A Black-Box Discrete Optimization Benchmarking (BB-DOB) Pipeline Survey: Taxonomy, Evaluation, and Ranking

Aleš Zamuda
University of Maribor
Maribor, Slovenia
ales.zamuda@um.si

Miguel Nicolau
University College Dublin
Dublin, Ireland
miguel.nicolau@ucd.ie

Christine Zarges
Aberystwyth University
Aberystwyth, United Kingdom
c.zarges@aber.ac.uk

ABSTRACT

This paper provides a taxonomical identification survey of classes in discrete optimization challenges that can be found in the literature including a proposed pipeline for benchmarking, inspired by previous computational optimization competitions. Thereby, a Black-Box Discrete Optimization Benchmarking (BB-DOB) perspective is presented for the BB-DOB@GECCO Workshop. It is motivated why certain classes together with their properties (like deception and separability or toy problem label) should be included in the perspective. Moreover, guidelines on how to select significant instances within these classes, the design of experiments setup, performance measures, and presentation methods and formats are discussed.

CCS CONCEPTS

• **Mathematics of computing** → **Evolutionary algorithms**; *Bio-inspired optimization*; Nonparametric statistics; • **Computing methodologies** → **Ranking**; **Planning under uncertainty**; *Continuous space search*; • **Theory of computation** → **Nonconvex optimization**; **Bio-inspired optimization**; *Stochastic control and optimization*; • **General and reference** → *Evaluation*; *Performance*; • **Information systems** → *Expert systems*; Data mining; • **Applied computing** → Decision analysis;

KEYWORDS

optimization, benchmarking, discrete optimization, black-box benchmark, ranking, computational intelligence

ACM Reference Format:
Aleš Zamuda, Miguel Nicolau, and Christine Zarges. 2018. A Black-Box Discrete Optimization Benchmarking (BB-DOB) Pipeline Survey: Taxonomy, Evaluation, and Ranking. In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3205651.3208307

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07…$15.00
https://doi.org/10.1145/3205651.3208307

1 INTRODUCTION

This paper presents a perspective on the Black-Box Discrete Optimization Benchmarking Workshop (BB-DOB@GECCO)[1], which is a part of the Genetic and Evolutionary Computation Conference (GECCO) 2018. It delivers a taxonomical identification survey of discrete optimization challenges that can be found in the literature, followed by a proposed pipeline for benchmarking, which is inspired by previous computational optimization competitions in continuous settings that used test functions for optimization application domains [64]: single-objective optimization [33, 34, 36, 53], constrained optimization [35, 41, 63], multi-modal optimization [48, 49], multi-objective optimization [3, 10, 11, 24, 73], target value optimization (black-box) [20, 21], noisy optimization [22], bi-objective optimization [57], many objective [32], large-scale optimization [54, 55], dynamic optimization [3, 31], real world [9], computationally expensive [8, 38], learning-based [34], and similar.

Previous approaches using specific algorithms to tackle these optimization application domains at the mentioned competitions include, e. g., [6, 7, 65, 67–70]. Alongside the presented algorithms, the authors of the optimization competition papers have usually also created frameworks for the development of experiments in order to produce algorithm candidates submitted to these competitions and subsequent journal publications [5, 72]. These frameworks range from early awk/sed based unpublished scripts as e.g. for [68], to public frameworks like jMetal [14], irace [40], COCO [20], etc. After producing results from the test functions, these had to be aggregated by means of statistics, e.g., non-parametric tests and Friedman ranking [18]. However, there is a lack of papers on how to prepare a benchmark for BB-DOB that would yield black-box algorithms that operate suitably well over different classes of discrete functions.

In the next section, a taxonomic list of classes for discrete optimization follows. In Section 3, some properties and usage for these classes are provided. In Section 4, a methodology for selecting significant instances is proposed. In Sections 5, 6, and 7, the benchmark pipeline is further constructed by suggesting experiments setup, performance measures, and formats for the presentation of results, which build upon the explanations provided for the proposed instances for benchmarking. In Section 8, conclusions are presented together with some opportunities for future work.

2 TAXONOMICAL CLASSES IDENTIFICATION SURVEY

In this section, the subsections list different classes for discrete optimization functions through five perspectives, including modality,

[1] http://iao.hfuu.edu.cn/bbdob-gecco18

programming representations, real-world challenges modeling, and budget planning.

2.1 Modality: Unimodal, Bimodal, and Multimodal Discrete Functions

We discuss pseudo-Boolean optimization problems, e. g., fitness functions of the form $f: \{0,1\}^n \rightarrow \mathbb{R}$ as an example and for reasons of simplicity only consider maximization.

In this context, we call a search point x^* a local optimum if for all x with $H(x^*, x) = 1$ (i.e., the direct Hamming neighbors of x^*), $f(x^*) \geq f(x)$ holds. A problem f is called *unimodal* if and only if there is a unique local optimum (which in turn is also the unique global optimum). A problem f is called *weakly unimodal* if all its local optima have the same fitness. Otherwise, the problem is called multimodal.

It is well known that for the (1+1) EA using standard bit mutations (SBM) OneMax is an easiest function (toy problem) with a unique optimum [13] while Trap is a hardest [23] (and also deceptive). An often considered special case of multimodal problems are bimodal problems, which have exactly two local optima. Here, a classic example is TwoMax [45], which consists of two symmetric branches (i. e., OneMax and ZeroMax) and has its two local optima as far away from each other as possible (i.e., in the all-zero and all-ones bit string, respectively).

A generalization of TwoMax to an arbitrary number of local optima has been considered [27] by taking inspiration from similar benchmark functions in continuous optimization [52]. This multimodal function class comes with the hope of capturing relevant and controllable problem features (parameters) by still being accessible from a theoretical perspective.

A large number of different pseudo-Boolean example functions has been considered in the literature. However, a complete overview is out of scope for this paper and left for future work. We remark that combinatorial optimization problems are usually multimodal and should therefore also be considered in this context. Moreover, similar considerations should be made for problems that do not have a fixed problem dimension.

Finally, pseudo-Boolean optimization problems can be categorized according to other general functions classes and properties and some of the above fall into several of these categories. Examples include linear functions (the function value for a search point is computed as a weighted sum of the values of its bits; OneMax), monotone functions (functions where a mutation flipping at least one 0-bit into a 1-bit and no 1-bit into a 0-bit strictly increases the function value; OneMax), functions of unitation (the fitness only depends on the number of 1-bits in the considered search point; OneMax, TwoMax), and separable functions (the fitness can be expressed as a sum of subfunctions that depend on mutually disjoint sets of bits of the search points; OneMax, TwoMax). It should be noted that relationships between these classes can be established. For example, the class of monotone functions includes linear functions where all weights are positive. Moreover, all monotone functions are unimodal. Such a categorization is also subject to future work.

2.2 Genetic Programming

Genetic Programming (GP) [30] has seen a recent effort towards standardization of benchmarks, particularly in the application area of Symbolic Regression and Classification [42, 62]. These have been mostly artificial problems: a function is provided, which allows the generation of input-output pairs for regression. Some of the most commonly used in recent GP literature include the sets defined by Keijzer [28] (15 functions), Pagie [44] (1 function), Korns [29] (15 functions) and Vladislavleva [61] (8 functions).

Not all proposed functions are suitable as benchmarks, however, and a lack of guidelines for benchmark application results in incomparable results between approaches. A recent study has highlighted this [43], and proposed guidelines for improving benchmarking in GP (see Table 1 for functions used to generate datasets), such as:

- Careful definition of the input variable ranges;
- Analysis of the range of the response variable(s);
- Availability of exact train/test datasets;
- Clear definition of function/terminal sets;
- Publication of baseline performance for performance comparison;
- Large test datasets for generalization performance analysis;
- Clear definition of error measures for generalization performance analysis;
- Introduction of controlled noise as simulation of real-world data.

Some real-world datasets have also been suggested and used during the last few years, but problems have also been detected with these [12]. Mostly GP researchers resort to UCI datasets for real-world data [37].

To learn and exploit model structures in black-box optimization for possibly atomic representations of partial solutions, gene-pool optimal mixing and input-space entropy-based building-block learning is an example approach for scalable Genetic Programming [60]; scalability of GP problems, for example in dimension (number of input variables), but also through definition of function/terminal set, is discussed there for artificial problems Order (GP version of OneMax) and Trap (with scalable size of the problem as the maximum binary tree height) and an applied domain of challenges in Boolean Circuits design (Comparator, Even Parity, Majority, and Multiplexer) with scalable number of inputs.

GP can of course be applied to many other areas; some of the most popular in recent years have been gaming and automatic program synthesis, with a range of competitions and workshops (e.g. Evolving levels for Super Mario Bros [51], Virtual Creatures Contest[2], and The General Video Game AI Competition[3]) organized. But in these and other areas, there has not been a concerted effort to provide benchmark data/setups which can be freely and easily used by researchers.

2.3 Modeling Real-world Challenges

Additionally to those already mentioned, real-world challenges for optimization algorithms include:

- knapsack problems (e.g. automatic summarization [39]),

[2] https://virtualcreatures.github.io/vc2018/
[3] http://www.gvgai.net/

Table 1: Sample Symbolic regression problems [43].

F_1 :	$f(x_1, x_2) = \frac{\exp(-(x_1-1)^2)}{1.2+(x_2-2.5)^2}$
F_2 :	$f(x_1, x_2) =$ $\exp(-x_1)x_1^3 \cos(x_1)\sin(x_1)(\cos(x_1)\sin^2 x_1 - 1)(x_2 - 5)$
F_3 :	$f(x_1, x_2, x_3, x_4, x_5) = \frac{10}{5+\sum_{i=1}^{5}(x_i-3)^2}$
F_4 :	$f(x_1, x_2, x_3) = 30\frac{(x_1-1)(x_3-1)}{x_2^2(x_1-10)}$
F_5 :	$f(x_1, x_2) = 6\sin(x_1)\cos(x_2)$
F_6 :	$f(x_1, x_2) = (x_1 - 3)(x_2 - 3) + 2\sin((x_1 - 4)(x_2 - 4))$
F_7 :	$f(x_1, x_2) = \frac{(x_1-3)^4+(x_2-3)^3-(x_2-3)}{(x_2-2)^4+10}$
F_8 :	$f(x_1, x_2) = \frac{1}{1+x_1^{-4}} + \frac{1}{1+x_2^{-4}}$
F_9 :	$f(x_1, x_2) = {x_1}^4 - {x_1}^3 + {x_2}^2/2 - x_2$
F_{10} :	$f(x_1, x_2) = \frac{8}{2+{x_1}^2+{x_2}^2}$
F_{11}	$f(x_1, x_2) = {x_1}^3/5 + {x_2}^3/2 - x_2 - x_1$
F_{12} :	$f(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) =$ $x_1x_2 + x_3x_4 + x_5x_6 + x_1x_7x_9 + x_3x_6x_{10}$
F_{13} :	$f(x_1, x_2, x_3, x_4, x_5) = -5.41 + 4.9\frac{x_4-x_1+x_2/x_5}{3x_4}$
F_{14} :	$f(x_1, x_2, x_3, x_4, x_5, x_6) = (x_5x_6)/(\frac{x_1}{x_2}\frac{x_3}{x_4})$
F_{15} :	$f(x_1, x_2, x_3, x_4, x_5) = 0.81 + 24.3\frac{2x_2+3x_3^2}{4x_4^3+5x_5^4}$
F_{16} :	$f(x_1, x_2, x_3, x_4, x_5) = 32 - 3\frac{\tan(x_1)}{\tan(x_2)}\frac{\tan(x_3)}{\tan(x_4)}$
F_{17} :	$f(x_1, x_2, x_3, x_4, x_5) = 22 - 4.2(\cos(x_1) - \tan(x_2))(\frac{\tanh(x_3)}{\sin(x_4)})$
F_{18} :	$f(x_1, x_2, x_3, x_4, x_5) = x_1x_2x_3x_4x_5$
F_{19} :	$f(x_1, x_2, x_3, x_4, x_5) = 12 - 6\frac{\tan(x_1)}{\exp(x_2)}(x_3 - \tan(x_4))$
F_{20} :	$f(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) = \sum_{i=1}^{5} 1/x_i$
F_{21} :	$f(x_1, x_2, x_3, x_4, x_5) = 2 - 2.1\cos(9.8x_1)\sin(1.3x_5)$

- routing (Traveling Salesperson Problem (TSP) [50], Chinese Postman (CP) [15], path planning [71]),
- scheduling [47] (including job shop and flow shop [56]),
- bioinformatics (including sequencing, alignment, and protein folding [4]),
- cryptography [46], and
- computer vision [66].

2.4 Reflective Budget Algorithm Design

In order to introduce solutions to industry that require process improvement after very latent evaluation of the process, like, e. g., process design in Design for Six Sigma or limited prototype production with necessary high reliability (e.g. aerospace [16]), an optimization algorithm might need to sufficiently quickly self-adjust itself to a black-box challenge. Such a process should take into account the work of designing the optimizer approach itself using the total number of fitness evaluations over a cycle of design and execution. Such a metric might also indirectly measure (in part) the ease and efficiency of use of an optimizer.

Measuring the total number of fitness evaluations is perhaps not an easy task, especially if the fitness evaluations are cheap. However, in a controlled environment, possibly even due to an economic requirement, such constraints might be enforced emergently. The application of an algorithm to a domain would thereby be reflective and influence the quality of the produced results due to a limited fitness evaluation budget allowed for setting up an optimizer [19, 25]. Finally, some challenges might require to run an expensive simulator for many times before designing an optimization algorithm that would autonomously carry out work.

As this paper strives to survey the propositions for benchmarking by proposing a fixed benchmark set with a fixed interface to black-box optimizers, a measure of performance inclination to any function class should therefore also be measured. This would yield the reflectivity measure (influence of the design and tuning of an algorithm to benchmarked evaluation) for an algorithm that might not be suitable for it with an arbitrary black-box challenge. An example function type for this perspective would be an optimizer that writes/adopts a successful optimizer for a black-box benchmark within a limited budget of communications to a benchmark descriptor. Namely, it should be avoided that the designer (human or automaton) is able to call the fitness function sufficiently often during the design phase to extract the encoded information to be optimized. This would yield a grey-box or even a white-box generated optimizer, or in the simplest case, unfairly save an encoded solution itself into the yielded optimizer code.

3 PROPERTIES AND USAGE

Motivations, why certain classes should be included in the perspective, together with their properties (like deception and separability or toy problem label) are presented in this section. Some explanations for enabling shifting and rotation follows.

In the previous section four aspects of BB-DOB challenges are identified. Within each perspective properties for the included function class types are also listed. To use these perspectives and especially the mentioned function class types, this paper suggests to pick a relevant set of representative instances from each of these perspectives and classes. Using the proposed methodology, the picked significant instances are presented in the next section.

The shifting and rotation of benchmarking functions is achieved by transforming the input structure from an optimizer into the input to the benchmarking function by the two mathematical transformations (shift and rotation) before each call to a fitness function in the benchmark. Defining such transformations might be easier for simpler genotypes while for the more advanced ones like trees this might be more challenging.

4 SIGNIFICANT INSTANCES METHODOLOGY

Guidelines for a methodology on how to select significant instances within the listed classes are proposed in this section. The section therefore provides a debate regarding which functions should be included in the benchmarking testbed.

When preparing a benchmark, formulation as well as data should be accessible. On the one hand, if designing a narrow application domain optimizer, then application and performance assessment to real world challenges for such an optimizer usually yields something like a domain specific benchmark, which is not applicable for general black-box re-application of the same optimizer in a different domain. Also, these domain specific benchmarks might not be fully disclosed to test other algorithms. On the other hand, some existing and recognized benchmark sets should be taken into account when preparing a new discrete optimizer benchmark, like the MIPLIB 2010 benchmark set[4]. Therefore, a new benchmark set should enable existing optimizer providers to utilize it, including the example

[4] http://plato.asu.edu/bench.html

solvers like Gurobi[5], CPLEX[6], XPRESS[7], Mosek[8], SCIP[9], CBC[10], GLPK[11], LP_Solve[12], and MATLAB[13]. As a black-box benchmarking suite, the perspectives of the COCO platform [20, 21] could be followed.

The functions referenced in this paper capture the difficulties of combinatorial optimization problems in practice and the references list strives also at the same time to be comprehensible w.r.t. the classes of challenges, so that the resulting algorithm behaviors can be understood or interpreted when using the benchmark. The proposed instances list also considers being scalable with the problem size and non-trivial in the black-box optimization sense, i. e., allow for shifting the optimum to any point as explained in the previous section.

As to how to select the instances, we suggest to choose few representative items from each function class. The instances to be chosen within a function class should be such that the instances thoroughly cover the underlying features of the function class that the class is representing in the terms of challenges it represents. This should foster the benchmarking of optimization algorithms that are designed for black-box functions. The distribution of features in the chosen instances should be non-biased with regard to a function class as well a benchmark as a whole.

5 EXPERIMENTS SETUP

A standardized experimental set-up proposition is presented in this section. This begins the overview of the benchmarking pipeline, explaining how to use the problems discussed in the previous sections, and is followed by post-processing (performance evaluation) and ranking in the next section. The experimental setup guidelines are as follows:

(1) To optimize the instance functions listed in the previous section, a budget of runtime should be used for determining the maximum number of function evaluations allowed. This suggestion is based on the design of algorithms with fixed budget.
(2) The number of independent runs for an optimizer on a specific test instance should be set based on the test instance complexity [1, 2].
(3) The runtime of optimizers should be measured proportional to the time it takes to execute the fitness functions.

6 PERFORMANCE MEASURES

The benchmark should be relevant from a practical perspective and accessible from a theoretical perspective, with a strong focus on discrete search spaces and discrete optimization problems. Gaining insight to mutual advantages and disadvantages of optimization algorithms is vital for the design of improving such algorithms.

To support this and in order for the benchmarks that are developed to be a highly visible focus point for anyone interested in the application of nature-inspired search and optimization heuristics, they should also be made publicly available as part of a website, which is maintained for a sufficiently lasting timespan so that others can submit their performance evaluations for new algorithms.

Rules for measuring the performance over the suggested test functions are presented here. Generally, optimizer performance measures take in account the:

- runtime,
- fixed budget [26], or
- fixed precision.

After collecting the performance measures, Friedman ranking [18] should be presented for the algorithms, using the set of collected values from performance measures. Post-hoc procedures as, e.g., listed in [17] should also be used to report the significance of differences among the algorithms.

As to enable more feedback to the designer of an optimizer, further deep statistics should be welcome, regarding, e.g., evolution and behavioral tracking of the population and memory of an optimizer (e.g., population values). Visualization of these traits using graph plot views should be supported, through drawing the values of these traits or their processing metrices. Examples include plots of fitness convergence, control parameters, optimizer memory (e.g., population), and in the case of generational optimizers, analysis of optimizer memory reading (e.g., generational connections of evolved population members).

In the case of limited budget for algorithm design, e.g., irace [40], fast chess rating system for evolutionary algorithms [58] for parameter tuning [59], or merely a single value performance mark using some aggregative formula [31], should be used instead for the intermediate evaluation of algorithms before producing a final algorithm.

7 RESULTS PRESENTATION METHODS AND FORMATS

To support compatibility, generation of data output, procedures for post-processing, ranking, and presentation formats for the results are presented here.

The tables necessary to report results, should present fitness function values attained at different stages of the optimization runs during several cut-off points, statistically as best, worst, median, average, and standard deviation values, as based on these further comparisons can usually be made between experiments. Furthermore, the evaluation results should be stored in a cloud-compatible format, possibly online as a structured database or archive, comprising the evaluation as well as its corresponding solution values for each of these cut-off points at each run, using binary compilant architecture to enable an Application Binary Interface (ABI) when re-using the experiments at different computers.

Based on the results presentation, competitions could be launched by creating composite functions, e. g., one perspective type competition per year, including some of the set of function classes mentioned.

[5] http://www.gurobi.com/downloads/download-center
[6] https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer
[7] https://www.solver.com/xpress-solver-engine
[8] https://www.mosek.com/
[9] http://scip.zib.de/
[10] https://projects.coin-or.org/Cbc
[11] https://www.gnu.org/software/glpk/
[12] http://lpsolve.sourceforge.net/
[13] https://www.mathworks.com/products/optimization.html

8 CONCLUSIONS

This paper presents a taxonomical identification survey of classes in discrete optimization through a perspective on Black-Box Discrete Optimization Benchmarking (BB-DOB). By listing these classes and providing their properties, usage, and instances, as well as an experimental setup, performance measures, and formats for result representation, it delivers a complete benchmarking pipeline for BB-DOB. It builds upon listing previous successes in benchmarking of optimization algorithms, as well as looking beyond toy problems and specific domain benchmarking, towards challenges for more general discrete optimization algorithms that will be able to tackle new unknown problems as a black box and will be automating performance over those problems.

Further work on Black-Box Discrete Optimization Benchmarking (BB-DOB) perspectives is expected with the BB-DOB@GECCO Workshop. When the WG3 of ImAppNIO wiki[14] will list sufficient benchmarking suggestions, it is expected that a benchmark code package could be facilitated. Fostering of the contributions is also expected through BB-DOB workshop at PPSN 2018 and inviting more researchers to contribute to the benchmark and competitions that will provide black-box algorithms.

ACKNOWLEDGMENTS

This article is based upon work from COST Action CA15140 'Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO)âĂŹ and COST Action IC1406 'High-Performance Modelling and Simulation for Big Data Applications (cHiPSet)' supported by COST (European Cooperation in Science and Technology). The work is also supported in part by the Slovenian Research Agency, Programme Unit P2-0041.

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

[1] Leonardo CT Bezerra, Manuel López-Ibáñez, and Thomas Stützle. 2017. A large-scale experimental evaluation of high-performing multi-and many-objective evolutionary algorithms. *Evolutionary Computation* (2017). DOI: https://doi.org/10.1162/evco_a_00217.

[2] Mauro Birattari. 2004. *On the Estimation of the Expected Performance of a Metaheuristic on a Class of Instances. How Many Instances, How Many Runs?* Technical Report. Technical Report TR/IRIDIA/2004-01, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

[3] Subhodip Biswas, Swagatam Das, Ponnuthurai N Suganthan, and Carlos A Coello Coello. 2014. Evolutionary multiobjective optimization in dynamic environments: A set of novel benchmark functions. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 3192–3199.

[4] Borko Bošković and Janez Brest. 2016. Genetic algorithm with advanced mechanisms applied to the protein structure prediction in a hydrophobic-polar model and cubic lattice. *Applied Soft Computing* 45 (2016), 61–70.

[5] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer. 2006. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Transactions on Evolutionary Computation* 10, 6 (2006), 646–657.

[6] J. Brest, A. Zamuda, B. Bošković, M. Sepesy Maučec, and V. Žumer. 2008. High-dimensional Real-parameter Optimization Using Self-adaptive Differential Evolution Algorithm with Population Size Reduction. In *2008 IEEE World Congress on Computational Intelligence*. IEEE Press, 2032–2039.

[7] J. Brest, A. Zamuda, B. Bošković, M. Sepesy Maučec, and V. Žumer. 2009. Dynamic Optimization using Self-Adaptive Differential Evolution. In *IEEE Congress on Evolutionary Computation 2009*. IEEE Press, 415–422.

[8] Q Chen, B Liu, Q Zhang, JJ Liang, PN Suganthan, and BY Qu. 2014. Problem definition and evaluation criteria for CEC 2015 special session and competition on bound constrained single-objective computationally expensive numerical optimization. *Computational Intelligence Laboratory, Zhengzhou University, China and Nanyang Technological University, Singapore, Technical report* (2014).

[9] S. Das and P. N. Suganthan. 2011. *Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems.* Technical Report. Jadavpur University, India & Nanyang Technological University.

[10] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. 2001. *Scalable Test Problems for Evolutionary Multi-Objective Optimization.* Technical Report 112. Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

[11] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. 2002. Scalable Multi-Objective Optimization Test Problems. In *Congress on Evolutionary Computation (CEC 2002)*, Vol. 1. IEEE Service Center, Piscataway, New Jersey, 825–830.

[12] Grant Dick, Aysha P. Rimoni, and Peter A. Whigham. 2015. A Re-Examination of the Use of Genetic Programming on the Oral Bioavailability Problem. In *Genetic and Evolutionary Computation Conference - GECCO 2015, Madrid, Spain, July 11-15, 2015, Proceedings*, Sara Silva (Ed.). ACM, 1015–1022.

[13] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. 2010. Drift analysis and linear functions revisited. In *2010 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1967–1974.

[14] Juan J Durillo and Antonio J Nebro. 2011. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software* 42, 10 (2011), 760–771.

[15] Jack Edmonds and Ellis L Johnson. 1973. Matching, Euler tours and the Chinese postman. *Mathematical programming* 5, 1 (1973), 88–124.

[16] Dan M Frangopol and Kurt Maute. 2003. Life-cycle reliability-based optimization of civil and aerospace structures. *Computers & structures* 81, 7 (2003), 397–410.

[17] Salvador Garcıa and Francisco Herrera. 2008. An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. *Journal of Machine Learning Research* 9, 2677-2694 (2008), 66.

[18] Salvador García, Daniel Molina, Manuel Lozano, and Francisco Herrera. 2009. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. *Journal of Heuristics* 15, 6 (2009), 617–644.

[19] Youssef Hamadi, Eric Monfroy, and Frédéric Saubion. 2011. An introduction to autonomous search. In *Autonomous Search*. Springer, 1–11.

[20] N. Hansen, A. Auger, O. Mersmann, T. Tušar, and D. Brockhoff. 2016. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *ArXiv e-prints* arXiv:1603.08785 (2016).

[21] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. 2009. *Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions.* Technical Report.

[22] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. 2009. *Real-parameter black-box optimization benchmarking 2009: Noisy functions definitions.* Technical Report.

[23] Jun He, Tianshi Chen, and Xin Yao. 2015. On the easiest and hardest fitness functions. *IEEE Transactions on evolutionary computation* 19, 2 (2015), 295–305.

[24] V. L. Huang, A. K. Qin, K. Deb, E. Zitzler, P. N. Suganthan, J. J. Liang, M. Preuss, and S. Huband. 2007. *Problem Definitions for Performance Assessment & Competition on Multi-objective Optimization Algorithms.* Technical Report TR-07-01. Nanyang Technological University et. al., Singapore.

[25] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2010. Tradeoffs in the empirical evaluation of competing algorithm designs. *Annals of Mathematics and Artificial Intelligence* 60, 1-2 (2010), 65–89.

[26] Thomas Jansen and Christine Zarges. 2014. Performance analysis of randomised search heuristics operating with a fixed budget. *Theoretical Computer Science* 545 (2014), 39–58.

[27] Thomas Jansen and Christine Zarges. 2016. Example Landscapes to Support Analysis of Multimodal Optimisation. In *International Conference on Parallel Problem Solving from Nature*. Springer, 792–802.

[28] Maarten Keijzer. 2003. Improving Symbolic Regression with Interval Arithmetic and Linear Scaling. In *Genetic Programming, 6th European Conference, EuroGP 2003, Essex, UK, April 14-16, 2003, Proceedings (Lecture Notes in Computer Science)*, Conor Ryan, Terence Soule, Maarten Keijzer, Edward Tsang, Riccardo Poli, and Ernesto Costa (Eds.), Vol. 2610. Springer, 70–82.

[29] Michael F. Korns. 2011. Accuracy in Symbolic Regression. In *Genetic Programming Theory and Practice IX*, Rick Riolo, Ekaterina Vladislavleva, and Jason H. Moore (Eds.). Springer, New York, 129–151.

[30] John R. Koza. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA.

[31] Changhe Li, Shengxiang Yang, TT Nguyen, E Ling Yu, Xin Yao, Yaochu Jin, H-G Beyer, and PN Suganthan. 2008. *Benchmark generator for CEC 2009 competition on dynamic optimization.* Technical Report. University of Leicester, UK.

[32] Hui Li, Kalyanmoy Deb, Qingfu Zhang, and P N Suganthan. 2018. *Challenging Novel Many and Multi-Objective Bound Constrained Benchmark Problems.* Technical Report, updated 11 Jan 2018.

[14] http://imappnio.dcs.aber.ac.uk/dokuwiki/doku.php?id=wg3

[33] JJ Liang, BY Qu, and PN Suganthan. 2013. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore* (2013).

[34] JJ Liang, BY Qu, PN Suganthan, and Q Chen. 2014. Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization. *Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore* (2014).

[35] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. C. Coello, and K. Deb. 2005. *Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization.* Technical Report Technical Report Report 2006005. Nanyang Technological University, Singapore.

[36] J. J. Liang, Qu B. Y., and Suganthan P. N. 2013. *Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization.* Technical Report 201212. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China And Nanyang Technological University, Singapore.

[37] M. Lichman. 2013. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. (2013).

[38] QCaQZB Liu, JJ Liang, PN Suganthan, and BY Qu. 2013. Problem definitions and evaluation criteria for computationally expensive single objective numerical optimization. *Computational Intelligence Laboratory and Nanyang Technological University, Zhengzhou and Singapore, Technical Report* (2013).

[39] Elena Lloret and Manuel Palomar. 2012. Text summarisation in progress: a literature review. *Artificial Intelligence Review* 37, 1 (2012), 1–41.

[40] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. 2016. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3 (2016), 43–58.

[41] Rammohan Mallipeddi and Ponnuthurai Nagaratnam Suganthan. 2010. *Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization.* Technical Report. Nanyang Technological University, Singapore.

[42] James McDermott, David R. White, Sean Luke, Luca Manzoni, Mauro Castelli, Leonardo Vanneschi, Wojciech Jaskowski, Krzysztof Krawiec, Robin Harper, Kenneth De Jong, and Una-May O'Reilly. 2012. Genetic programming needs better benchmarks. In *Genetic and Evolutionary Computation - GECCO 2012, Genetic and Evolutionary Computation Conference, Philadelphia, USA, July 7-11, 2012, Proceedings*, Terry Soule et al. (Ed.). ACM, 791–798.

[43] Miguel Nicolau, Alexandros Agapitos, Michael O'Neill, and Anthony Brabazon. 2015. Guidelines for defining benchmark problems in Genetic Programming. In *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015, Proceedings*.

[44] Ludo Pagie and Paulien Hogeweg. 1997. Evolutionary Consequences of Coevolving Targets. *Evolutionary Computation* 5, 4 (1997), 401–418.

[45] Martin Pelikan and David E. Goldberg. 2000. Genetic Algorithms, Clustering, and the Breaking of Symmetry. In *Proceedings of Parallel Problem Solving from Nature (PPSN VI) (Lecture Notes in Computer Science)*, Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan J. Merelo Guervós, and Hans-Paul Schwefel (Eds.), Vol. 1917. Springer, 385–394.

[46] Stjepan Picek, Domagoj Jakobovic, and Una-May O'Reilly. 2017. CryptoBench: benchmarking evolutionary algorithms with cryptographic problems. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 1597–1604.

[47] Michael L Pinedo. 2016. *Scheduling: theory, algorithms, and systems*. Springer.

[48] BY Qu, JJ Liang, ZY Wang, Q Chen, and Ponnuthurai N Suganthan. 2016. Novel benchmark functions for continuous multimodal optimization with comparative results. *Swarm and Evolutionary Computation* 26 (2016), 23–34.

[49] Bo-Yang Qu and Ponnuthurai Nagaratnam Suganthan. 2010. Novel multimodal problems and differential evolution with ensemble of restricted tournament selection. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 1–7.

[50] Gerhard Reinelt. 1991. TSPLIBàĂŤA traveling salesman problem library. *ORSA journal on computing* 3, 4 (1991), 376–384.

[51] Noor Shaker, Miguel Nicolau, Georgios N Yannakakis, Julian Togelius, and Michael O'Neill. 2012. Evolving levels for Super Mario Bros using grammatical evolution. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*. IEEE, 304–311.

[52] J. Shekel. 1971. Test Functions for Multimodal Search Techniques. In *Fifth Annual Princeton Conference on Information Science and Systems*.

[53] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari. 2005. *Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization.* Technical Report. Nanyang Technological University, Singapore, URL http://www.ntu.edu.sg/home/EPNSugan.

[54] K Tang, X Li, PN Suganthan, Zh Yang, and Th Weise. 2009. Benchmark functions for the CEC 2010 special session and competition on large-scale global optimization: Nature Inspired Computation and Applications Laboratory, University of Science and Technology of China. *Applicat. Lab., Univ. Sci. Technol. China, Hefei, China, Tech. Rep* (2009).

[55] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang. 2007. *Benchmark Functions for the CEC 2008 Special Session and Competition on Large Scale Global Optimization.* Technical Report. Nature Inspired Computation and Applications Laboratory et. al., USTC, China, URL http://nical.ustc.edu.cn/cec08ss.php.

[56] M Fatih Tasgetiren, Quan-Ke Pan, PN Suganthan, and Tay Jin Chua. 2011. A differential evolution algorithm for the no-idle flowshop scheduling problem with total tardiness criterion. *International Journal of Production Research* 49, 16 (2011), 5033–5050.

[57] Tea Tusar, Dimo Brockhoff, Nikolaus Hansen, and Anne Auger. 2016. COCO: the bi-objective black box optimization benchmarking (bbob-biobj) test suite. *arXiv preprint arXiv:1604.00359* (2016).

[58] Niki Veček, Marjan Mernik, and Matej Črepinšek. 2014. A chess rating system for evolutionary algorithms: A new method for the comparison and ranking of evolutionary algorithms. *Information Sciences* 277 (2014), 656–679.

[59] Niki Veček, Marjan Mernik, Bogdan Filipič, and Matej Črepinšek. 2016. Parameter tuning with Chess Rating System (CRS-Tuning) for meta-heuristic algorithms. *Information Sciences* 372 (2016), 446–469.

[60] Marco Virgolin, Tanja Alderliesten, Cees Witteveen, and Peter AN Bosman. 2017. Scalable genetic programming by gene-pool optimal mixing and input-space entropy-based building-block learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 1041–1048.

[61] Ekaterina J. Vladislavleva, Guido F. Smits, and Dick den Hertog. 2009. Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *IEEE Transactions on Evolutionary Computation* 13, 2 (2009), 333–349.

[62] David R. White, James McDermott, Mauro Castelli, Luca Manzoni, Brian W. Goldman, Gabriel Kronberger, Wojciech JaÅŻkowski, Una-May OâĂŹReilly, and Sean Luke. 2013. Better GP benchmarks: community survey results and proposals. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 14, 1 (2013), 3–29.

[63] G. Wu, R. Mallipedi, and Suganthan P. N. 2017. *Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization.* Technical Report 201709. National University of Defense Technology, Changsha, Hunan, P.R. China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore.

[64] A. Zamuda. April 2016. Differential Evolution and Large-Scale Optimization Applications. *IGI Global, InfoSci-Videos* (April 2016). https://doi.org/10.4018/978-1-5225-0729-1

[65] A. Zamuda and J. Brest. 2012. Population Reduction Differential Evolution with Multiple Mutation Strategies in Real World Industry Challenges. In *Swarm and Evolutionary Computation (Lecture Notes in Computer Science)*, Leszek Rutkowski, Marcin Korytkowski, Rafal Scherer, Ryszard Tadeusiewicz, Lotfi Zadeh, and Jacek Zurada (Eds.). Springer Berlin / Heidelberg, 154–161.

[66] A. Zamuda and J. Brest. 2014. Vectorized procedural models for animated trees reconstruction using differential evolution. *Information Sciences* 278 (2014), 1–21.

[67] A. Zamuda and J. Brest. 2015. Self-adaptive control parameters' randomization frequency and propagations in differential evolution. *Swarm and Evolutionary Computation* 25 (2015), 72–99.

[68] A. Zamuda, J. Brest, B. Bošković, and V. Žumer. 2007. Differential Evolution for Multiobjective Optimization with Self Adaptation. In *The 2007 IEEE Congress on Evolutionary Computation CEC 2007*. IEEE Press, 3617–3624.

[69] A. Zamuda, J. Brest, B. Bošković, and V. Žumer. 2008. Large Scale Global Optimization Using Differential Evolution with Self Adaptation and Cooperative Co-evolution. In *2008 IEEE World Congress on Computational Intelligence*. IEEE Press, 3719–3726.

[70] A. Zamuda, J. Brest, B. Bošković, and V. Žumer. 2009. Differential Evolution with Self-adaptation and Local Search for Constrained Multiobjective Optimization. In *IEEE Congress on Evolutionary Computation 2009*. IEEE Press, 195–202.

[71] A. Zamuda, J. D. HernÃąndez Sosa, and L. Adler. 2016. Constrained Differential Evolution Optimization for Underwater Glider Path Planning in Sub-mesoscale Eddy Sampling. *Applied Soft Computing* 42 (2016), 93–118.

[72] J. Zhang and A. C. Sanderson. 2009. JADE: adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation* 13, 5 (2009), 945–958.

[73] Qingfu Zhang, Aimin Zhou, S. Zhao, P. N. Suganthan, Wudong Liu, and Santosh Tiwari. 2008. *Multiobjective optimization Test Instances for the CEC 2009 Special Session and Competition.* Technical Report CES-487. University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report.