modules and settings mostly affect the strength and type of bias? In addition to these questions, we investigate how the different types of structural bias that we find in CMA-ES configurations affect the performance of these algorithms. We do this by comparing configurations with different types of SB on different benchmark function instances where we know that the optimum of the instance is either close to the centre of the search space or far away. We also investigate the relation between the strength and type of bias with the fraction of infeasible solutions that the algorithms generate. Using a wide set of experiments and configurations we can finally come up with recommendations on using particular CMA-ES modules and raising awareness of potential issues in different configuration settings.

This paper is structured as follows: In Section 2 the modular CMA-ES framework is introduced in more detail and the methods for detecting Structural Bias and the strength of SB are presented. In Section 3 the first experiment is discussed, including setup and results, showing which CMA-ES configurations and modules have what kind of Structural Bias. In further experiments in Section 4, an analysis of the performance over different BBOB function instances is performed to see how different Bias types affect the performance. Finally we draw conclusion in Section 5.

2 METHODOLOGY

To investigate the strength and types of structural bias, we make use of several algorithmic frameworks and methods. In the following subsections, the Modular CMA-ES framework and employed benchmark functions are explained in detail. The concept of SB in stochastic optimisation algorithm is also defined, and an in-depth description of how it is measured is provided.

2.1 Modular CMA-ES

To investigate several commonly and less commonly used configurations and variants of the CMA-ES algorithm, we use the Modular CMA-ES framework as introduced by van Rijn et al. [21] and further developed by de Nobel et al. [4]. The framework is open-source and available as part of the IOHprofiler [5] environment[1]. The latter is further employed to run some of the experiments in this paper and easily access benchmark functions of the BBOB and COCO [11] software platforms.

The modular CMA-ES package consists of 11 modules with various options. These are Active update (2), Elitism (2), Orthogonal Sampling (2), Sequential Selection (2), Threshold Convergence (2), Step-Size adaptation (7), Mirrored Sampling (3), Quasi-Gaussian Sampling (3), Recombination Weights (2), Restart Strategy (3) and Boundary Correction (6). We indicated in brackets the number of possibilities for each module, giving a total of 72.576 possible configurations. A detailed description of each of these modules and their settings can be read in [4]. From previous work, we know that especially the strategy of dealing with infeasible solutions (SDIS) can greatly affect structural bias [3]. A SDIS transforms/corrects candidate solutions that are sampled outside the search domain into feasible points inside such domain. Without such component, the algorithm cannot be claimed to address constrained problems fairly. The following boundary correction methods are available in the framework:

- **uniform** Uniform Re-sample replaces all infeasible coordinates of a solution with new coordinates sampled uniformly at random within the given search space.
- **mirror** The mirror strategy mirrors all infeasible coordinates of a solution with respect to its closest boundary.
- **COTN** The Complete One-tailed Normal Correction Strategy replaces all infeasible coordinates with new coordinates inside the search space according to a re-scaled one-sided normal distribution centred on the boundary.
- **saturation** The Saturation strategy set all infeasible coordinates to the closest corresponding bound.
- **toroidal** The Toroidal strategy reflects all infeasible coordinates off the opposite boundary inwards.

However, the effect of these strategies is still unknown for CMA-ES. For all other modules, it is also unknown how these options affect structural bias and the type of structural bias, which we will investigate in this paper. For the experiments in our work a sub-set of 1 620 configurations, with two different initialisation methods (making a total of 3 240 algorithms), is used to make the experiments computational feasible. Some modules are left to their default setting as we assume these modules (such as Orthogonal Sampling) do not relate strongly to the presence of structural bias. Specific details of the used modules and their options are defined in Section 3.

[1] https://github.com/IOHprofiler/ModularCMAES

2.2 Structural bias

In the context of heuristic optimisation, SB refers to the set of algorithmic behaviours arising from the iterative application of the employed operators, including their parameter setting and their interplay, preventing the algorithm from equally exploring the entire search space regardless of the problem at hand.

When these artificial biases emerge during the search, they can interfere with the direction being explored by the algorithm to generate new candidate solutions. Hence, even if this can be beneficial to speed up the search, or to improve upon the quality of the returned near-optimal solution in some very specific cases, being structurally biased is generally a strong deficiency for a general-purpose algorithm meant to address black-box problems.

2.2.1 Structural bias and use of probabilistic objective function f_0. To separate the structural effects of the employed algorithmic framework from those induced by the problem at hand, the 'special' function f_0 was first introduced in [17]. By executing a heuristic for optimisation over f_0 (whose definition is reported in Equation (1)) for a number N of independent runs, its SB would show as non-uniform distribution of the obtained final best solutions.

$$f_0 : [0,1]^n \to [0,1], \text{ where } \forall x\ f_0(x) \sim \mathcal{U}(0,1). \tag{1}$$

Besides the mathematical proof in [17], one might also intuitively expect such an outcome on f_0, as its stochastic nature makes sure that each component of its minimum is randomly located within the feasible range without giving preferences to particular values in $[0,1]$. Hence, information obtained on the fitness by the cyclic application of the search operators forming a heuristic algorithm, and their search actions, gets 'decoupled' thus unveiling only the actual structural algorithmic behaviour. In turn, as the minima of f_0 across multiple runs are uniformly distributed, an unbiased algorithm too should return the same distribution of final best solutions. As such, failing to return a uniform distribution of optima found across runs is a clear symptom of SB.

2.2.2 Measuring and classifying structural bias. The simple experimental practice based on f_0 has become an established approach, turning the problem of detecting SB into the mathematical problem of testing for the uniformity of the distribution of the (near-) optimal solutions obtained over multiple independent optimisation processes.

By allotting components of the final solutions in dedicated axes, to be displayed in parallel coordinates [12, 17], first results on swarm intelligence and evolutionary algorithms have graphically shown such distributions not to follow the expected (uniform) pattern for many classic algorithmic under different parameters settings and configurations [2, 3, 17]. To remove errors from subjective interpretations of visual inspections of such results, and automatise the process, several statistical tests have been then investigated to process results from large and diverse sets of algorithms[2] and experimental setups [15, 16, 22, 24], before being able to design the

[2] These include single-solutions and estimation of distribution heuristic algorithms.

BIAS toolbox [26]. This represents the current state-of-the-art in detecting SB.

BIAS is an open-source software tool. Once fed with raw data (i.e. final best positions) from multiples optimisation runs over f_0, it applies 36 statistical tests on each dimension of the available optimal positions and aggregates them to make a decision on the presence/absence of SB. Furthermore, through the use two random forest models, it also returns the type of SB among 5 possible scenarios having the self-explanatory names of `bounds`, `clusters`, `centre`, `discretisation` and `none`.

2.2.3 Existing results on SB. Analysing SB reveals interesting patterns in the behaviour of several heuristics for real-valued continuous optimisation based on their parameters setting and employment of auxiliary operators.

An interesting relation between the number of individuals and SB strength exists for genetic algorithms, where this deficiency is expected to grow stronger with increasing population sizes [17]. The opposite scenario occurs for a standard particle swarm optimisation algorithm [15], which presents a strong bias for small swarm sizes. 'Compact' counterparts of these (and other established) population-based algorithm, i.e. simplified estimation of distribution algorithms mimicking the working logic of known heuristics, seems to be less plagued by SB, but still display mild levels of SB and, and very strong ones in some specific implementations [15] Single solution heuristic are too affected by SB [16]. In the latter, SB introduced during the algorithmic design process is difficult to be mitigated with parameter tuning or by changing auxiliary operators such as the adopted bound correction methods. This is not the case of differential evolution, where the strategy used for dealing with infeasible solutions can make the difference, see e.g. [3, 22, 25], with `saturation` being the one introducing most SB, while other strategies can be applied only with some specific parameter settings, e.g. `mirror` does not introduce SB if used in conjunction with a relatively high scale factor (i.e. > 1.13). Generally speaking, high values of both the control parameters cause more bias than low ones, apart from when some specific mutation operators are used, such as `DE/curr-to-rand/1`. This shows how complex mitigating SB can be, and how its emergence depends on many factors such as the interplay between operators and parameter configurations.

2.3 COCO and BBOB

COCO's optimisation benchmarking environment [10] has been established in the field of heuristic optimisation as a recommended tool a fair comparison with other state-of-the-art methods using state-of-the-art practices for benchmarking algorithms [1]. In the decade since its introduction, its suites of Black-Box Optimisation Benchmarks (BBOB) have continuously gained popularity, for example through its yearly workshops organised at GECCO and large set of publicly available performance data for a wide variety of optimisation algorithms. Among other things, COCO makes available a set of 24 single-objective, distinct noiseless functions [7] defined using box-constraints of $[-5, 5]^n$ referred to as *BBOB function set*. Each of these functions can be instantiated with different transformations – such as rescaling, translation and rotation, referred to as *instances* of these functions – leading to a different location of the global optimum and a different optimal function value [10]. Instances are identified by their numbers where instance 0 corresponds to the original untransformed function. Since all applied transformations are known, the location of optima can be computed for any instance, but generally only instances between 0 and 99 are considered.

The mechanism of generation of such instances is not ideal and has been criticised in the past concerning the reliability of features computed on the landscapes [19], especially in terms of stability across instances of the same function generated through translations or rotations [18], [13]. Other papers have concluded that differences between instances can be ignored [27], [20] or learnt from [6]. In our view, the absence of a definitive answer necessitates further studies on BBOB and its instance generating functionality.

3 BEHAVIOURAL CHARACTERISATION OF MODULAR CMA-ES

The first experiment covers a large number of CMA-ES configurations with commonly used settings and modules. In this experiment we use the BIAS toolbox [26] to detect the presence and type of SB in each of the configurations. In addition, we use the number of rejections of the null-hypothesis of uniformity from the statistical tests included in the BIAS toolbox to indicate the 'strength' or 'severity' of SB.

3.1 Experimental setup

For our experiments with the modular CMAE-ES framework [4], we make use of a reduced set of modules. The used setup contains a total of 1 620 configurations, constructed by using all combinations of the following modules: elitism (`false`, `true`), mirrored sampling (`mirrored`, `mirrored pairwise`, `none`), step-size adaptation (`cumulative step-size adaptation`, `population success rule`), base sampler (`Gaussian`, `Halton`, `Sobol'`), recombination weights ($\frac{1}{2}^{\lambda}$, `default`, `equal`), local restart (`BIPOP`, `IPOP`, `none`) and SDIS (`COTN`, `mirror`, `saturate`, `toroidal`, `uniform`). The remaining modules are set to their default values in the modular CMA-ES framework, with the initial step-size set of 20% of the domain as recommended in [9].

Using this large set of configurations we follow the methodology as described in Section 2.2, and minimise function f_0 (see Equation (1)) in $n = 30$ dimensions. All considered configurations run with a fixed budget of $10000 \times n$ fitness evaluations, in a series of 100 independent runs. The aforementioned setup is considered for 2 initialisation variants (which is determined by the location of the initial centre of mass): `centre` of the space as commonly used in the field [8] and `uniform` at random as recommended in [9].

The BIAS toolbox [26] is used to test for presence of SB and to predict the type of SB, `IOHprofiler` is used for logging the details of the optimisation process on f_0 and BBOB and processing this data [28].

3.2 Structural bias on CMA-ES variants

By analysing the outcome of the extensive aforementioned experimental setup, we discover that also state-of-the-art CMA-ES algorithms are not free from SB. Surprisingly, 90.3% of the 1 620

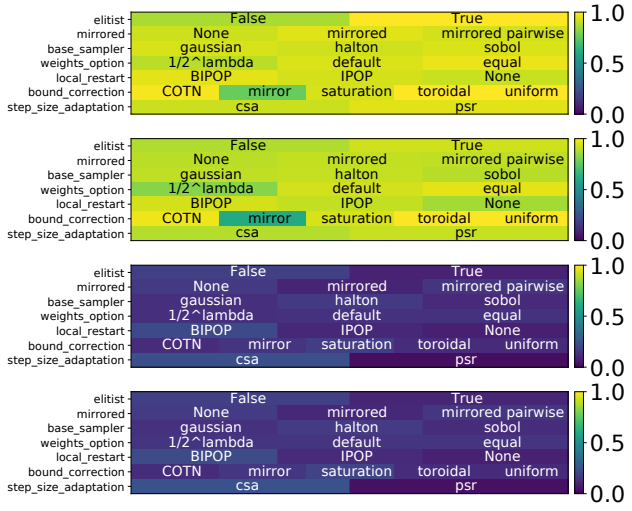

Figure 1: The top two heat maps show the fraction of configurations that are detected to have SB for each individual module option, while the lower two heat maps show the fraction of infeasible solutions per module option. The first and third heat map are using centre of the space initialisation and the second and fourth using uniform.

presents SB of various 'strength' (measured heuristically as previously explained) and kind, according to the BIAS toolbox.

Results showing SB are visually reported in the top two diagrams of Figure 1, which show the fractional contribution to SB carried by each module under investigation for the case of centre and uniform initialisation respectively, and in Figure 2, where the strength of the biases for all the configurations at hand is depicted in relation to the fraction of infeasible solutions generated by the algorithm during the search.

From the graphical results in the top part of Figure 1 we can see that the vast majority of module configurations lead to biased searches, regardless of the initialisation method. In both cases, the same modules are coherently contributing with similar intensity to the global SB, with a clear difference being made by the employed strategy for dealing with infeasible solutions, where mirror leads to the lowest proportion of biased configurations. While other modules also affect the overall number of biased configurations, their impact is comparatively small.

In line with the previous considerations, a close glance at the left-hand diagram of Figure 2 shows that SB appears with most deleterious strengths when saturation is used, followed by uniform and COTN - mirror is raked last. Interestingly, the saturation is the main responsible for discretisation SB, as can be deducted from the diagram on the right-hand side of 2, where all kind of detected biases are reported. Note that all possible kind of biases that can be detected with the BIAS toolbox appears in this graph, thus showing that different combinations of modules can make CMA-ES behave very differently. However, centre is the most frequent kind of SB. This is intuitively explainable by the fact that the working mechanism of CMA-ES is based on the use of a Normal distribution. Using the centre of the search space for initialising the search could be another reason for this finding.

3.3 Infeasibility

Unlike other evolutionary algorithms, see [14], most CMA-ES variants feature a more constrained search. Indeed, in support of this observation, the toolbox identifies the most biased configurations as having centre SB. Not surprisingly, in Figure 2 one can spot a negative correlation between the strength of this kind of SB and the proportion of infeasible solutions.

However, all configurations do generate infeasible solutions and, as highlighted in the previous section, the way used to handle them can significantly alter the algorithmic behaviour. This means that configurations having modules generating higher proportions of infeasible solutions would activate the selected SDIS more frequently, thus strengthening the SB. In turn, this might alter the search direction into generating more solutions violating the search boundaries. To easily identify such configurations, the fractional contribution per module on the proportion of infeasible solutions, for the centre and the uniform initialisation methods, are reported in the two bottom diagrams of Figure 1 respectively. An example of a module influencing the proportion of infeasible solutions is provided in Figure 3, which shows the consequences of employing a non-elitist approach versus an enlists one for the uniform initialisation methods. Note that the non-elitist approach introduces both unbiased and discretisation-biased cases, both leading to high numbers of infeasible solutions (probably due to the use of the saturation SDIS for the cases with discretisation SB). A full gallery of images showing both SB strengths and proportions of infeasible solutions per module is made available in our online repository at [23].

To conclude this analysis, it has to be pointed out that saturation is the strategy leading to more proportions of infeasible solutions, regardless of the initialisation method. This makes sense, given that it places infeasible solutions to the boundaries. Interestingly, a clear positive correlation between the strength of SB induced by saturation, i.e. discretisation, and the proportion of infeasible solutions emerge from Figure 2. This confirms that saturation should not be recommended for these configurations.

3.4 Impact of initialisation

From the observation made in the previous section, it is obvious that choosing the right initialisation method matters on CMA-ES. This aspect is often overlooked, with the most popular method consisting in centring the initial multivariate normal distribution at the centre of the search space. This might be a very practical and quick choice, used by practitioners and most researchers in the field [8], but not necessarily the best one. For this study, we compared the differences in terms of algorithmic behaviour while using a different, still common, initialisation method that randomly draws the mean vector of the initial distribution uniformly on the search space [9]. As we show that the second option leads to less severe SB, see Figure 1, we argue that the initialisation module is as important as other more complex modules, and we call for more attention to details that might lead to undesired results. Hence, for

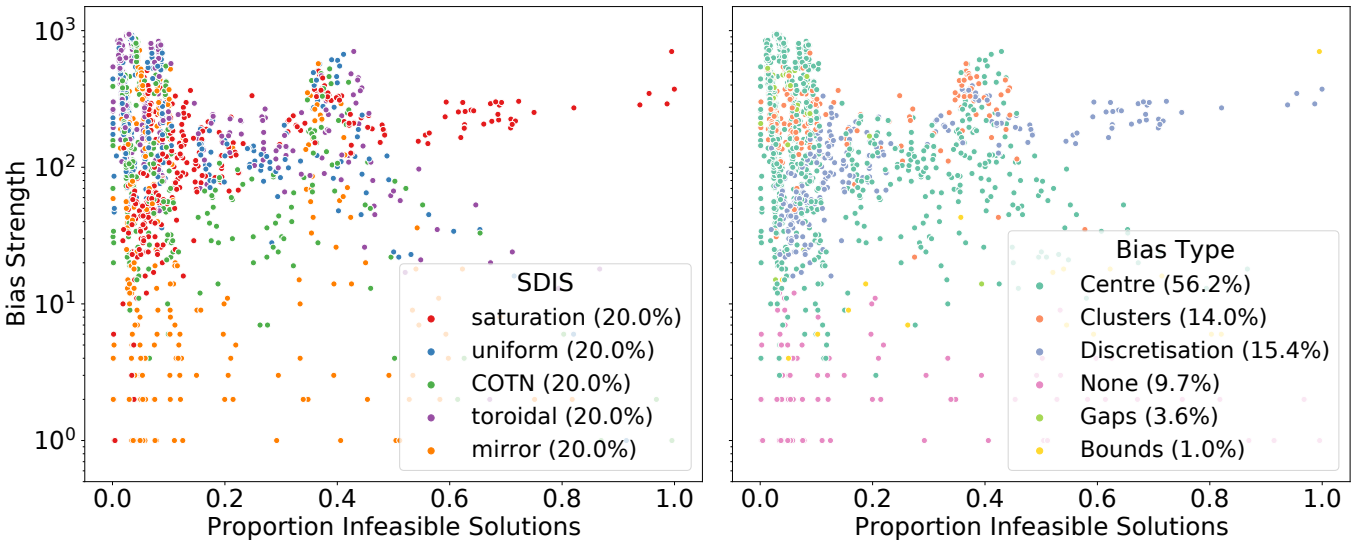

Figure 2: Bias Strength versus the fraction of infeasible solutions for all 1 620 configurations. Colours on the left plot indicate the boundary correction strategy (i.e. SDIS) and on the right the predicted type of SB using the BIAS toolbox.

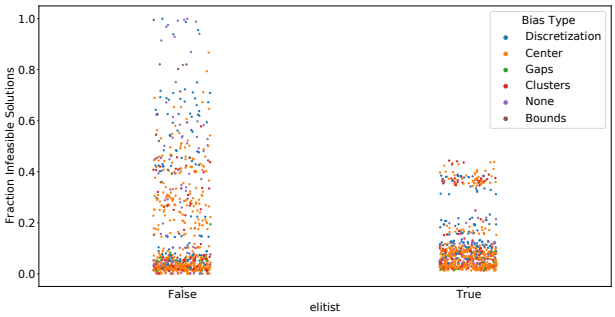

Figure 3: The proportion of infeasible solutions per configuration split by the setting of the `elitist` module, colours indicate the SB type for each configuration.

all cases where one does not know if the optimum is at the centre of the search space, we do not recommend using `centre` initialisation.

4 LOCATION-DEPENDENT PERFORMANCE COMPARISON ON BBOB

4.1 Selection of instances from BBOB

As shown in Section 3.2, 90.3% of considered CMA-ES configurations have been identified as suffering from structural bias. Following the definition of SB (see Section 2.2), what this means is that the algorithm might be subjected to an internal 'force' acting on the evolving population and additionally steering solutions irrespective of the objective function, in a 'superpositional' manner. Using the stochastic objective function f_0 allows disentangling the SB 'force' from the landscape 'force' naturally acting on the evolving population since the latter is effectively averaged out. However, such disentanglement on runs on any functions other than f_0 is not straightforward due to the contribution of the landscape that cannot be easily averaged out.

To investigate the effect of Structural Bias on the performance of the optimisation algorithms, f_0 cannot be used, since performance cannot be meaningfully interpreted on this random function. Unfortunately, it is also impossible to directly compare algorithm configurations without bias and configurations with different types of SB because we cannot disentangle the effects of the different modules versus the effects of the different types of SB. To mitigate these issues we propose here an approach to compare the configurations to themselves. The proposed method relies on the BBOB mechanism of instance generation. This mechanism is claimed [10] to fully preserve landscape features and only modify the location and height of the optima. Therefore, a structurally biased algorithm running on maximally different problem instances will show performance differences that can only be caused by SB and not by differences in landscapes (given that the assumption of different instances preserving the landscape features holds). We suggest using the Euclidean distance between the location of the optima and the centre of the BBOB domain ($[-5, 5]^n$) as a measure of the difference between instances. Therefore, by examining 100 instances available per BBOB function with can find instances that have optima closest to and furthest away from the centre of the domain. To make results more robust we consider two additional instances: the 1^{st}, 2^{nd}, 99^{th} and 100^{th} closest to the centre. Results of such procedure are shown in Table 1.

4.2 Experimental setup

In this second experiment, a subset of 30 representative CMA-ES configurations with different kinds of structural bias (see Table 2) have been selected from the 1 620 configurations analysed in Section 3.2. These configurations have been run on 24 single-objective noiseless versions of BBOB functions in $n = 5$ dimensions, on 4 instances selected as described in Section 4.1, using 25 runs on each

Table 1: Instances selected from the first 100 intances per BBOB function in dimensionality 5 which have optima 1st, 2nd, 99th and 100th closest to the centre of domain, Euclidean distance per such instance (in colour, with 4 values of spectrum shown below) and maximal distance between extremal optima.

	1st	2nd	99th	100th	1st	2nd	99th	100th	100th-1st
	closest optimum to the centre of domain				distance to centre				distance
f_i	instance id								difference
1	85	54	40	74					5.00786680
2	93	84	91	68					5.23236165
3	7	38	51	87					4.87961205
4	7	38	51	87					4.87961205
5	0	72	36	99					0
6	14	21	78	79					4.52860968
7	93	90	35	21					4.76917961
8	92	79	5	18					3.50059649
9	63	21	19	97					9.1e-15
10	95	12	27	50					5.82548516
11	85	42	8	75					5.15158430
12	16	24	17	15					5.03646435
13	89	36	48	3					5.82278654
14	59	56	19	74					5.99024808
15	28	70	22	1					5.30984707
16	9	34	93	23					5.10054334
17	2	27	11	87					4.62649956
18	2	27	11	87					4.62649956
19	7	88	71	75					1.29e-14
20	0	72	36	99					0
21	80	85	9	91					6.02089024
22	79	48	8	84					5.66685170
23	64	81	19	22					5.80242661
24	0	72	36	99					0
			colour legend		0.0	3.73	7.45	11.2	

instance. The stopping criteria for each run is set to either $10\,000 \times n$ or attained precision of $< 10^{-8}$.

To evaluate the different CMA-ES configurations on the BBOB instances we use the Area Under the ECDF Curve as defined in Definition 4.1.

Definition 4.1 (Area Under the ECDF Curve, AUC). For a given optimisation algorithm A with a budget of B function evaluations for minimising a function $f : X \to \mathbb{R}$ and a given finite set of targets $\mathcal{V} \subset \mathbb{R}$, the AUC value of A on f is approximated by

$$\mathrm{AUC}(A, f, \mathcal{V}) = \int_1^B \widehat{F}(t; A, f, \mathcal{V}) \mathrm{d}t \ , \tag{2}$$

where

$$\widehat{F}(t; A, f, \mathcal{V}) = \frac{1}{N|\mathcal{V}|} \sum_{\phi \in \mathcal{V}} \sum_{i=1}^{N} \mathbb{1}(t_i(A, f, \phi) \leq t) \ , \tag{3}$$

and N is the number of runs for which we have performance logs, and $\mathbb{1}(t_i(A, f, \phi) \leq t)$ is an indicator function that returns 1 if the first hitting time of target ϕ in run i of A is not larger than t. If the target ϕ is not hit in this run, the indicator always returns 0.

Table 2: Module settings and predicted bias types for `centre` and `uniform` initialisation for the different configurations used in the location-dependent performance comparison.

Config id	weights option	step size adapt-n	SDIS	SB Type (`centre`)	SB Type (`uniform`)
1056	$\frac{1}{2}^{\lambda}$	csa	toroidal	bounds	centre
992	default	csa	uniform	bounds	bounds
1020	equal	csa	saturation	bounds	bounds
993	default	psr	uniform	centre	centre
994	default	csa	COTN	centre	none
995	default	psr	COTN	centre	centre
996	default	csa	toroidal	centre	centre
997	default	psr	toroidal	centre	clusters
1057	$\frac{1}{2}^{\lambda}$	psr	toroidal	centre	centre
1052	$\frac{1}{2}^{\lambda}$	csa	uniform	centre	centre
1055	$\frac{1}{2}^{\lambda}$	psr	COTN	centre	centre
1054	$\frac{1}{2}^{\lambda}$	csa	COTN	centre	none
1053	$\frac{1}{2}^{\lambda}$	psr	uniform	centre	centre
1025	equal	psr	COTN	centre	centre
1027	equal	psr	toroidal	clusters	clusters
1023	equal	psr	uniform	clusters	centre
990	default	csa	saturation	discr.	discr.
991	default	psr	saturation	discr.	discr.
1051	$\frac{1}{2}^{\lambda}$	psr	saturation	discr.	discr.
1050	$\frac{1}{2}^{\lambda}$	csa	saturation	discr.	discr.
1024	equal	csa	COTN	none	none
1028	equal	csa	mirror	none	none
1026	equal	csa	toroidal	none	none
1058	$\frac{1}{2}^{\lambda}$	csa	mirror	none	none
1022	equal	csa	uniform	none	none
1021	equal	psr	saturation	none	discr.
999	default	psr	mirror	none	none
998	default	csa	mirror	none	none
1029	equal	psr	mirror	none	none
1059	$\frac{1}{2}^{\lambda}$	psr	mirror	none	none

4.3 Analysis of results

In Figure 4 we show the results from two one-sided statistical tests on the performance of each configuration (25 runs per function instance) on two instances with the optimum closest to the centre of the domain and two instances with the optimum furthest from the centre of the domain. White dotted vertical lines separate the different types of SB in the configurations and white dashed horizontal lines indicate the BBOB functions that do not have a difference in the distance from optima to the centre. In the latter case, we would expect no significant performance difference between the instances. However, we can see that there are for some configurations significant performance differences even in those function ids, suggesting that the optimisation landscapes between different function instances are significantly different while the BBOB benchmark suite advertises them to be similar in behaviour.

In Figure 5 the AUC-based performance per run is visualised for f_5 and f_{22}. For these two functions, we can observe interesting behaviours of the configurations. For f_5, which is a linear slope function, we see that discretisation and unbiased configurations

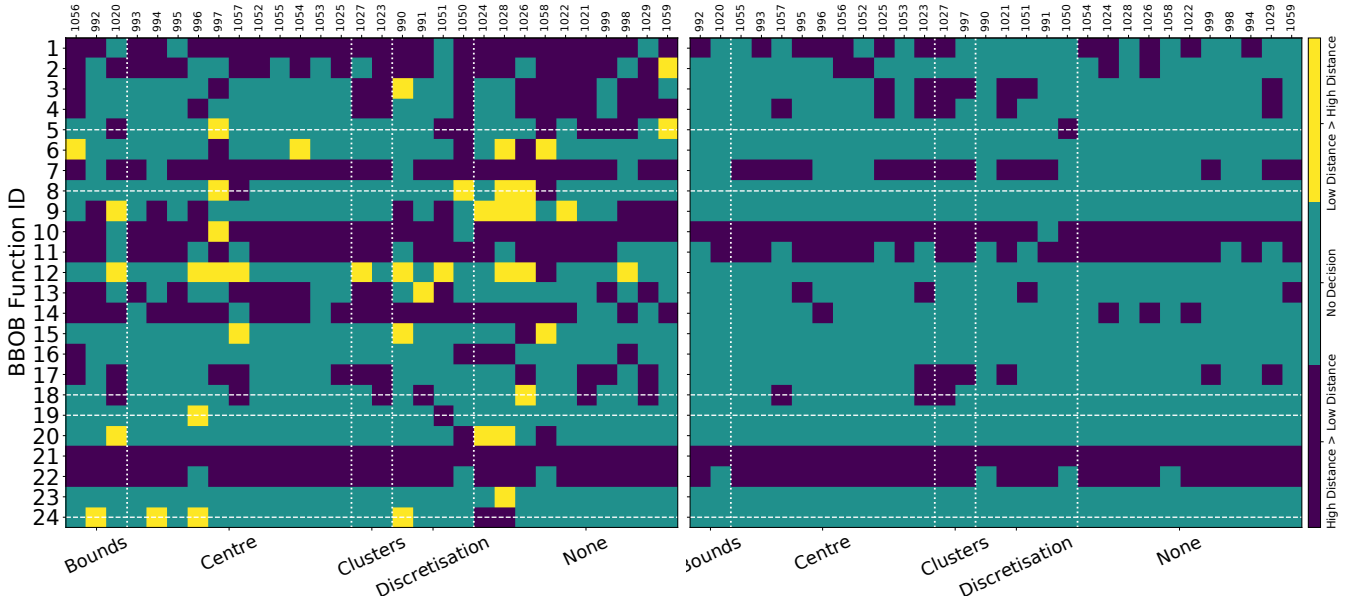

Figure 4: The configurations grouped by bias type are indicated on the X-axis and the BBOB function id is on the Y-axis. Each cell represents the results from two one-sided Wilcoxon rank-sum tests on the AUC-based performance (25 runs) of the two instances (1^{st} and 2^{nd}) with the optimum closest to the centre, versus the two instances with the optimum furthest from the centre (99^{th} and 100^{th}). Yellow indicates that the configuration performed significantly better on the 1^{st} and 2^{nd} instance, dark violet indicates significant worse performance on these instances ($\alpha = 0.01$, Bonferroni correction across all functions and configurations for each test direction). On the left diagram the experiment with `centre` initialisation is shown and on the right the one with `uniform` initialisation.

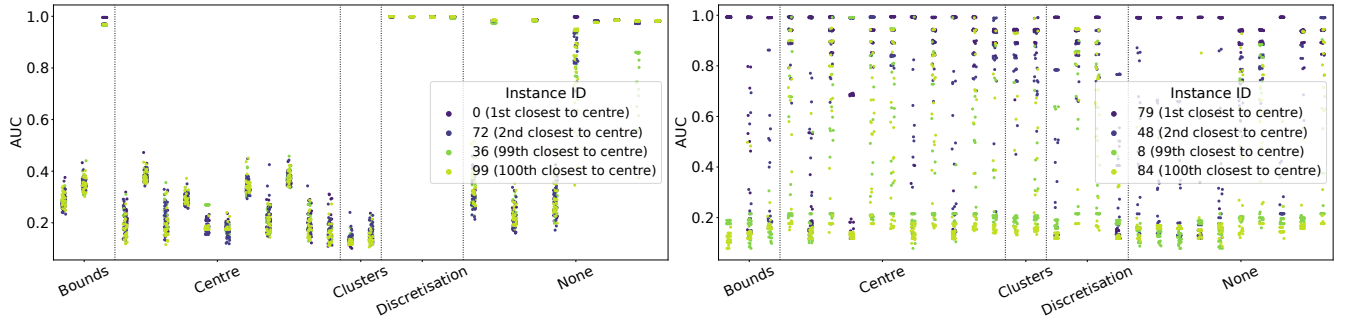

Figure 5: Performance results (AUC) for f_5 and f_{22} instances of the BBOB suite with 25 runs per configuration. Configurations are grouped by bias type on the X-axis. The instances with an optimum closest to the centre are shown in blue and the instances with an optimum furthest from the centre are shown in green.

perform better than configurations with Cluster and Centre bias. This makes sense as the f_5 function never has the optimum near the centre, and therefore centre bias would logically work against the search. For f_{22} one can observe that the instances that have an optimum close to the centre of the search space give better performance than those that have an optimum further away from the centre.

In general, it is hard to compare configurations with different bias types against each other, because these configurations also use different CMA-ES modules. Analysing which part of the performance difference comes from the different module settings and which part from having some type of SB is not easily possible. It is interesting to notice that even unbiased configurations of CMA-ES show many significant differences between the function instances, even more than biased configurations. This result shows that the assumption of instances preserving landscape features and thus being equally hard to solve *might not be valid* for all problems. We can however not conclude that centre bias configurations perform significantly better on instances with their optimum close to the centre, so the effect of the structural bias seems to not affect the performance of the algorithm strongly enough in this experiment.

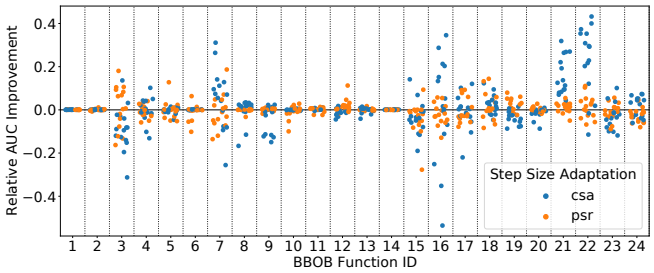

Figure 6: Relative improvement in AUC for each selected configuration when using `centre` initialisation instead of `uniform` initialisation, for each BBOB function.

Having collected performance data for two different 'centre of mass' initialisation methods lets us consider the relative improvement in AUC achieved for each configuration when switching from the random initialisation to the one using the centre of the search space. This relative improvement is shown in Figure 6, where we can see that the impact of initialisation differs rather significantly between functions. The figure also highlights the interplay between the used step-size adaptation mechanism and initialisation, where we note that the population success rule is more severely impacted by this change than the cumulative step-size adaptation.

5 CONCLUSIONS

1 620 configurations of commonly used module settings of the very popular CMA-ES algorithm have been analysed using several experiments. A large portion (90.3%) of these configurations shows the structural bias of various types. While the type and strength of bias seems to be largely impacted by the used SDIS, it is important to note that this module is an integral component of the CMA-ES implementation necessary to handle constraints. The strength of Bias also seems to be correlated with the fraction of infeasible solutions generated by the algorithm configurations. This is especially the case for discretisation bias (which is mostly caused by the saturation boundary correction strategy), which has the bias strength positively correlated with the fraction of infeasible solutions. centre bias on the other hand, which is very common for many different configurations of CMA-ES, shows a negative correlation between the strength of bias and the fraction of infeasible solutions. This makes perfect sense since strongly biased configurations towards the centre of the search space rarely generate candidate solutions that are outside the bounds. This partly confirms the validity of the heuristic introduced to measure the strength of SB and the predicted types of bias using the BIAS toolbox.

In addition to these findings, we also show how much each module setting affects the presence and strength of SB in most overall configurations. This is done using both the recommended initialisation around the centre of the search space and with a uniform sampling initialisation. It is interesting to note that using the recommended initialisation structural bias is more present than when using uniform sampling.

In an attempt to investigate the effect of structural bias on the performance of these algorithms an additional experiment is carried out using specific function instances that have their optimum either far away or close to the centre of the search space. We observed that there are many significant differences in performance between the different function instances. This indicated that the different function instances have a significantly different function landscape which is not expected as BBOB advertises these function instances as comparable behaviour. These significant differences even occur when the optimum is similarly positioned with respect to the centre of the search space. Unfortunately, there are no clear patterns that prove or disprove the efficiency of biased versus unbiased configurations.

Reproducibility To ensure reproducibility, we provide our source code and several intermediary artefacts in our figshare repository [23]. Additionally, this repository contains a large number of extra figures, among them the equivalent of Figure 5 for all other functions and initialisation methods, and the distribution and exact test failures for all 3 240 configurations run on f_0.

REFERENCES

[1] Claus Aranha, Christian L. Camacho Villalón, Felipe Campelo, Marco Dorigo, Rubén Ruiz, Marc Sevaux, Kenneth Sörensen, and Thomas Stützle. 2021. Metaphor-based metaheuristics, a call for action: the elephant in the room. *Swarm Intelligence* (2021). https://doi.org/10.1007/s11721-021-00202-9

[2] Fabio Caraffini and Anna V. Kononova. 2019. Structural bias in differential evolution: A preliminary study. *AIP Conference Proceedings* 2070, 1 (2019), 020005. https://doi.org/10.1063/1.5089972

[3] Fabio Caraffini, Anna V. Kononova, and David W. Corne. 2019. Infeasibility and structural bias in differential evolution. *Information Sciences* 496 (2019), 161–179. https://doi.org/10.1016/j.ins.2019.05.019

[4] Jacob de Nobel, Diederick Vermetten, Hao Wang, Carola Doerr, and Thomas Bäck. 2021. *Tuning as a Means of Assessing the Benefits of New Ideas in Interplay with Existing Algorithmic Modules.* Association for Computing Machinery, New York, NY, USA, 1375–1384. https://doi.org/10.1145/3449726.3463167

[5] Carola Doerr, Hao Wang, Furong Ye, Sander van Rijn, and Thomas Bäck. 2018. IOHprofiler: A benchmarking and profiling tool for iterative optimization heuristics. *arXiv preprint arXiv:1810.05281* (2018).

[6] Tome Eftimov, Gorjan Popovski, Quentin Renau, Peter Korosec, and Carola Doerr. 2020. Linear Matrix Factorization Embeddings for Single-objective Optimization Landscapes. In *2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020, Canberra, Australia, December 1-4, 2020.* IEEE, 775–782. https://doi.org/10.1109/SSCI47803.2020.9308180

[7] Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. 2010. *Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions.* Technical Report. INRIA.

[8] N. Hansen. 2006. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, J.A. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea (Eds.). Springer, 75–102.

[9] Nikolaus Hansen. 2016. The CMA Evolution Strategy: A Tutorial. https://doi.org/10.48550/ARXIV.1604.00772

[10] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. 2021. COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software* 36, 1 (2021), 114–144.

[11] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation* 9, 2 (2001), 159–195. https://doi.org/10.1162/106365601750190398

[12] Alfred Inselberg. 1985. The plane with parallel coordinates. *The visual computer* 1, 2 (1985), 69–91. https://doi.org/10.1007/BF01898350

[13] Pascal Kerschke and Heike Trautmann. 2019. Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning. *Evol. Comput.* 27, 1 (mar 2019), 99–127. https://doi.org/10.1162/evco_a_00236

[14] Anna V. Kononova, Fabio Caraffini, and Thomas Bäck. 2021. Differential evolution outside the box. *Information Sciences* 581 (2021), 587–604. https://doi.org/10.1016/j.ins.2021.09.058

[15] Anna V. Kononova, Fabio Caraffini, Hao Wang, and Thomas Bäck. 2020. Can Compact Optimisation Algorithms Be Structurally Biased?. In *Parallel Problem Solving from Nature – PPSN XVI*, T. Bäck, M. Preuss, A. Deutz, H. Wang, C. Doerr, M. Emmerich, and H. Trautmann (Eds.). Springer International Publishing, Cham, 229–242. https://doi.org/10.1007/978-3-030-58112-1_16

[16] Anna V. Kononova, Fabio Caraffini, Hao Wang, and Thomas Bäck. 2020. Can Single Solution Optimisation Methods Be Structurally Biased?. In *2020 IEEE Congress on Evolutionary Computation (CEC).* IEEE, Glasgow, 1–9. https://doi.

org/10.1109/CEC48606.2020.9185494

[17] Anna V. Kononova, David W. Corne, Philippe De Wilde, Vsevolod Shneer, and Fabio Caraffini. 2015. Structural bias in population-based algorithms. *Information Sciences* 298 (2015), 468–490. https://doi.org/10.1016/j.ins.2014.11.035

[18] Mario A. Munoz, Michael Kirley, and Saman K. Halgamuge. 2015. Exploratory Landscape Analysis of Continuous Space Optimization Problems Using Information Content. *Trans. Evol. Comp* 19, 1 (feb 2015), 74–87. https://doi.org/10.1109/TEVC.2014.2302006

[19] Mario A. Munoz, Michael Kirley, and Kate Smith-Miles. 2021. Analyzing randomness effects on the reliability of exploratory landscape analysis. *Natural Computing* (2021). https://doi.org/110.1007/s11047-021-09847-1

[20] Quentin Renau, Johann Dréo, Carola Doerr, and Benjamin Doerr. 2021. Towards Explainable Exploratory Landscape Analysis: Extreme Feature Selection for Classifying BBOB Functions. In *Applications of Evolutionary Computation - 24th International Conference, EvoApplications 2021, Held as Part of EvoStar 2021, Virtual Event, April 7-9, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 12694)*, Pedro A. Castillo and Juan Luis Jiménez Laredo (Eds.). Springer, 17–33. https://doi.org/10.1007/978-3-030-72699-7_2

[21] Sander van Rijn, Hao Wang, Bas van Stein, and Thomas Bäck. 2017. Algorithm configuration data mining for cma evolution strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 737–744.

[22] Bas van Stein, Fabio Caraffini, and Anna V. Kononova. 2021. Emergence of Structural Bias in Differential Evolution. In *Proceedings of the 2021 Genetic and Evolutionary Computation Conference Companion* (Lille, France) *(GECCO '21 Companion)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3449726.3463223

[23] Diederick Vermetten, Anna V. Kononova, Fabio Caraffini, and Bas van Stein. 2022. Using Structural Bias to Analyse the Behavior of Modular CMA-ES - Figures. (2022). https://doi.org/10.6084/m9.figshare.19578991.v1

[24] Diederick Vermetten, Anna V. Kononova, Fabio Caraffini, Hao Wang, and Thomas Bäck. 2021. Is There Anisotropy in Structural Bias?. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Lille, France) *(GECCO '21)*. Association for Computing Machinery, New York, NY, USA, 1243–1250. https://doi.org/10.1145/3449726.3463218

[25] Diederick Vermetten, Bas van Stein, Anna V Kononova, and Fabio Caraffini. 2022. Analysis of Structural Bias in Differential Evolution Configurations. In *Differential Evolution: From Theory to Practice*. Springer, 1–22.

[26] Diederick Vermetten, Bas van Stein, Fabio Caraffini, Leandro Minku, and Anna V. Kononova. 2021. BIAS: A Toolbox for Benchmarking Structural Bias in the Continuous Domain. https://doi.org/10.36227/techrxiv.16594880.v1

[27] Diederick Vermetten, Hao Wang, Thomas Bäck, and Carola Doerr. 2020. Towards Dynamic Algorithm Selection for Numerical Black-Box Optimization: Investigating BBOB as a Use Case. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (Cancún, Mexico) *(GECCO '20)*. Association for Computing Machinery, New York, NY, USA, 654–662.

[28] Hao Wang, Diederick Vermetten, Furong Ye, Carola Doerr, and Thomas Bäck. 2022. IOHanalyzer: Detailed Performance Analyses for Iterative Optimization Heuristics. *ACM Transactions on Evolutionary Learning and Optimization* 2, 1, Article 3 (apr 2022), 29 pages. https://doi.org/10.1145/3510426