Deep BIAS: detecting structural bias using explainable AI

Stein, N. van; Vermetten, D.L.; Caraffini, F.; Kononova, A.V.

Deep-BIAS: Detecting Structural Bias using Explainable AI

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

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

Fabio Caraffini
Swansea University
Swansea, UK
fabio.caraffini@swansea.ac.uk

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

ABSTRACT

Evaluating the performance of heuristic optimisation algorithms is essential to determine how well they perform under various conditions. Recently, the BIAS toolbox was introduced as a behaviour benchmark to detect structural bias (SB) in search algorithms. The toolbox can be used to identify biases in existing algorithms, as well as to test for bias in newly developed algorithms. In this article, we introduce a novel and explainable deep-learning expansion of the BIAS toolbox, called Deep-BIAS. Where the original toolbox uses 39 statistical tests and a Random Forest model to predict the existence and type of SB, the Deep-BIAS method uses a trained deep-learning model to immediately detect the strength and type of SB based on the raw performance distributions. Through a series of experiments with a variety of structurally biased scenarios, we demonstrate the effectiveness of Deep-BIAS. We also present the results of using the toolbox on 336 state-of-the-art optimisation algorithms, which showed the presence of various types of structural bias, particularly towards the centre of the objective space or exhibiting discretisation behaviour. The Deep-BIAS method outperforms the BIAS toolbox both in detecting bias and for classifying the type of SB. Furthermore, explanations can be derived using XAI techniques.

CCS CONCEPTS

• **Computing methodologies** → **Continuous space search**; • **Theory of computation** → **Design and analysis of algorithms**.

KEYWORDS

Structural Bias, Algorithm Behaviour, Explainable AI, Optimisation

ACM Reference Format:
Bas van Stein, Diederick Vermetten, Fabio Caraffini, and Anna V. Kononova. 2023. Deep-BIAS: Detecting Structural Bias using Explainable AI. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3583133.3590551

*Corresponding author

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0120-7/23/07.
https://doi.org/10.1145/3583133.3590551

1 INTRODUCTION

As the amount of data and complexity of the optimisation problems continue to increase, the demand for effective heuristic optimisation algorithms also increases. Since no single heuristic algorithm is universally best [16], it is necessary to benchmark these algorithms to understand which performs best under specific conditions. Most continuous optimisation benchmarks are performance-based, such as the Black-Box Optimisation Benchmark (BBOB) [3] test suite. These benchmarks provide information on the performance of an algorithm and how it compares with others in various situations. For example, one algorithm may excel at optimising separable functions while another performs well on uni-modal, highly conditioned functions. Resource-based benchmarks measure the number of resources (computation power, memory, and energy) required under certain conditions, but do not offer much insight into the behaviour of the algorithms under different circumstances. Behaviour-based benchmarks, on the other hand, provide additional information about how an algorithm behaves under different conditions, such as the movement of a population of candidate solutions in a swarm-based optimisation algorithm.

The previously proposed BIAS toolbox [14] is such a behaviour-based benchmarking tool. It can be used to analyse whether algorithms or components of an algorithm induce structural bias (SB). SB is a type of bias inherent in iterative heuristic optimisation algorithms that affects their performance in the objective space. By detecting the presence, strength, and type of SB in a heuristic optimisation algorithm, it is possible to identify areas for improvement and understand under which conditions SB is less likely to occur. This information can be used to optimise the performance of these algorithms.

In this work, we propose an improved methodology for the BIAS toolbox for both the detection of structural bias and the classification of the type of SB. Instead of using 39 statistical tests and their p-values, we propose using a single convolutional deep learning model to predict the presence and type of SB in the raw final-point distributions of various optimisation runs on a special test function f_0. In this work, we propose a deep learning approach on raw algorithm performance data (on a special test function) to identify which

SB type (if any) is most likely to occur in a given optimiser. The complete SB benchmark deep learning test suite (Deep-BIAS) and data generators for different SB scenarios are made open-source [12]. In addition, we propose to use explainable artificial intelligence techniques adapted for the proposed deep learning approach to visualise and analyse the SB results. We evaluate both the proposed deep learning approach and the previous statistical test approach using a large set of 189 different artificially generated parameterised distributions containing 11 different scenarios of structural bias. In addition, we compare the results of both methods on a wide set of state-of-the-art optimisers.

2 STRUCTURAL BIAS AND RELATED WORK

In many complex optimisation problems, there is no a priori information about which regions of the space contain good solutions. In such a setting, the search has to start "from scratch", within the defined domain boundaries. The search should then be able to identify and progress towards promising regions with good values of objective function. Only points sampled thus far should steer algorithm's logic and operators in the subsequent steps of the search. This means that the algorithm does not inherently favour one region of the space over others. In other words, if the algorithm is to be deployed in a *general situation*, it should be able to find high-quality solutions regardless of where they are located inside the feasible domain of the problem. The degree to which an algorithm exhibits such flexibility in locating optima is clearly among the reasons for its success.

Unfortunately, detecting the propensity of an iterative algorithm towards some parts of the domain is difficult due to the interplay between the sampled landscape of the function and the internal workings of the algorithm [6]. In order to disentangle these two components, the test function f_0 has been defined as follows:

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

For this function, the optimum is located uniformly at random throughout the domain. As such, an unbiased search is expected to return a uniform distribution of final best solutions. If the distribution of these points is non-uniform, this indicates a *structural bias* of the algorithm. Structural bias thus represents an algorithm's inflexibility. Because of this, we consider structural bias to be an undesired behaviour which potentially limits the algorithm's performance in a generic setting. The exact relation between the operators of an algorithm and its structural bias and the way it might influence performance on different function landscapes is poorly understood. However, through testing on f_0, SB can be identified and thus potentially removed via a prudent choice of operators.

Visual inspection of the distributions of the final solutions found in f_0, collected in multiple independent runs of the method under investigation, and displayed component-wise is the most intuitive approach to detecting SB. However, such a procedure can be subject to personal interpretations and is time-consuming when a large number of images need to be generated and inspected (see [2, 6, 9] and repositories [1, 10]). Moreover, it is unable to provide reliable results in the presence of mild SB or figure-rendering artefacts.

The use of statistical testing methods removes the subjective component of SB inspection and leads to an automated decision-making process over a large data set of results, where the distributions of the final solutions obtained with multiple runs in f_0 are tested for uniformity. The best results are obtained with $N = 600$, which helps to detect SB more often, but does not guarantee the detection of all different types of SB at any significance level [9] - even higher values of N are needed for smaller levels of significance and higher statistical power [5]. Using multiple statistical tests to detect SB and its type reliably can be complex and laborious. From the point of view of a practitioner or an algorithm designer, these processes should be automated.

BIAS [15] is an open-source Python package, available from [13], to benchmark SB in the continuous domain. The toolbox provides an SB detection mechanism based on the aggregation of the results of 39 statistical tests and a Random Forest model to identify the type of structural bias. It furthermore contains a data generator to sample data from a set of scenarios producing synthetic results; a component producing the parallel coordinate plots of the final best positions to display SB and those reporting the outcome of the decisions made with statistical analysis while detecting SB.

3 METHODOLOGY

The method proposed here extends the functionality of BIAS. It uses a deep learning-based model to predict the presence and type of structural bias. The model is trained on a large portfolio of bias scenarios and is optimised using AutoKeras [4]. In this section, we first provide an overview of the scenarios and the generator used to train the model. Then, the model training and testing procedures are explained in detail.

Portfolio of scenarios. The proposed Deep-BIAS method is based on the parameterised SB scenarios proposed in [14]. This portfolio of scenarios includes the most common types of structural bias as observed in previous studies, including

- bias towards the centre of the search space,
- bias towards the bounds of the search space,
- bias towards certain parts of the search space forming clusters,
- bias towards avoiding certain parts of the search space, creating gaps and
- strong discretisation.

In the original paper, there are 11 different scenarios with different parameter settings, giving a total set of 194 parameterized scenarios. After visual and analytical analysis of the parameters of these scenarios, we removed 5 of these parameter settings, as they frequently generated distributions visually and statistically equivalent to random uniform distributions using only 600 samples. This leaves a total of 189 parameterized distribution generators (still 11 scenarios), which we use to generate train and test data sets.

To train the deep learning model, we divide the 11 scenarios into four classes, *Center, Bounds, Gaps/Clusters* and *Discretisation*. We argue that gaps and cluster bias are highly overlapping, as you have gaps in the search space when there are clusters and vice versa. Therefore, they are added together under one class label. For each class 20.000 distributions are generated, equally divided over the parameter settings and scenarios that belong to each class label. For the uniform (no bias) label, we generated 80.000 distributions

to make the prediction task balanced between bias and non-bias samples. In total, this gives a data set of 160.000 one-dimensional distributions, of which 80% is used as training and 20% as final validation (using stratified sampling).

Optimised Convolutional Neural Network. In the next step, a deep one-dimensional convolutional neural network (1d CNN) is trained and optimised using the AutoKeras [4] algorithm. Each distribution is first ordered in ascending order of the values and then used as input for the 1d CNN. AutoKeras is set to run with an evaluation budget of 100 trials and without image augmentation and pre-processing (this would have no meaning in our situation since we are not dealing with images). Each network is trained for 50 epochs and uses a small randomly selected validation set (from the original training data) to compare different models with each other. The best neural network instance is stored and used in this work. The general architecture consists of two blocks, each containing two convolutional layers followed by a max pooling layer. The second block is half the size of the first. Followed by a dense layer and a SoftMax classification head. In total, there are four models, one for each sample size of 30, 50, 100 and 600.

Predicting SB and SB type. The optimised CNN models can now be used to directly predict any distribution for the presence and type of structural bias.

To analyse the misclassifications of each of the models, we used the explainable artificial intelligence (XAI) method, SHAP [8], which approximates Shapley values [7] based on a background sample set from training data. Using the Shapley values for each point in the distribution, it is possible to highlight regions of interest for a particular prediction. In Figure 1, an example of a prediction is shown that does not match the ground truth. This example comes from the 600 sample size model. It is important to note that there is a high level of randomness in many scenario generators. It can therefore occur that clusters are very much overlapping, creating a uniformly distributed sample, or that clusters can be located in the centre or at the bounds, which deceives the classifier. Additional examples can be seen in the full paper [11].

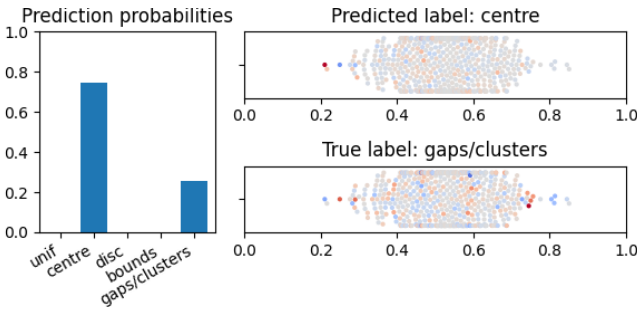

Figure 1: Example of (wrongly) classified samples due to overlap in classes. Clusters can be located by chance on the bounds or in the centre of the space. Colours indicate Shapley values where dark red indicates a positive contribution towards the class label (either predicted (up) or ground truth (below)), and dark blue indicates a negative contribution. Sample points with similar values are stacked on top of each other.

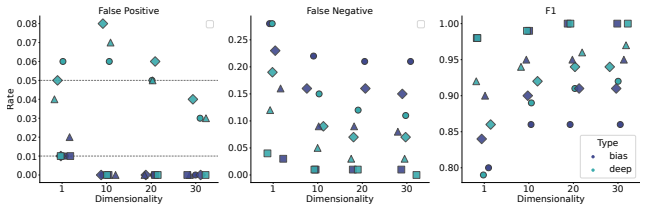

Figure 2: Comparison (with $\alpha = 0.01$) of the original BIAS toolbox (blue) and the Deep-BIAS (teal) in terms of false positives (left), false negatives (middle) and F1-score (right). On all figures, markers identify the used sample size: ○, ◇, △ and □ are 30, 50, 100 and 600, respectively.

Next, the Deep-BIAS method is compared to the original BIAS toolbox. The comparison is done by transforming the problem into a binary problem to detect SB. Since the original BIAS toolbox is also validated in this way on a different number of dimensions (not just one-dimensional distributions), the same experiment is repeated here. The following dimensions are evaluated: 1, 10, 20 and 30. For each of these dimensionalities (d), a test set of 1890 biased and 1890 unbiased d-dimensional distributions is generated. These distributions are then predicted by both the original BIAS toolbox and the newly proposed Deep-BIAS method. For the proposed method, the class probabilities are averaged over all dimensions to give a final prediction per d-dimensional distribution. We define a configuration as biased if at least 10% of its dimensions are classified as non-uniform. This threshold is chosen to remain consistent with the original BIAS toolbox [14].

The results of this experiment are shown in Figure 2, where false positives, false negatives, and F1 scores are compared for different dimensions and sample sizes. We can see that, while Deep-BIAS has a slightly higher rate of false positives, this is compensated by a significantly lower false negative rate. In general, the F1 score for Deep-BIAS is slightly higher, indicating that it slightly outperforms the statistical approach.

4 BENCHMARKING STRUCTURAL BIAS

This section benchmarks a large set of heuristic optimisation algorithms by applying the BIAS toolbox.

We use data from a heterogeneous pool of heuristics executed on f_0 at dimensionality $n = 30$ for a maximum of $10000 \cdot n$ fitness functional calls. In total, we consider 336 optimisation heuristics that fall into the following categories (all use $N = 100$):

- Variants of Differential Evolution (195 configurations), See the full paper [11].
- Compact optimisation algorithms (81 configurations),
- Single-solution algorithms (60 configurations),

For the sake of clarity and reproducibility, the exact composition and setup of these algorithmic configurations are fully described in a dedicated document available from [13].

First, we compare the decisions made by the Deep-BIAS method with those of the original toolbox on the set of single-solution algorithms. This is done by showing the class probabilities of both methods for each algorithm in Figure 3. From this figure, we see that, in most cases, both methods give the same biased/non-biased

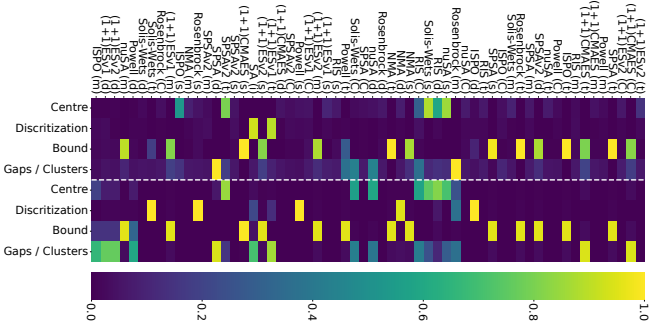

Figure 3: Predictions of types of non-uniformity made by the Deep-BIAS on single solution algorithms where the two methods give a different biased / non-biased outcome. For the original method, the probabilities are determined by the random forest model when the configuration is considered biased and set to 0 otherwise. For Deep-BIAS, the predictions are the average of the per-dimension predictions.

outcome. However, the type of bias detected varies for most of the algorithms. This often occurs when the random forest (RF) model of the original toolbox predicts 'clusters'. This might be due in part to the differences in training data for the two models, combined with the fact that for some cluster settings, the clusters might be located near the bounds, making the distinction between these two classes somewhat fuzzy.

For Figure 3, it is also important to note that for the RF model, the probabilities of the class sum up to 1 by design, which is not the case for the deep model, as the uniform class still gets some of the probability mass. Thus, the outcome of Deep-BIAS gives indirectly a measure of the strength of the bias.

5 CONCLUSION

In light of the analysis performed in this study, we conclude that the use of deep learning is a viable option to detect SB with satisfactory results. With only 50 samples, we can correctly detect most uniformly distributed points, and performances increase with larger sample sizes. We find that the optimal network architecture for detecting SB is not as complex as those often designed in other deep learning applications in the literature, and yet it behaves similarly to the original BIAS toolbox, which is based on statistical tests, and performs very well in terms of the F1 score evaluation metric.

Compared to BIAS, the proposed Deep-BIAS alternative displays some interesting features and advantageous behaviours.

- It outperforms the statistical test-based approach in classifying the type of SB.
- It gives a better measure of strength of the SB by using the class probabilities.
- It provides additional insights by using XAI, where regions of interest in distributions can be further analysed to understand the mechanism behind the classifications.

A disadvantage of using a neural model is that the resulting SB detection system is less generalisable than that obtained with a statistical test approach, where the network might miss SB types that are not considered in the training data. However, the training data set prepared for Deep-BIAS appears to be adequate, and the network fails to find bias mainly when the saturate constraint handling method is used. This is not a problem for BIAS. For these reasons, the overarching conclusion of this study envisages the joint use of the 2 systems for optimal SB detection. We recommend using BIAS as the primary method for binary bias/non-bias classification and Deep-BIAS to inspect the type of SB and determine its strength.

REFERENCES

[1] Fabio Caraffini and Anna V. Kononova. 2021. Structural Bias in Optimisation Algorithms: Extended Results. https://doi.org/10.17632/zdh2phb3b4.2
[2] 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
[3] 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.
[4] Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1946–1956.
[5] S. S. Kar and A. Ramalingam. 2013. Is 30 the Magic Number? Issues in Sample Size Estimation. *National Journal of Community Medicine* 4, 1 (2013), 175–179.
[6] 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
[7] L. Shapley. 1953. A Value for n-Person Games. In *Contributions to the Theory of Games II*, H. Kuhn and A. Tucker (Eds.). Princeton University Press, 307–317. https://doi.org/10.1515/9781400881970-018
[8] Tjeerd van Campen, Herbert Hamers, Bart Husslage, and Roy Lindelauf. 2018. A new approximation method for the Shapley value applied to the WTC 9/11 terrorist attack. *Social Network Analysis and Mining* 8, 1 (2018), 1–12.
[9] 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
[10] Bas van Stein, Fabio Caraffini, and Anna V. Kononova. 2021. Emergence of Structural Bias in Differential Evolution - Source code & extended graphical results. https://doi.org/10.17632/pb2bdp2gkp.1
[11] Bas van Stein, Diederick Vermetten, Fabio Caraffini, and Anna V. Kononova. 2023. Deep-BIAS: Detecting Structural Bias using Explainable AI. arXiv:2304.01869 [cs.NE]
[12] Bas van Stein, Diederick Vermetten, Fabio Caraffini, and Anna Kononova V. 2023. *Deep-BIAS v1.0.0.* https://doi.org/10.5281/zenodo.7614586
[13] Diederick Vermetten, Anna V. Kononova, Fabio Caraffini, Bas van Stein, and Leandro Minku. 2021. BIAS: A Toolbox for Benchmarking Structural Bias in the Continuous Domain – Code. https://doi.org/10.6084/m9.figshare.16546245
[14] Diederick Vermetten, Bas van Stein, Fabio Caraffini, Leandro L. Minku, and Anna V. Kononova. 2022. BIAS: A Toolbox for Benchmarking Structural Bias in the Continuous Domain. *IEEE Transactions on Evolutionary Computation* 26, 6 (2022), 1380–1393. https://doi.org/10.1109/TEVC.2022.3189848
[15] Diederick Vermetten, Bas van Stein, Fabio Caraffini, Leandro L. Minku, and Anna V. Kononova. 2022. BIAS: A Toolbox for Benchmarking Structural Bias in the Continuous Domain. *IEEE Transactions on Evolutionary Computation* 26, 6 (2022), 1380–1393. https://doi.org/10.1109/TEVC.2022.3189848
[16] D. Wolpert and W. Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1 (1997), 67–82. Issue 1. https://doi.org/10.1109/4235.585893